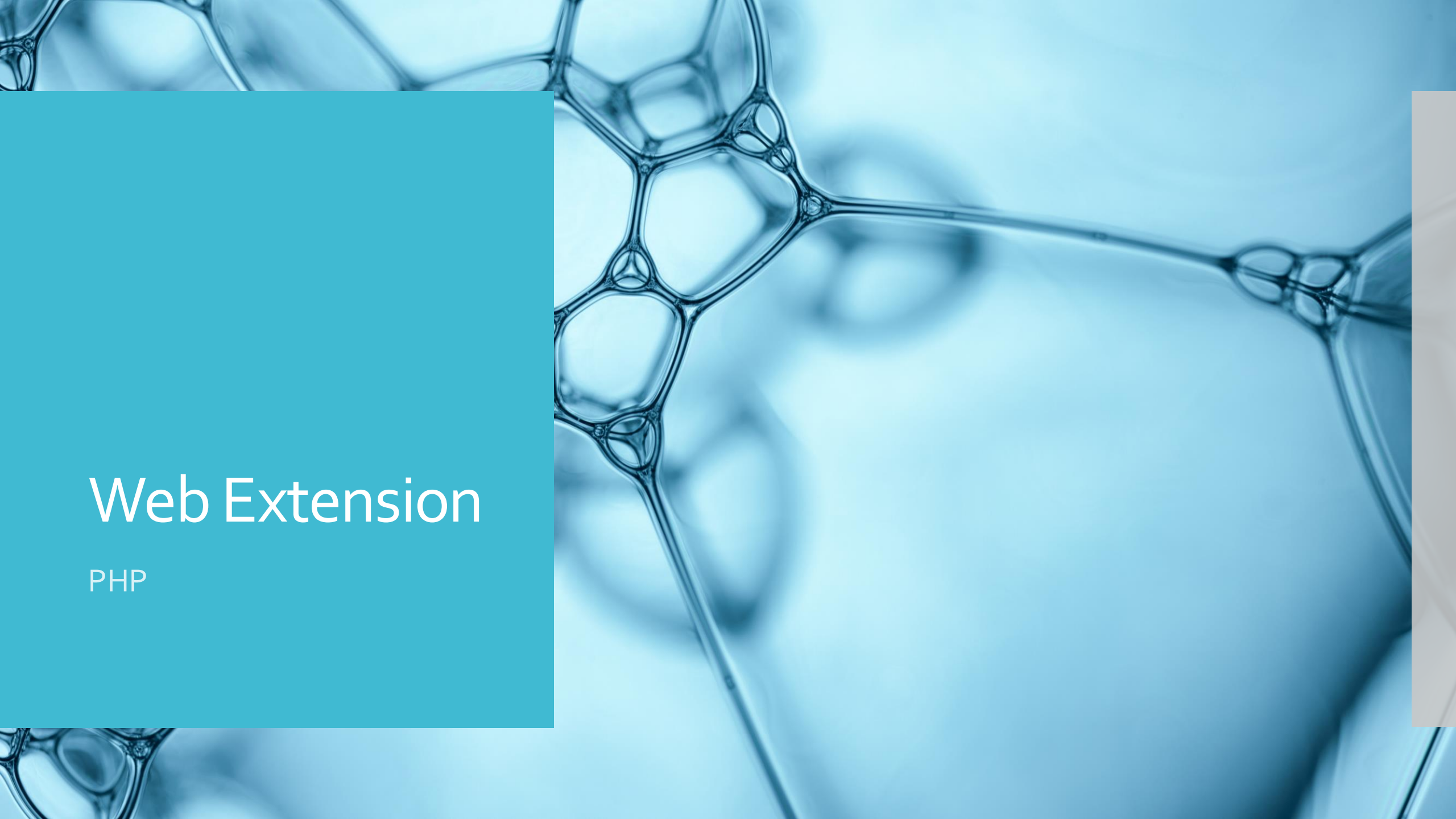


Web Extension

PHP

Content

What is PHP?

- 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

Popular Language

- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)!
- It is also easy enough to be a beginner's first server side language!

What can PHP do?

