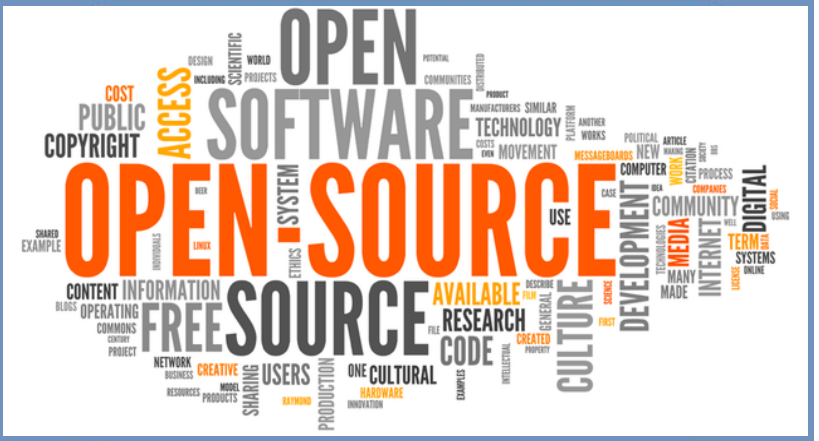
Open Source Software

Van Tham Nguyen

Thuyloi Univerity
175 Tay Son, Dong Da, Ha Noi

Programming with PHP



1. PHP Introduction
2. PHP Syntax
3. PHP Variables

....

1 PHP Introduction

What is a PHP File?

PHP is a recursive acronym for “PHP: Hypertext Preprocessor”

It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

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"

What Can PHP Do?

PHP can generate dynamic page content

PHP can create, open, read, write, delete, and close files on the server

PHP can collect form data

PHP can send and receive cookies

PHP can add, delete, modify data in your database

PHP can be used to control user-access

PHP can encrypt data

PHP Introduction

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5   <?php
6   echo "My first PHP script!";
7   ?>
8
9   </body>
10  </html>
```

2 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.

PHP Case Sensitivity

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

In the example below, all three echo statements below are equal and legal:

`ECHO` is the same as `echo`:

```
<!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!

PHP Case Sensitivity

All variable names are case-sensitive!

Look at the example below; 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>
```

`$COLOR` is *not* same as `$color`:

My car is red
My house is
My boat is

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

- Let others understand your code
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code
- Leave out some parts of your code

PHP supports several ways of commenting:

Comments in PHP

PHP supports several ways of commenting:

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

# This is also a single-line comment

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

```
<?php

// Output "Hello Jone"
echo "Hello Jone";

?>
```

3 PHP Variables

Variables

Variables in a program are used to store some values or data that can be used later in a program.

The variables are also like containers that store character values, numeric values, memory addresses, and strings.

PHP has its own way of declaring and storing variables.

There are a few rules, that need to be followed and facts that need to be kept in mind while dealing with variables in PHP:

Variables

There are a few rules, that need to be followed and facts that need to be kept in mind while dealing with variables in PHP:

(1) Any variables declared in PHP must begin with a dollar sign ($), followed by the variable name.

(2) A variable can have long descriptive names (like $factorial, $even_nos) or short names (like $n or $f or $x)

(3) A variable name can only contain alphanumeric characters and underscores (i.e., ‘a-z’, ‘A-Z’, ‘0-9, and ‘_’) in their name. Even it cannot start with a number.

(4) A constant is used as a variable for a simple value that cannot be changed. It is also case-sensitive.

Variables

There are a few rules, that need to be followed and facts that need to be kept in mind while dealing with variables in PHP:

(4) Assignment of variables is done with the assignment operator, “equal to (=)”. The variable names are on the left of equal and the expression or values are to the right of the assignment operator ‘=’.

(5) One must keep in mind that variable names in PHP names must start with a letter or underscore and no numbers.

Creating (Declaring) PHP Variables

```
$<Variables Name> = <Value>
```

```
$x = 5;
$y = "John"
```

Output Variables

The PHP echo statement is often used to output data to the screen.

