Course: PHP from scratch

by Sergey Podgornyy

PHP basics

W EB ACADEMY
PROGRAMMING COURSES

About me

Sergey Podgornyy

Full-Stack Web Developer

WebbyLab

and

ignite software outsourcing

zend CERTIFIED PHP ENGINEER

Linux Professional Institute LPIC-1

Overview

- How PHP works? What is php.ini?
- PHP tags and comments
- Error reporting / error_reporting in php.ini
- Variables
- Output data: `echo`, `print`, `HEREDOC`, `NOWDOC`
- Magic constants
- Operators in PHP
- Data types
- Comparison operators
- Logic operators
- Conditional Statements

What is PHP?

PHP: Hypertext Preprocessor

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP

The configuration file

The configuration file (`php.ini`) is read when PHP starts up. For the server module versions of PHP, this happens only once when the web server is started. For the CGI and CLI versions, it happens on every invocation

www.web-academy.com.ua

Error reporting

`error_reporting` — sets which PHP errors are reported

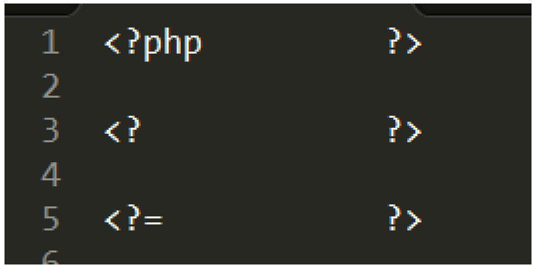
PHP 5.3 or later, the default value is `E_ALL` & `~E_NOTICE` & `~E_STRICT` & `~E_DEPRECATED`

This setting does not show `E_NOTICE`, `E_STRICT` and `E_DEPRECATED` level errors. You may want to show them during development

PHP Syntax

- A PHP script can be placed anywhere in the document
- The default file extension for PHP files is "`.php`"
- A PHP script starts with `<?php` and ends with `?>`

✓ ✗ ✓



```
<?php          ?>

<?             ?>

<?=            ?>
```

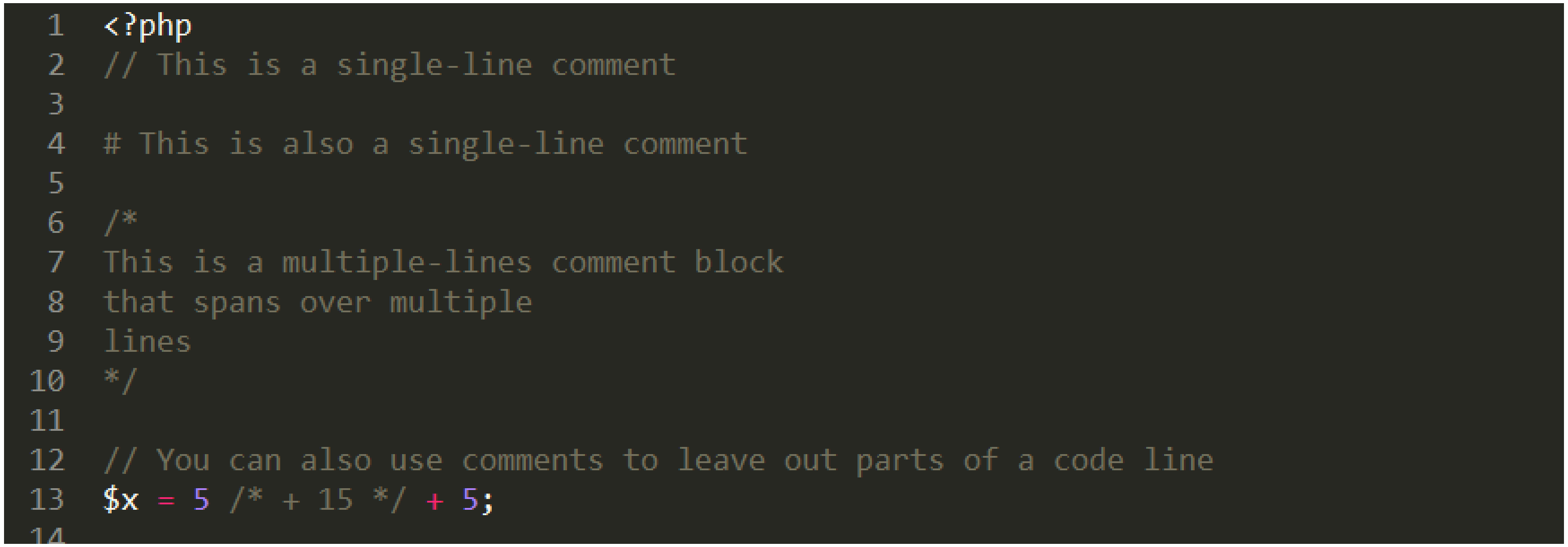
available if `short_open_tag` in `php.ini`

available if `short_open_tag` in `php.ini`

Comments

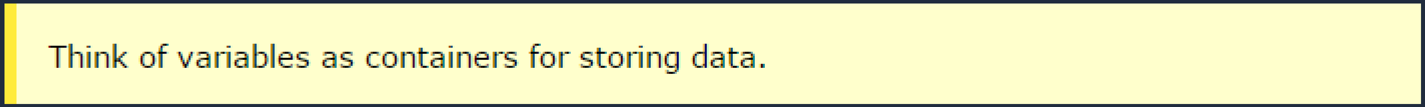
- Let others understand what you are doing
- Remind yourself of what you did