- Can generate dynamic page content
- Can create, open, read, write, delete, and close files on the server
- Can collect form data
- Can send and receive cookies
- Can add, delete, modify data in your database
- Can be used to control user-access
- Can encrypt data
- With PHP you are not limited to output HTML.
- You can output images, PDF files, and even Flash movies.
- You can also output any text, such as XHTML and XML.

Why PHP??

- Runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Compatible with almost all servers used today (Apache, IIS, etc.)
- Supports a wide range of databases
- It is free. Download it from the official PHP resource: www.php.net
- Easy to learn and runs efficiently on the server side

PHP File

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

PHP Installation

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

PHP Installation

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything.

- Create some .php files
- Place them in your web directory
- Server will automatically parse them for you.

You do not need to compile anything or install any extra tools.

PHP Installation

Set Up PHP on Your Own PC

Steps to use:

1. Install a web server
2. Install PHP
3. Install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

PHP Syntax

Basic PHP syntax

A PHP script can be placed anywhere in the document.

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

```
<?php

// PHP code goes here

?>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Example:

- Simple PHP file, with a PHP script
- uses a built-in PHP function "echo" to output the text on a web page
- PHP statements end with a semicolon

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

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

</body>
</html>
```

Case Sensitivity

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

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Hello World!
Hello World!
Hello World!

Case Sensitivity

All variable names are case-sensitive

For Example

- only the first statement will display the value of the $color variable
- This is because $color, $COLOR, and $coLOR are treated as three different variables

```
<!DOCTYPE html>
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

My car is red
My house is
My boat is

Comments

Both type of Comments are supported:

1. Single Line Comment: // and #
2. Multi line Comment: Block within /* and */

```
<!DOCTYPE html>
<html>
<body>

<?php

// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

?>


</body>
</html>
```

PHP Variable

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

PHP has no command for declaring a variable.

It is created the moment you first assign a value to it.

Rules for PHP variables:

1. A variable starts with the $ sign, followed by the name of the variable
2. A variable name must start with a letter or the underscore character
3. A variable name cannot start with a number
4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
5. Variable names are case-sensitive ($age and $AGE are two different variables)

PHP Variable

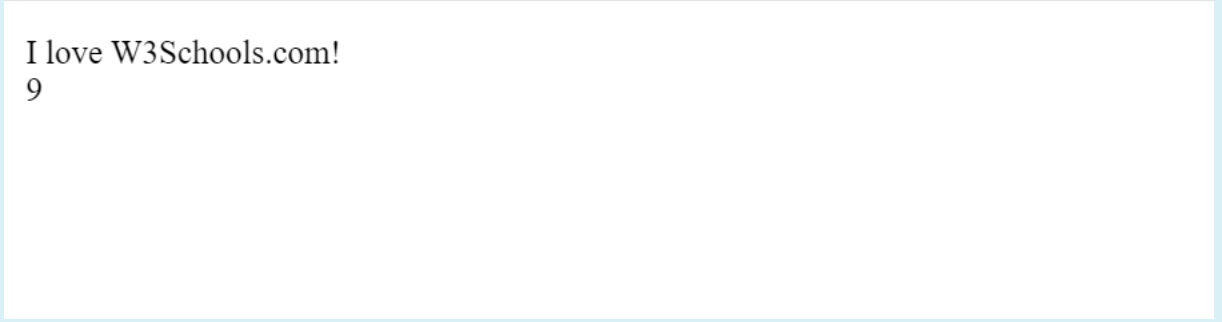
Example:

```
<!DOCTYPE html>
<html><!DOCTYPE html>
<html>
<body>

<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!<br>";

$x = 5;
$y = 4;
echo $x + $y;
?>

</body>
</html>
```



Data Types

Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

1. String
2. Integer
3. Float (floating point numbers - also called double)
4. Boolean
5. Array
6. Object
7. NULL
8. Resource

Strings

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes.
- You can use single or double quotes

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

1. An integer must have at least one digit
2. An integer must not have a decimal point
3. An integer can be either positive or negative
4. Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer.

The PHP var_dump() function returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float.

The PHP var_dump() function returns the data type and value

```
<?php
$x = 10.365;
var_dump($x);
?>
```

Boolean

A Boolean represents two possible states: TRUE or FALSE.

Booleans are often used in conditional testing.