```php
<?php
$txt = "Hanoi";
echo "I love $txt!";
?>
```

```php
<?php
$txt = "Hanoi";
echo "I love " . $txt . "!";
?>
```

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

I love Hanoi!

Example

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

<?php
$x = 5;
$y = "John";

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

</body>
</html>
```

5

John

Variable Types

PHP has no command for declaring a variable, and the data type depends on the value of the variable.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

```php
<?php
$x = 5;       // $x is an integer
$y = "John"; // $y is a string

echo $x;
echo $y;
?>
```

Get the Type

To get the data type of a variable, use the `var_dump()` function.

```php
<?php
var_dump(5);
var_dump("John");
var_dump(3.14);
var_dump(true);
var_dump([2, 3, 56]);
var_dump(NULL);
?>
```

```
int(5)
string(4) "John"
float(3.14)
bool(true)
array(3) {
  [0]=>
  int(2)
  [1]=>
  int(3)
  [2]=>
  int(56)
}
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```php
<?php
$x = "John";
echo $x;
?>
```

Assign Multiple Values

You can assign the same value to multiple variables in one line:

```php
<?php
$x = $y = $z = "Fruit";

echo $x;
echo $y;
echo $z;

?>
```

```
FruitFruitFruit
```

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```php
<?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Variable x inside function is:

Variable x outside function is: 5

PHP The global Keyword

The `global` keyword is used to access a global variable from within a function.

To do this, use the `global` keyword before the variables (inside the function):

```php
<?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();  // run function
echo $y; // output the new value for variable $y
?>
```

```
15
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the `static` keyword when you first declare the variable:

```php
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

```
0
1
2
```

4 PHP echo and print Statements

PHP echo and print Statements

`echo` and `print` are more or less the same. They are both used to output data to the screen.

The differences are small:

- `echo` has no return value while `print` has a return value of 1 so it can be used in expressions.

- `echo` can take multiple parameters (although such usage is rare) while `print` can take one argument.

`echo` is marginally faster than `print`.

The PHP echo Statement

The `echo` statement can be used with or without parentheses: `echo` or `echo()`.

Display Text

The following example shows how to output text with the `echo` command (notice that the text can contain HTML markup):

```php
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

PHP is Fun!

Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

Display Variables

The following example shows how to output text and variables with the `echo` statement:

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "Website";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

Learn PHP

Study PHP at Website
9

5 PHP Data Types

PHP Data Types

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

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Getting the Data Type

You can get the data type of any object by using the `var_dump()` function.

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

```php
<?php
$x = 5;
var_dump($x);
?>
```

```
int(5)
```

PHP String

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
<?php
$x = "Hello world!";
$y = 'I love Halong bay!';

var_dump($x);
echo "<br>";
var_dump($y);
?>
```

```
string(12) "Hello world!"
string(18) "I love Halong bay!"
```

PHP Integer

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

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- 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 Integer

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

```
int(5985)
```

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

```
float(10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

```
bool(true)
```

PHP Array

An array stores multiple values in one single variable.

In the following example `$cars` is an array. The PHP `var_dump()` function returns the data type and value:

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

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

PHP 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.

PHP Object

Let's assume we have a class named `Car` that 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.

PHP Object

```php
<?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 .
"!";
  }
}

$myCar = new Car("red", "Volvo");
var_dump($myCar);
?>
```

```
object(Car)#1 (2) { ["color"]=> string(3) "red" ["model"]=> string(5) "Volvo" }
```

PHP 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:

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

```
NULL
```

Change Data Type

If you assign an integer value to a variable, the type will automatically be an integer.

If you assign a string to the same variable, the type will change to a string:

```php
<?php
$x = 5;
var_dump($x);

echo "<br>";

$x = "Hello";
var_dump($x);
?>

<p>Line breaks were added for better readability.</p>
```

```
int(5)
string(5) "Hello"

Line breaks were added for better readability.
```

5 PHP Strings

Strings

Strings in PHP are surrounded by either double quotation marks, or single quotation marks.

```php
<?php
echo "Hello";
print 'Hello';
?>
```

```
HelloHello
```

Double or Single Quotes?

You can use double or single quotes, but you should be aware of the differences between the two.

Double quoted strings perform action on special characters.

E.g. when there is a variable in the string, it returns the value of the variable:

```php
<?php
$x = "John";
echo "Hello $x";
?>
```

Hello John

Double or Single Quotes?

Single quoted strings does not perform such actions, it returns the string like it was written, with the variable name:

```php
<?php
$x = "John";
echo 'Hello $x';
?>
```

Hello $x

String Length

The PHP `strlen()` function returns the length of a string.