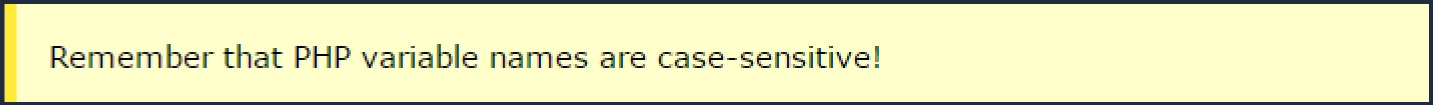
PHP supports several ways of commenting:



```php
<?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
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
```

Variables


Think of variables as containers for storing data.


Remember that PHP variable names are case-sensitive!

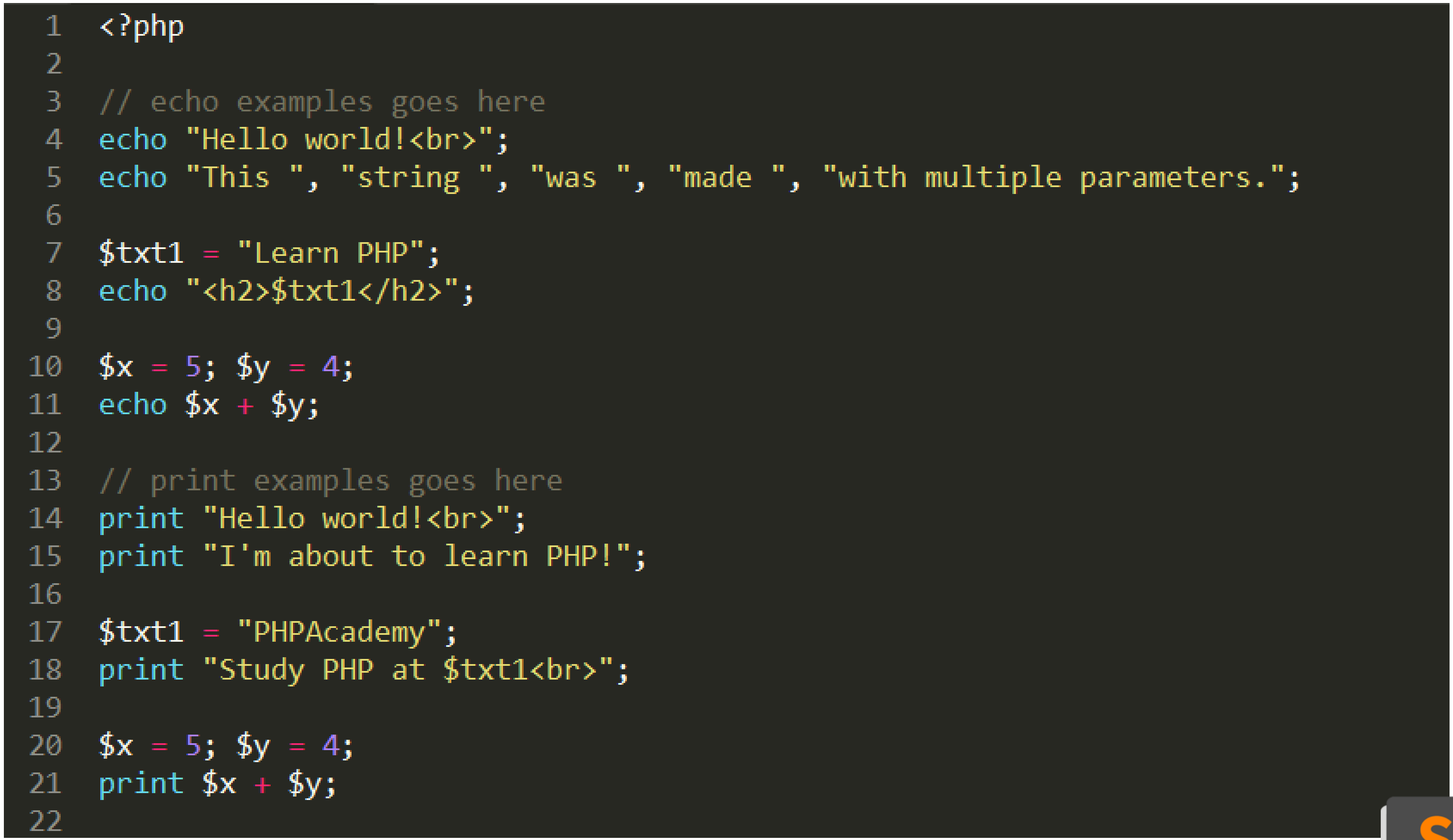
- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names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