```
$x = true;

$y = false;
```

Arrays

- An array stores multiple values in one single variable.
- In the given example $cars is an array.
- The PHP var_dump() function returns the data type and value

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>

</body>
</html>
```

```
array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
}
```

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.

Object

```
<!DOCTYPE html>
<html>
<body>

<?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();
?>

</body>
</html>
```

My car is a black Volvo!
My car is a red Toyota!

NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- Tip: If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>

</body>
</html>
```

```
NULL
```

Resource

- The special resource type is not an actual data type.
- It is the storing of a reference to functions and resources external to PHP.
- A common example of using the resource data type is a database call.

Operators

Type of Operators

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators
7. Array operators
8. Conditional assignment operators

Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

Assignment Operators

- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Comparison Operators

- The PHP comparison operators are used to compare two values (number or string

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y.

Increment/ Decrement Operators

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Logical Operators

- The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

String Operators

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

Array Operators

Operator	Name	Example	Result	Show it
+	Union	$x + $y	Union of $x and $y	
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs	
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types	
!=	Inequality	$x != $y	Returns true if $x is not equal to $y	
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y	
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y	

Conditional Assignment Operators

- The PHP conditional assignment operators are used to set a value depending on conditions:

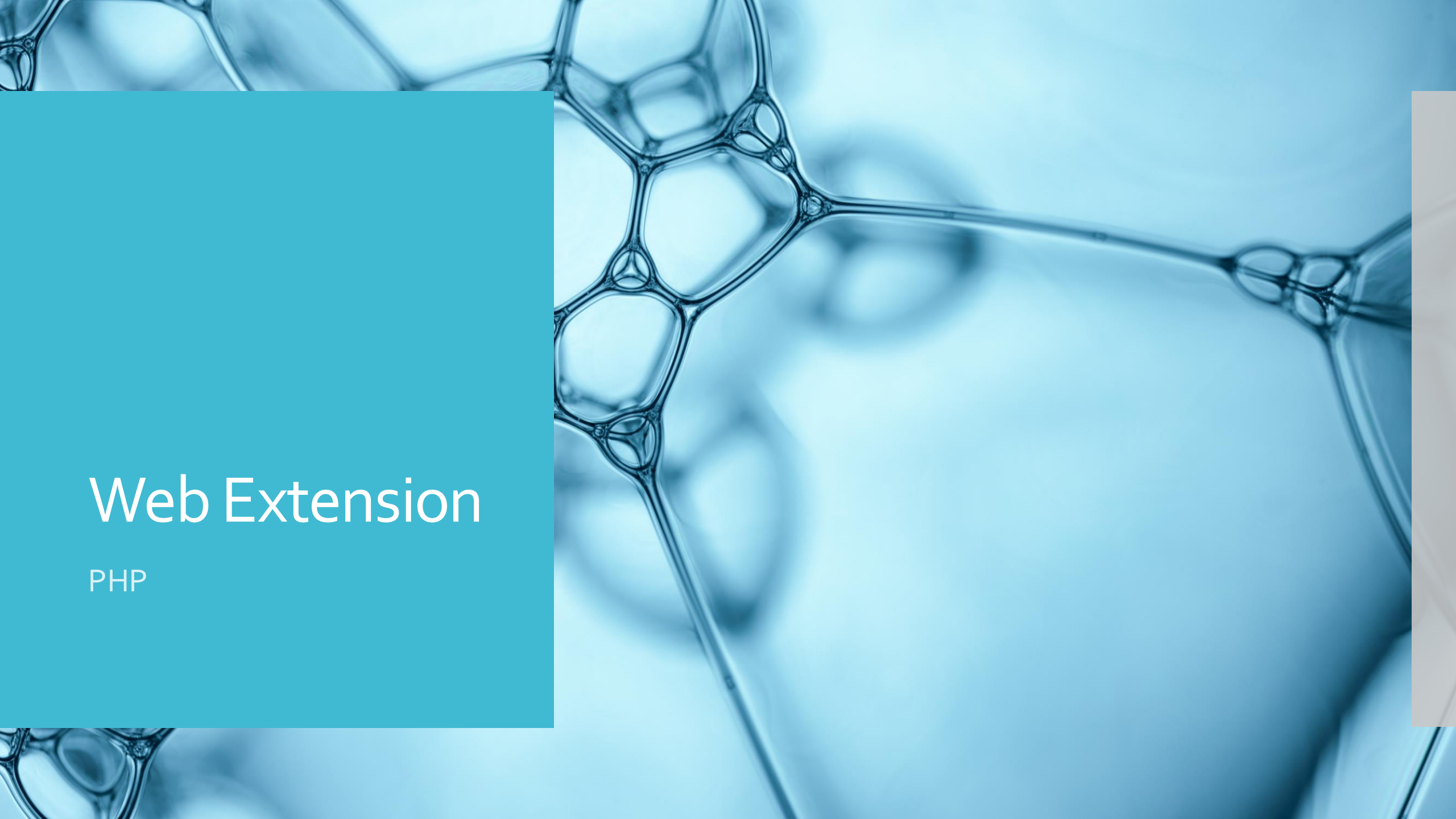
Operator	Name	Example	Result	Show it
?:	Ternary	$x = expr1 ? expr2 : expr3	Returns the value of $x. The value of $x is expr2 if expr1 = TRUE. The value of $x is expr3 if expr1 = FALSE	
??	Null coalescing	$x = expr1 ?? expr2	Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2. Introduced in PHP 7	

Title Lorem Ipsum Dolor

LOREM IPSUM DOLOR

LOREM IPSUM DOLOR

LOREM IPSUM DOLOR



Web Extension

PHP

Content

- Control Structures
- Functions

Control Structures

Conditional Statements

- **if statement -** executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif...else statement -** executes different codes for more than two conditions
- **switch statement -** selects one of many blocks of code to be executed

if Statements

The if statement executes some code if one condition is true.

Syntax