```php
<?php
echo strlen("Hello world!");
?>
```

```
12
```

Word Count

The PHP `str_word_count()` function counts the number of words in a string.

```php
<?php
echo str_word_count("Hello world!");
?>
```

```
2
```

Search For Text Within a String

The PHP `strpos()` function searches for a specific text within a string.

If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

```php
<?php
echo strpos("Hello world!", "world");
?>
```

```
6
```

PHP - Modify Strings

- The strtoupper() function returns the string in upper case
- The strtolower() function returns the string in lower case
- The PHP str_replace() function replaces some characters with some other characters in a string
- The PHP strrev() function reverses a string.
- Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

Convert String into Array

The PHP `explode()` function splits a string into an array.

The first parameter of the `explode()` function represents the "separator".

The "separator" specifies where to split the string.

```php
<?php
$x = "Hello World!";
$y = explode(" ", $x);

//Use the print_r() function to display the result:
print_r($y);
?>
```

```
Array ( [0] => Hello [1] => World! )
```

Escape Characters

Code	Result
\'	Single Quote
\"	Double Quote
\$	PHP variables
\n	New Line
\r	Carriage Return
\t	Tab
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

6 PHP Numbers

PHP Numbers

There are three main numeric types in PHP:

- `Integer`
- `Float`
- `Number Strings`

In addition, PHP has two more data types used for numbers:

- `Infinity`
- `NaN`

Variables of numeric types are created when you assign a value to them:

PHP Numbers

```php
<?php
$a = 5;
$b = 5.34;
$c = "25";

var_dump($a);
echo "<br>";
var_dump($b);
echo "<br>";
var_dump($c);
?>

<p>Line breaks were added for better readability.</p>
```

int(5)
float(5.34)
string(2) "25"

Line breaks were added for better readability.

PHP Integers

PHP has the following predefined constants for integers:

- `PHP_INT_MAX` - The largest integer supported
- `PHP_INT_MIN` - The smallest integer supported
- `PHP_INT_SIZE` - The size of an integer in bytes

PHP has the following functions to check if the type of a variable is integer:

- `is_int()`
- `is_integer()` - alias of `is_int()`
- `is_long()` - alias of `is_int()`

PHP Integers

```php
<?php
// Check if the type of a variable is integer
$x = 5985;
var_dump(is_int($x));

echo "<br>";

// Check again...
$x = 59.85;
var_dump(is_int($x));
?>
```

```
bool(true)
bool(false)
```

PHP Floats

PHP has the following predefined constants for floats (from PHP 7.2):

- `PHP_FLOAT_MAX` - The largest representable floating point number
- `PHP_FLOAT_MIN` - The smallest representable positive floating point number
- `PHP_FLOAT_DIG` - The number of decimal digits that can be rounded into a float and back without precision loss
- `PHP_FLOAT_EPSILON` - The smallest representable positive number x, so that x + 1.0 != 1.0

PHP has the following functions to check if the type of a variable is float:

- `is_float()`
- `is_double()` - alias of `is_float()`

PHP Floats

```php
<?php
// Check if the type of a variable is float
$x = 10.365;
var_dump(is_float($x));
?>
```

```
bool(true)
```

PHP Infinity

A numeric value that is larger than `PHP_FLOAT_MAX` is considered infinite.

PHP has the following functions to check if a numeric value is finite or infinite:

- `is_finite()`
- `is_infinite()`

However, the PHP `var_dump()` function returns the data type and value:

PHP Infinity

```php
<?php
// Check if a numeric value is finite or infinite
$x = 1.9e411;
var_dump($x);
?>
```

```
float(INF)
```

PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

- is_nan()

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

```php
<?php
// Invalid calculation will return a NaN value
$x = acos(8);
var_dump($x);
?>
```

```
float(NAN)
```

PHP Numerical Strings

```php
<?php
// Check if the variable is numeric
$x = 5985;
var_dump(is_numeric($x));

echo "<br>";

$x = "5985";
var_dump(is_numeric($x));

echo "<br>";

$x = "59.85" + 100;
var_dump(is_numeric($x));

echo "<br>";

$x = "Hello";
var_dump(is_numeric($x));
?>
```

```
bool(true)
bool(true)
bool(true)
bool(false)
```

PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The `(int)`, `(integer)`, and `intval()` functions are often used to convert a value to an integer.