`echo` and `print` statements

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

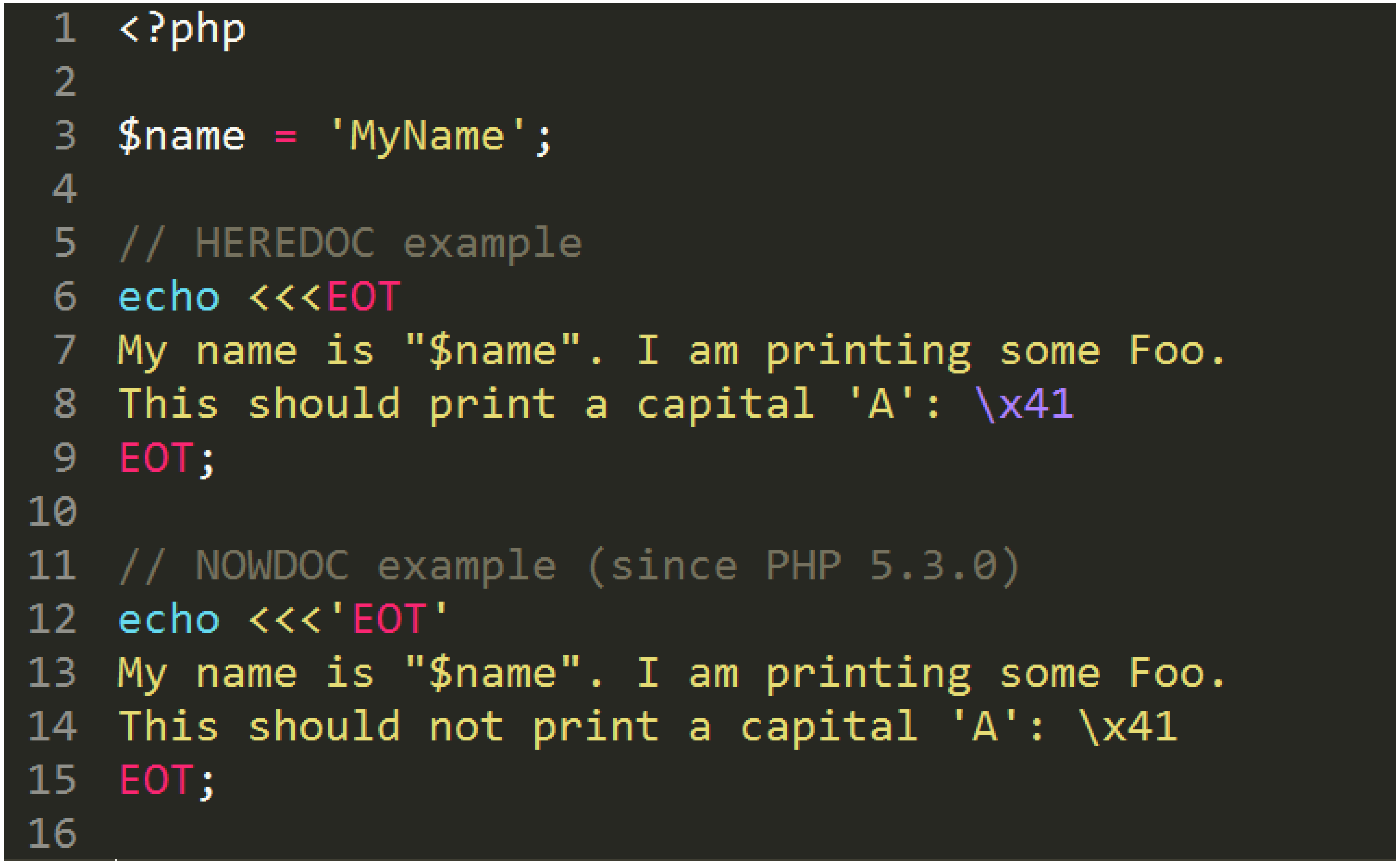
The differences are small:
`echo` has no return value
while `print` has a return value of 1 so it can be used in expressions

echo and print statements



```php
<?php

// echo examples goes here
echo "Hello world!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";

$txt1 = "Learn PHP";
echo "<h2>$txt1</h2>";

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

// print examples goes here
print "Hello world!<br>";
print "I'm about to learn PHP!";

$txt1 = "PHPAcademy";
print "Study PHP at $txt1<br>";

$x = 5; $y = 4;
print $x + $y;