```
if (condition) {
  code to be executed if condition is true;
}
```

Example:

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
}
?>

</body>
</html>
```

if ... else Statements

The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```

Example:

Output "Have a good day!"

if the current time is less than 20,

and "Have a good night!" otherwise:

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>

</body>
</html>
```

if ... elseif ... else Statements

The if...elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

if ... elseif ... else Statements

Example:

Output "Have a good morning!" if the current time is less than 10,

and "Have a good day!" if the current time is less than 20.

Otherwise it will output "Have a good night!":

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>

</body>
</html>
```

switch Statements

Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

switch Statements

```
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor
green!";
}
?>

</body>
</html>
```

Loops

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

while loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
  code to be executed;
}
```

Example

The example below displays the numbers from 1 to 5:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>

</body>
</html>
```

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

do..while loop

The do…while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax:

do {

code to be executed;

} while (condition is true);Example

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 1;

do {
 echo "The number is: $x <br>";
 $x++;
} while ($x <= 5);
?>

</body>
</html>
```

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

for loop

- The for loop - Loops through a block of code a specified number of times.
- The for loop is used when you know in advance how many times the script should run.

Syntax:

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues.

  If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value

for loop

The example below displays the numbers from 0 to 10

Example:

```
<!DOCTYPE html>
<html>
<body>

<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>

</body>
</html>
```

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10

foreach loop

- The foreach loop works only on arrays
- It is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

foreach loop

The following example will output the values of the given array ($colors):

```
<!DOCTYPE html>
<html>
<body>

<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>

</body>
</html>
```

red
green
blue
yellow

foreach loop

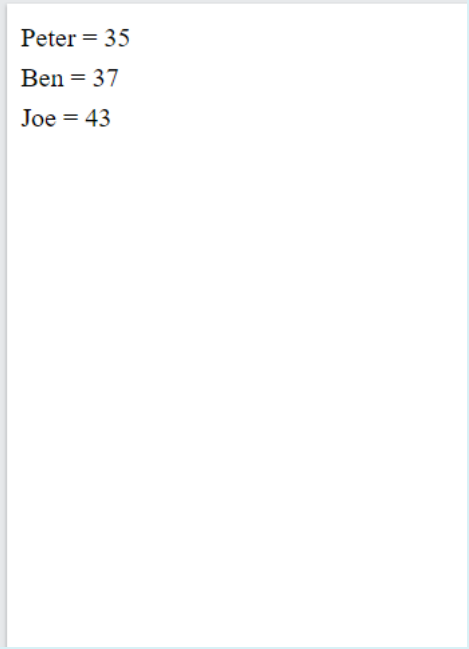
The following example will output both the keys and the values of the given array ($age):