```php
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;

echo "<br>";

// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

23465

23465

7 PHP Constants

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

To create a constant, use the `define()` function.

```
define(name, value, case-insensitive);
```

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false.

- **Note:** Defining case-insensitive constants was deprecated in PHP 7.3. PHP 8.0 accepts only false, the value true will produce a warning.

Create a PHP Constant

```php
<?php
// case-sensitive constant name
define("GREETING", "Welcome to Hanoi!");
echo GREETING;
?>
```

Welcome to Hanoi!

PHP const Keyword

You can also create a constant by using the `const` keyword.

```php
<?php
const MYCAR = "Volvo";

echo MYCAR;
?>
```

Volvo

Const vs. Define()

- `const` are always case-sensitive
- `define()` has has a case-insensitive option.
- `const` cannot be created inside another block scope, like inside a function or inside an `if` statement.
- `define` can be created inside another block scope.

PHP Predefined Constants

PHP has nine predefined constants that change value depending on where they are used, and therefor they are called "magic constants".

These magic constants are written with a double underscore at the start and the end, except for the ClassName::class constant.

Magic Constants

Constant	Description
__CLASS__	If used inside a class, the class name is returned.
__DIR__	The directory of the file.
__FILE__	The file name including the full path.
__FUNCTION__	If inside a function, the function name is returned.
__LINE__	The current line number.
__METHOD__	If used inside a function that belongs to a class, both class and function name is returned.
__NAMESPACE__	If used inside a namespace, the name of the namespace is returned.
__TRAIT__	If used inside a trait, the trait name is returned.
ClassName::class	Returns the name of the specified class and the name of the namespace, if any.

Magic Constants

Constant	Description
__CLASS__	If used inside a class, the class name is returned.
__DIR__	The directory of the file.
__FILE__	The file name including the full path.
__FUNCTION__	If inside a function, the function name is returned.
__LINE__	The current line number.
__METHOD__	If used inside a function that belongs to a class, both class and function name is returned.
__NAMESPACE__	If used inside a namespace, the name of the namespace is returned.
__TRAIT__	If used inside a trait, the trait name is returned.
ClassName::class	Returns the name of the specified class and the name of the namespace, if any.

8 PHP Operators

PHP const Keyword

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

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

PHP Assignment Operators

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

PHP Comparison Operators

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. Introduced in PHP 7.

PHP Increment / Decrement Operators

Operator	Same as...	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

PHP Logical Operators

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

PHP String Operators

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

PHP Array Operators

Operator	Name	Example	Result
+	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

PHP Conditional Assignment Operators

Operator	Name	Example	Result
?:	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

9 PHP if and switch Statements

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following 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

PHP - The if-else Statement

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

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

PHP - The if...elseif...else Statement

```
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;
}
```

```
<?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!";
}
?>
```

The hour (of the server) is 10, and will give the following message:
Have a good day!

The PHP switch Statement

```
switch (expression) {
  case label1:
    //code block
    break;
  case label2:
    //code block;
    break;
  case label3:
    //code block
    break;
  default:
    //code block
}
```

This is how it works:

- The *expression* is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The `break` keyword breaks out of the switch block
- The `default` code block is executed if there is no match

The PHP switch Statement

```php
<?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>
```

Your favorite color is red!

10 PHP Loops

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- `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

PHP while Loop

The `while` loop - Loops through a block of code as long as the specified condition is true.

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

```
while (condition):
// code to be executed if condition is true;
endwhile;
```

PHP do while Loop

The `do...while` loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.

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

The PHP for Loop

The `for` loop is used when you know how many times the script should run.

```
for (expression1, expression2, expression3) {
  // code block
}
```

This is how it works:

- *expression1* is evaluated once
- *expression2* is evaluated before each iterarion
- *expression3* is evaluated after each iterarion

PHP foreach Loop

The most common use of the `foreach` loop, is to loop through the items of an array.