```

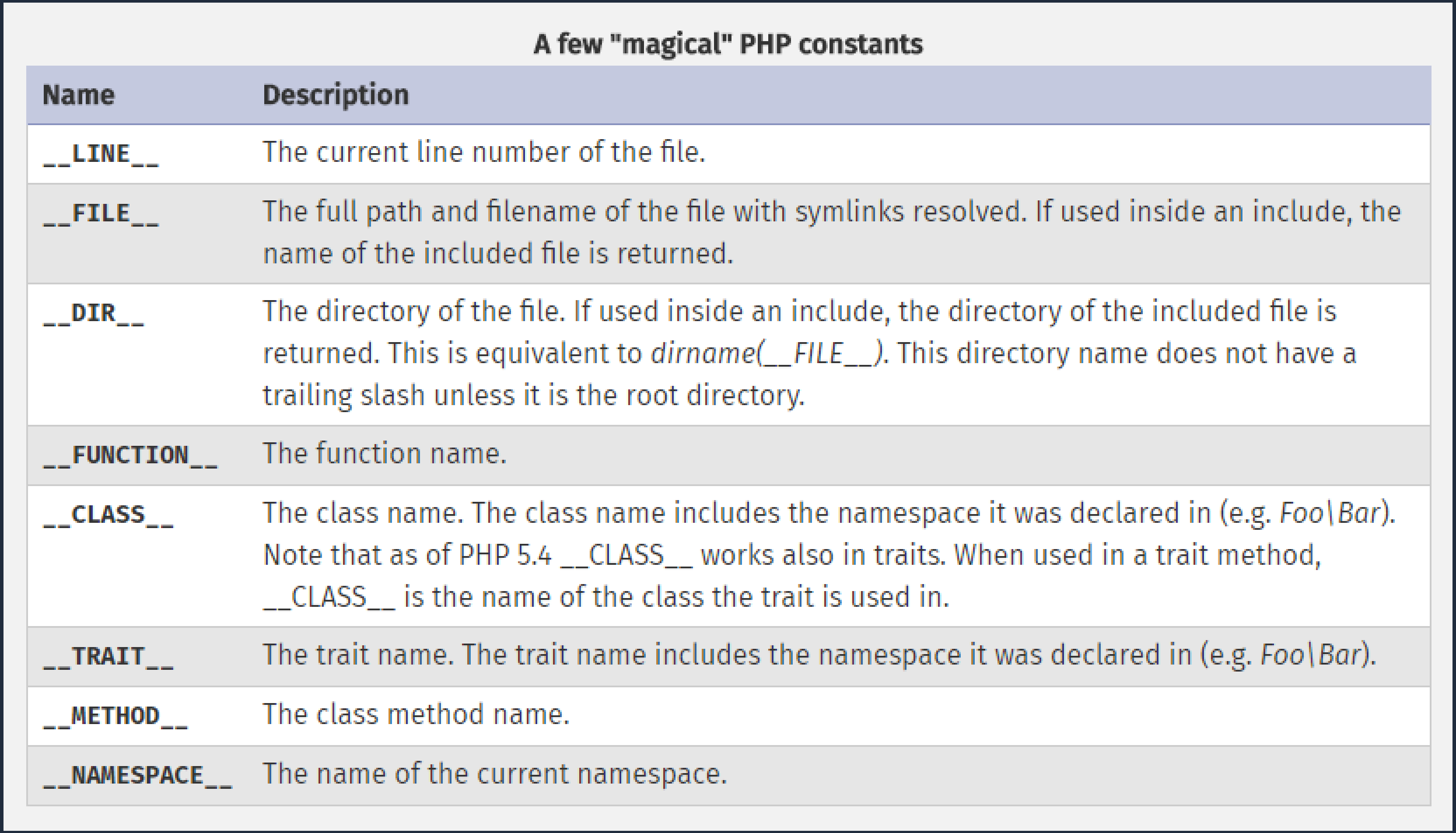
HEREDOC & NOWDOC



```php
<?php

$name = 'MyName';

// HEREDOC example
echo <<<EOT
My name is "$name". I am printing some Foo.
This should print a capital 'A': \x41
EOT;

// NOWDOC example (since PHP 5.3.0)
echo <<<'EOT'
My name is "$name". I am printing some Foo.
This should not print a capital 'A': \x41
EOT;

```

www.web-academy.com.ua

Predefined magic constants



A few "magical" PHP constants

Name	Description
`__LINE__`	The current line number of the file.
`__FILE__`	The full path and filename of the file with symlinks resolved. If used inside an include, the name of the included file is returned.
`__DIR__`	The directory of the file. If used inside an include, the directory of the included file is returned. This is equivalent to *dirname(__FILE__)*. This directory name does not have a trailing slash unless it is the root directory.
`__FUNCTION__`	The function name.
`__CLASS__`	The class name. The class name includes the namespace it was declared in (e.g. *Foo\Bar*). Note that as of PHP 5.4 __CLASS__ works also in traits. When used in a trait method, __CLASS__ is the name of the class the trait is used in.
`__TRAIT__`	The trait name. The trait name includes the namespace it was declared in (e.g. *Foo\Bar*).
`__METHOD__`	The class method name.
`__NAMESPACE__`	The name of the current namespace.

www.web-academy.com.ua

Data Types

PHP supports the following data types:

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

Data Types

`isset` — Determine if a variable is set and is not `NULL`

`empty` — Determine whether a variable is empty

`gettype` — Get the type of a variable

Returns the type of the PHP variable var. For type checking, use `is_*` functions

Data Types

- `is_array()` - Finds whether a variable is an array
- `is_bool()` - Finds out whether a variable is a boolean
- `is_callable()` - Verify that the contents of a variable can be called as a function
- `is_float()` - Finds whether the type of a variable is float
- `is_int()` - Find whether the type of a variable is integer
- `is_null()` - Finds whether a variable is NULL
- `is_numeric()` - Finds whether a variable is a number or a numeric string
- `is_object()` - Finds whether a variable is an object
- `is_resource()` - Finds whether a variable is a resource
- `is_scalar()` - Finds whether a variable is a scalar
- `is_string()` - Find whether the type of a variable is string

Operators

PHP divides the operators in the following groups:

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

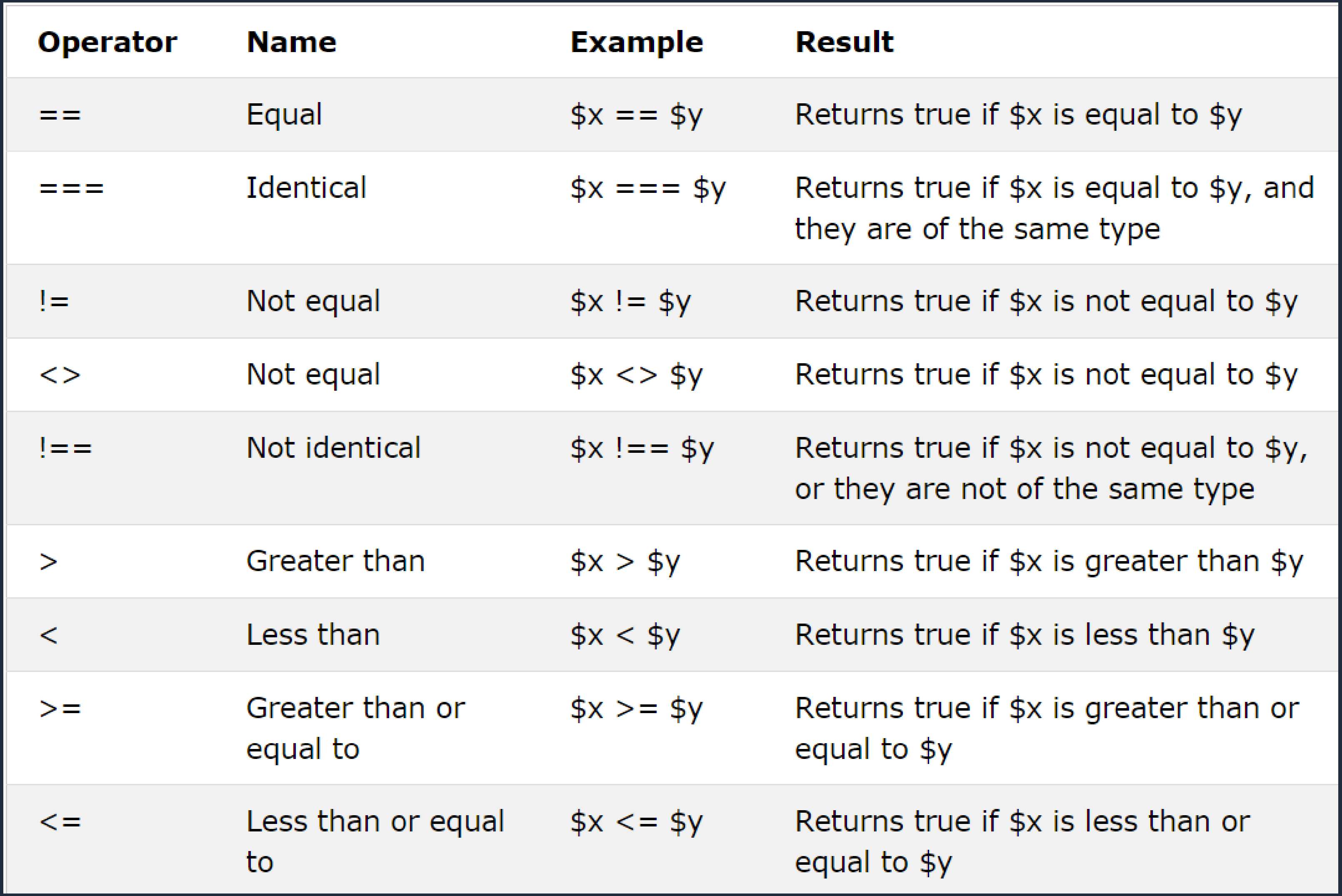
Arithmetic Operators

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 (Introduced in PHP 5.6)