```
<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>

</body>
</html>
```



Functions

Built-in Functions

Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function

- The real power of PHP comes from its functions.
- PHP has more than 1000 built-in functions
- Those functions can be called directly, from within a script, to perform a specific task.
- you can create your own custom functions

User defined Functions

A user-defined function declaration starts with the word function:

A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

Syntax

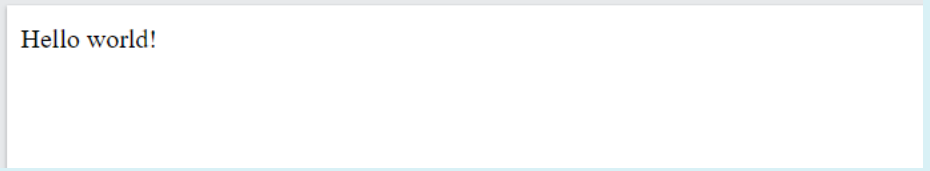
```
function functionName() {
  code to be executed;
}
```

```
<!DOCTYPE html>
<html>
<body>

<?php
function writeMsg() {
  echo "Hello world!";
}

writeMsg();
?>

</body>
</html>
```



Function Arguments

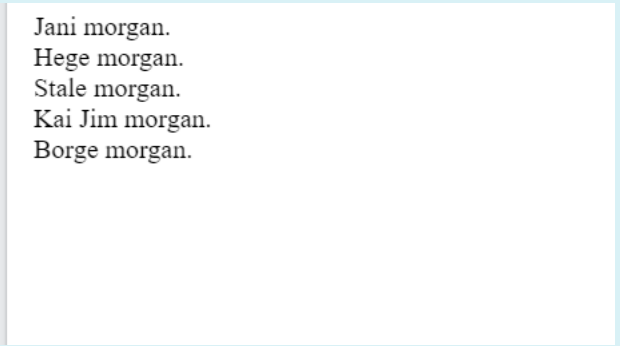
- Information can be passed to functions through arguments.
- An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses.
- You can add as many arguments as you want, just separate them with a comma.

```
<!DOCTYPE html>
<html>
<body>

<?php
function familyName($fname) {
  echo "$fname morgan.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>

</body>
</html>
```



Loosely Typed Language

- 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 us an option to specify the expected data type when declaring a function, and by adding the strict declaration, it will throw a "Fatal Error" if the data type mismatches.
- In the following example we try to send both a number and a string to the function without using strict

```
<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

Loosely Typed Language

- To specify strict we need to set declare(strict_types=1);. This must be on the very first line of the PHP file.
- In the following example we try to send both a number and a string to the function, but here we have added the strict declaration

```
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

Default values

```
<?php declare(strict_types=1); // strict requirement ?>
<!DOCTYPE html>
<html>
<body>

<?php
function setHeight(int $minheight = 50) {
  echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>

</body>
</html>
```

The height is : 350
The height is : 50
The height is : 135
The height is : 80

Returning values

```
<?php declare(strict_types=1); // strict requirement ?>
<!DOCTYPE html>
<html>
<body>

<?php
function sum(int $x, int $y) {
 $z = $x + $y;
 return $z;
}

echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>

</body>
</html>
```

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

Return Type Declaration

- PHP 7 also supports Type Declarations for the return statement.
- Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
- To declare a type for the function return, add a colon (:) and the type right before the opening curly ({)bracket when declaring the function.

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
  return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

Passing arguments by reference

- In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.
- When a function argument is passed by reference, changes to the argument also change the variable that was passed in.
- To turn a function argument into a reference, the & operator is used:

```
<?php
function add_five(&$value) {
 $value += 5;
}

$num = 2;
add_five($num);
echo $num;
?>
```

Title Lorem Ipsum Dolor

LOREM IPSUM DOLOR

LOREM IPSUM DOLOR

LOREM IPSUM DOLOR