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

```
foreach ($array as $key => $element) {
    //code to be executed
}
```

PHP foreach Loop

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

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

red

green

blue

yellow

PHP foreach Loop

```php
<?php
    //declare array
    $employee = array (
        "Name" => "Alex",
        "Email" => "alex_jtp@gmail.com",
        "Age" => 21,
        "Gender" => "Male"
    );

    //display associative array element through foreach loop
    foreach ($employee as $key => $element) {
        echo $key . " : " . $element;
        echo "</br>";
    }
?>
```

Name : Alex
Email : alex_jtp@gmail.com
Age : 21
Gender : Male

11 PHP Functions

PHP Functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

HP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the PHP built-in functions.

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own 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.

Create a Function

```
function functionname([$<Var1>,….,$<Varn>]){
//code to be executed
}
```

```
function functionname([$<Var1>,….,$<Varn>]){
//code to be executed
return $<Value>;
}
```

Call a Function

To call the function, just write its name followed by parentheses ():

```php
// function along with three parameters
function proGeek($num1, $num2, $num3)
{
    $product = $num1 * $num2 * $num3;

    return $product; //returning the product
}

// storing the returned value
$retValue = proGeek(2, 3, 5);
echo "The product is $retValue";

?>
```

```
The product is 30
```

PHP 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.

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

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

</body>
</html>
```

Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.

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:

Passing Arguments by Reference

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

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

```
7
```

Variable Number of Arguments

By using the ... operator in front of the function parameter, the function accepts an unknown number of arguments. This is also called a variadic function.

The variadic function argument becomes an array.

```php
<?php
function sumMyNumbers(...$x) {
  $n = 0;
  $len = count($x);
  for($i = 0; $i < $len; $i++) {
    $n += $x[$i];
  }
  return $n;
}

$a = sumMyNumbers(5, 2, 6, 2, 7, 7);
echo $a;
?>
```

```
29
```

PHP is a Loosely Typed Language

In the examples above, notice that we did not have to tell PHP which data type the variable is.

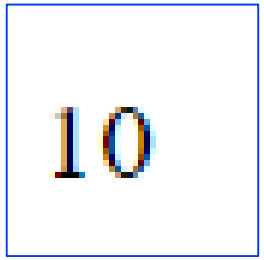
PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

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 is a Loosely Typed Language

```php
<?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
?>
```



PHP Return Type Declarations

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.

In the following example we specify the return type for the function:

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

```
6.4
```

12 PHP Arrays

What is an Array?

An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.

PHP Array Types

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with a numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

Array Items

Array items can be of any data type.

The most common are strings and numbers (`int, float`), but array items can also be objects, functions or even arrays.

You can have different data types in the same array.

```php
<?php
// function example:
function myFunction() {
  echo "This text comes from a function";
}

// create array:
$myArr = array("Volvo", 15, ["apples", "bananas"], myFunction);

// calling the function from the array item:
$myArr[3]();
?>
```

This text comes from a function

PHP Indexed Arrays

In indexed arrays each item has an index number.

By default, the first item has index 0, the second item has item 1, etc.

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

echo $cars[0]
?>
```

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

Access Indexed Arrays

To access an array item you can refer to the index number.

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo $cars[0];
?>
```

```
Volvo
```

Access Indexed Arrays

To access an array item you can refer to the index number.

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo $cars[0];
?>
```

```
Volvo
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a `foreach` loop, like this:

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

foreach ($cars as $x) {
  echo "$x <br>";
}
?>
```

Volvo

BMW

Toyota

Index Number

The key of an indexed array is a number, by default the first item is 0 and the second is 1 etc., but there are exceptions.

New items get the next index number, meaning one higher than the highest existing index.