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

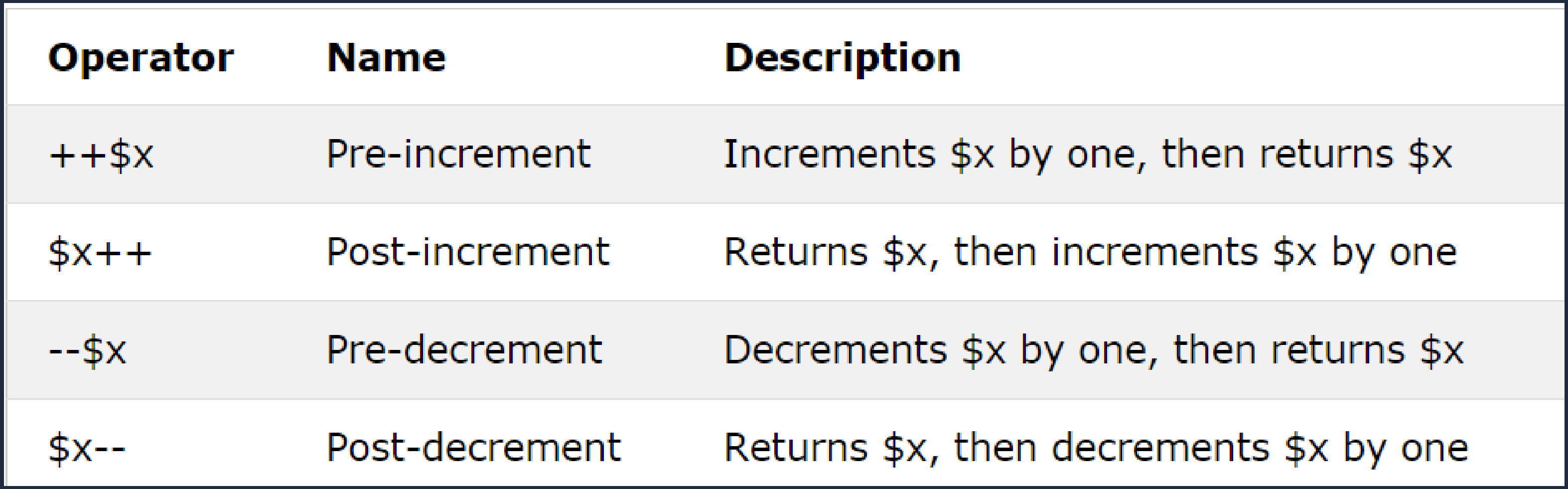
Comparison Operators



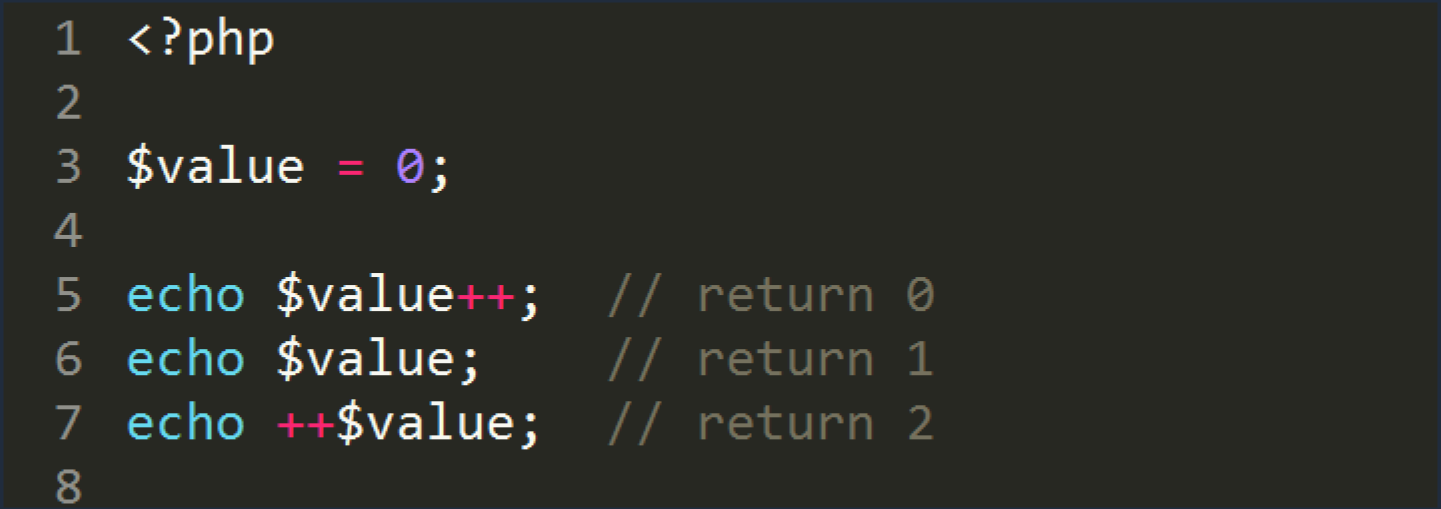
Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$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

www.web-academy.com.ua

Increment / Decrement Operators



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



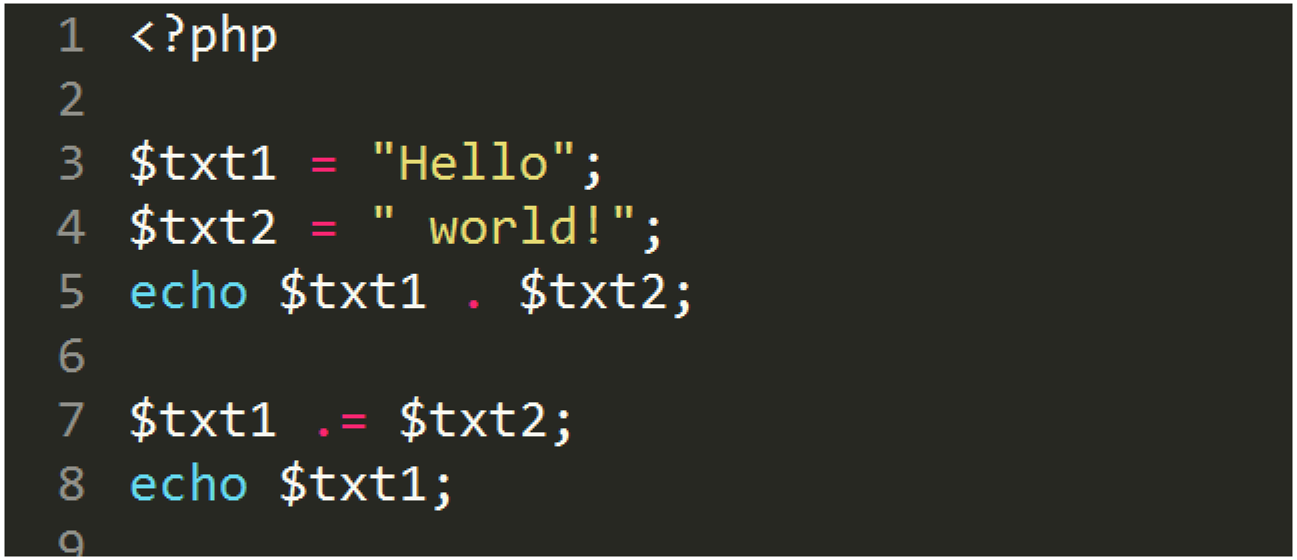
```php
<?php

$value = 0;

echo $value++;  // return 0
echo $value;    // return 1
echo ++$value;  // return 2

```

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

String Operators

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



```php
<?php

$txt1 = "Hello";
$txt2 = " world!";
echo $txt1 . $txt2;

$txt1 .= $txt2;
echo $txt1;

```

Conditional Statements

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

if statement syntax

```
$time = date("H");

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

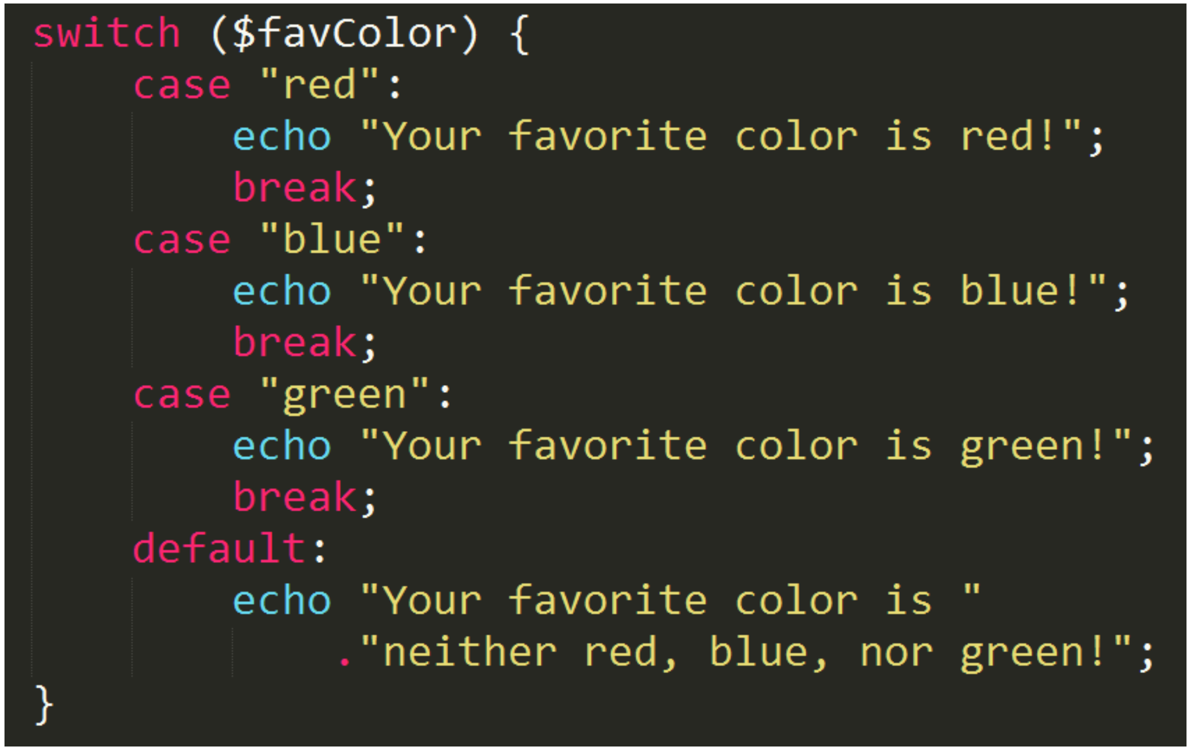
The block is executed if the condition is true