And if you use the `array_push()` function to add a new item, the new item will get the index 3:

```php
<?php
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";

array_push($cars, "Ford");
var_dump($cars);
?>
```

```
array(4) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
  [3]=>
  string(4) "Ford"
}
```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

```php
<?php
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);
var_dump($car);
?>
```

```
array(3) {
  ["brand"]=>
  string(4) "Ford"
  ["model"]=>
  string(7) "Mustang"
  ["year"]=>
  int(1964)
}
```

Access Associative Arrays

To access an array item you can refer to the key name.

```php
<?php
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);
echo $car["model"];
?>
```

Mustang

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a `foreach` loop, like this:

```php
<?php
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

foreach ($car as $x => $y) {
  echo "$x: $y <br>";
}
?>
```

```
brand: Ford
model: Mustang
year: 1964
```

Create Array

(1) Using the array() function:

```php
$cars = array("Volvo", "BMW", "Toyota");
```

(2) Using a shorter syntax by using the [] brackets:

```php
$cars = ["Volvo", "BMW", "Toyota"];
```

(3) Using Line breaks are not important, so an array declaration can span multiple lines:

```php
$cars = [
  "Volvo",
  "BMW",
  "Toyota"
];
```

Create Array

(4) Array Keys:When creating indexed arrays the keys are given automatically, starting at 0 and increased by 1 for each item, so the array above could also be created with keys:

```
$cars = [
  0 => "Volvo",
  1 => "BMW",
  2 =>"Toyota"
];
```

(5) Declare Empty Array

```
$cars = [];
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Create Array

(6) Mixing Array Keys:

```php
$myArr = [];
$myArr[0] = "apples";
$myArr[1] = "bananas";
$myArr["fruit"] = "cherries";
```

PHP Add Array Items

(1) Add Array Item: To add items to an existing array, you can use the bracket [] syntax.

```php
$fruits = array("Apple", "Banana", "Cherry");
$fruits[] = "Orange";
```

(2) Associative Arrays: To add items to an associative array, or key/value array, use brackets [] for the key, and assign value with the = operator..

```php
$cars = array("brand" => "Ford", "model" => "Mustang");
$cars["color"] = "Red";
```

PHP Add Array Items

(3) Add Multiple Array Items: To o add multiple items to an existing array, use the array_push() function

```php
$fruits = array("Apple", "Banana", "Cherry");
array_push($fruits, "Orange", "Kiwi", "Lemon");
```

(4) Add Multiple Items to Associative Arrays: To add multiple items to an existing array, you can use the += operator.

```php
$cars = array("brand" => "Ford", "model" => "Mustang");
$cars += ["color" => "red", "year" => 1964];
```

Remove Array Item

(1) To remove an existing item from an array, you can use the array_splice() function.

With the array_splice() function you specify the index (where to start) and how many items you want to delete.

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
array_splice($cars, 1, 1);
echo $cars[0],",",$cars[1];
?>
```

```
Volvo,Toyota
```

Remove Array Item

(2) Using the unset Function. The unset() function does not re-arrange the indexes, meaning that after deletion the array will no longer contain the missing indexes. .

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
unset($cars[1]);
echo $cars[0],",",$cars[2],"\n";
echo $cars[0],",",$cars[1];
?>
```

```
Volvo,Toyota
Volvo,
```

Remove Array Item

(3) Remove Multiple Array Items:

- To remove multiple items, the array_splice() function takes a length parameter that allows you to specify the number of items to delete.
- The unset() function takes a unlimited number of arguments, and can therefor be used to delete multiple array items:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
array_splice($cars, 1, 2);
echo $cars[0],$cars[1],$cars[2]
?>
```

```
Volvo
```

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
unset($cars[0], $cars[1]);
echo $cars[0],$cars[1],$cars[2]
?>
```

```
Toyota
```

Remove Array Item

(4) Remove Item From an Associative Array:

To remove items from an associative array, you can use the unset() function.

Specify the key of the item you want to delete.

```php
<?php
$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);
unset($cars["model"]);
echo $cars["model"],$cars["brand"]
?>
```

```
Ford
```

Remove Array Item

(5) Using the array_diff Function:

You can also use the array_diff() function to remove items from an associative array. This function returns a new array, without the specified items.

```php
<?php
$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);
$newarray = array_diff($cars, ["Mustang", 1964]);
echo $newarray["model"],$newarray["brand"]
?>
```

```
Ford
```

Remove Array Item

(6) Remove the Last Item: The array_pop() function removes the last item of an array.

(7) Remove the First Item: The array_shift() function removes the first item of an array.

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
array_shift($cars);
array_pop($cars);
echo $cars[0],$cars[1],$cars[2]
?>
```

```
BMW
```

Any

Question