```
<?php if ($time < "10") : ?>
    <h1>Have a good morning!</h1>
<?php elseif ($time < "20") : ?>
    <h1>Have a good day!</h1>
<?php else : ?>
    <h1>Have a good night!</h1>
<?php endif; ?>
```

The syntax that is used for view

`switch...case` statement syntax

PHP continues to execute the statements until the end of the **`switch`** block, or the first time it sees a **`break`** (or **`return`**) statement



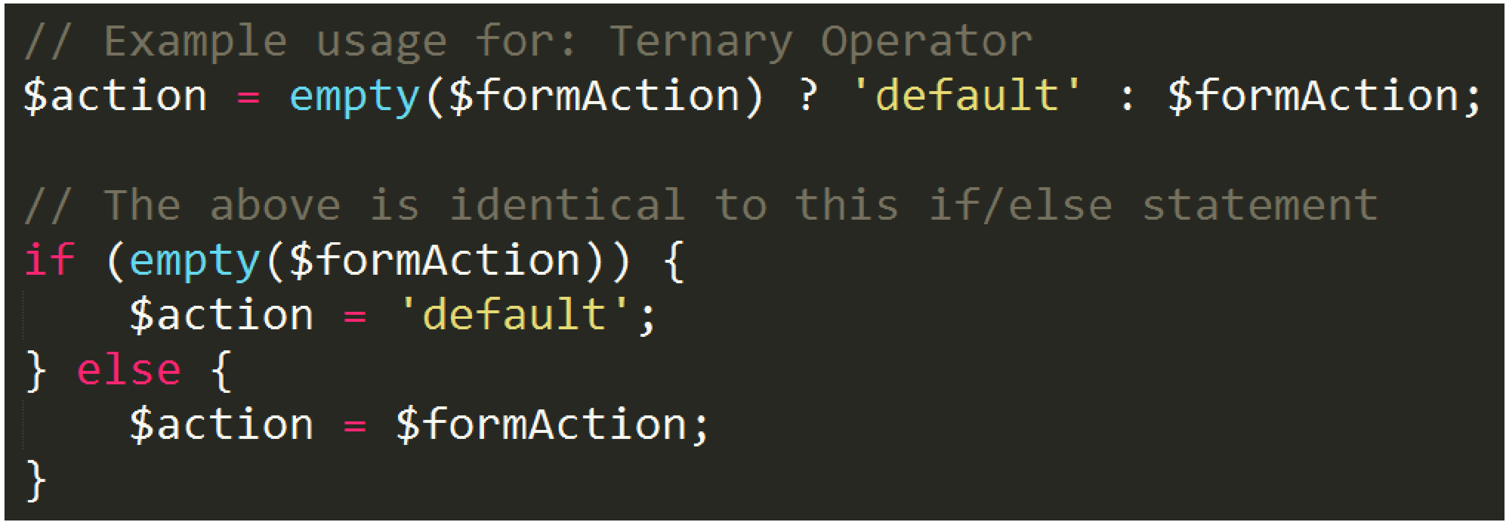
```
switch ($favColor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color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

A special case is the **`default`** case. This case matches anything that wasn't matched by the other cases

Ternary Operator

`(expr1) ? (expr2) : (expr3)`

This expression evaluates to `expr2` if `expr1` evaluates to `TRUE`, and `expr3` if `expr1` evaluates to `FALSE`



```
// Example usage for: Ternary Operator
$action = empty($formAction) ? 'default' : $formAction;

// The above is identical to this if/else statement
if (empty($formAction)) {
    $action = 'default';
} else {
    $action = $formAction;
}
```

Simplified ternary operator

Since PHP 5.3, it is possible to leave out the middle part of the ternary operator. Expression expr1 ?: expr3 returns expr1 if expr1 evaluates to TRUE, and expr3 otherwise

```
$action = $formAction ?: 'default';
```

Null Coalescing Operator

Since PHP 7, it exists the "??" (Or null coalescing) operator

```
// Example usage for: Null Coalesce Operator
$action = $formAction ?? 'default';

// The above is identical to this if/else statement
if (isset($formAction)) {
    $action = $formAction;
} else {
    $action = 'default';
}
```

Useful resources

- Error reporting in PHP
- php.net whole documentation
- Strings in PHP
- PHP basics

Thanks for your attention

Q & A