PHP Syntax

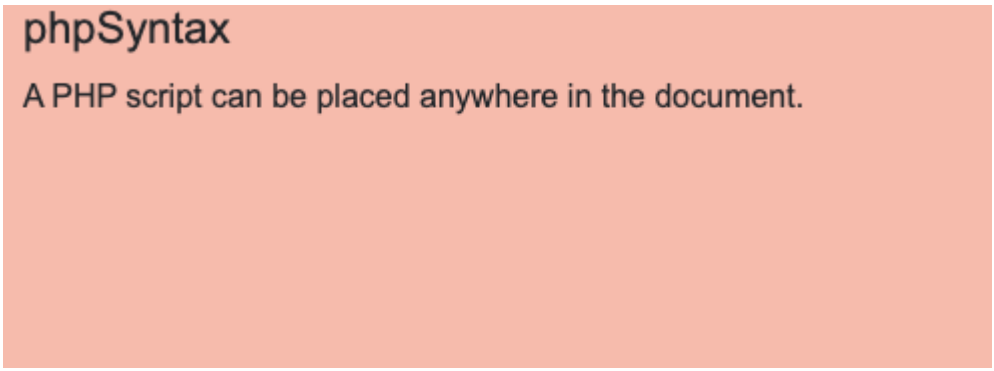
Table of Contents

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with :

```
<h4>phpSyntax</h4>
<p>A PHP script can be placed anywhere in the document. </p>
<?php
    // PHP code goes here
?>
```



Example 1

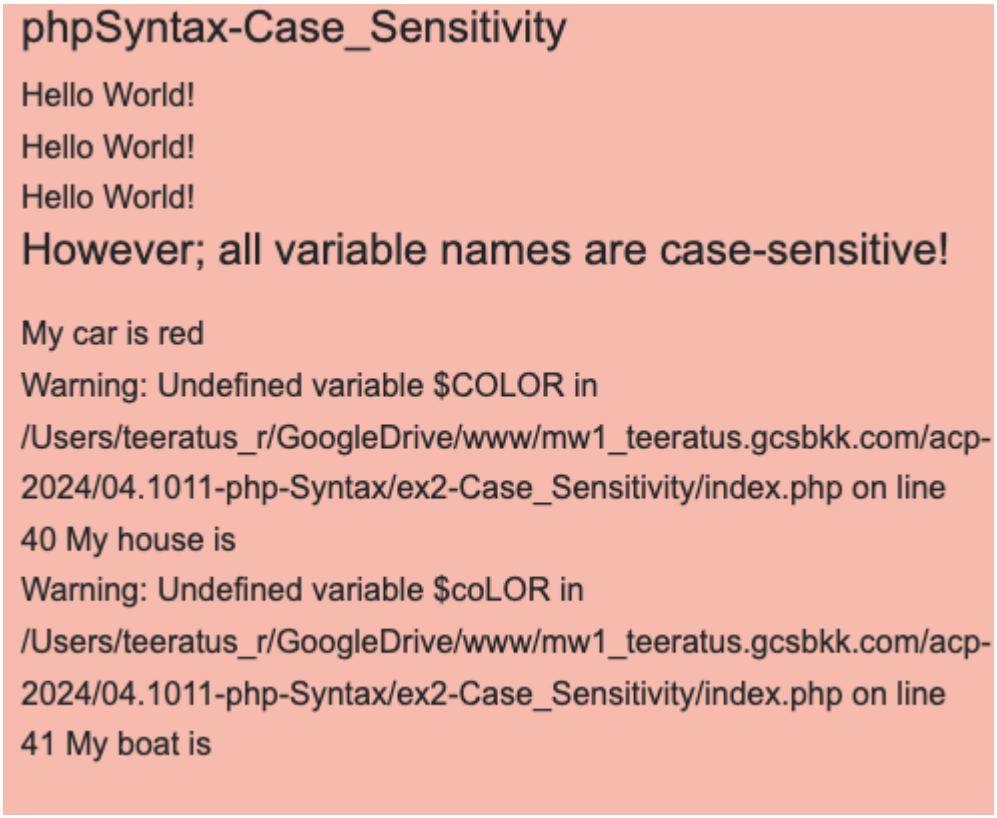
Result View Example

PHP Syntax Case Sensitivitys

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Odio porro similique eveniet tenetur voluptas at? Perferendis sunt temporibus fuga in?

```
<h4>phpSyntax-Case_Sensitivity</h4>

<?php
```

```php
    // ECHO is the same as echo:
    ECHO "Hello World!<br>";
    echo "Hello World!<br>";
    EcHo "Hello World!<br>";
?>
<h4>However; all variable names are case-sensitive!</h4>
<p></p>
<?php
    // $COLOR is not same as $color:
    $color = "red";
    echo "My car is " . $color . "<br>";
    echo "My house is " . $COLOR . "<br>";
    echo "My boat is " . $coLOR . "<br>";
?>
```



Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Comments

Table of Contents

PHP Single Line Comments

```
<h4>phpSingle_Line_Comments</h4>
<p>Single line comments start with //. Any text between // and the end of
the line will be ignored (will not be executed).
You can also use # for single line comments, but in this tutorial we will
use //.
The following examples uses a single-line comment as an explanation: </p>
<?php
    // Outputs a welcome message:
    echo "Welcome Home!";
    echo "<br>";

    echo "Welcome Home!"; // Outputs a welcome message
    echo "<br>";

    // echo "Welcome Home!";
    // echo "<br>";
?>
```

phpSingle_Line_Comments

Single line comments start with //. Any text between // and the end of the line will be ignored (will not be executed). You can also use # for single line comments, but in this tutorial we will use //. The following examples uses a single-line comment as an explanation:

Welcome Home!
Welcome Home!

Example 1

Result View Example

PHP Multiline Comments

```
<h4>php-Multiline_Comments</h4>
<p>Multi-line comments start with /* and end with */.</p>
<p>Multi-line comment as an explanation:</p>
<?php
    /*
    The next statement will
    print a welcome message
    */
    echo "Welcome Home!";
?>
<hr>
<h4>Multi-line comment to ignore code:</h4>
<p>We can use multi-line comments to prevent blocks of code from being
executed:</p>
<?php
    /*
    echo "Welcome to my home!";
    echo "Mi casa su casa!";
    */
    echo "Hello!";
?>
<hr>
<h4>Comments in the Middle of the Code</h4>
<p>The multi-line comment syntax can also be used to prevent execution of
parts inside a code-line:</p>
<?php
    $x = 5 /* + 15 */ + 5;
```

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

Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Variables

Table of Contents

PHP Creating Variables

```
<h2>Creating_Variables</h2>
<p>In PHP, a variable starts with the $ sign, followed by the name of the
variable: </p>
<?php
    $x = 5;
    $y = "John";

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

Example 1

Result View Example

PHP Output Variables

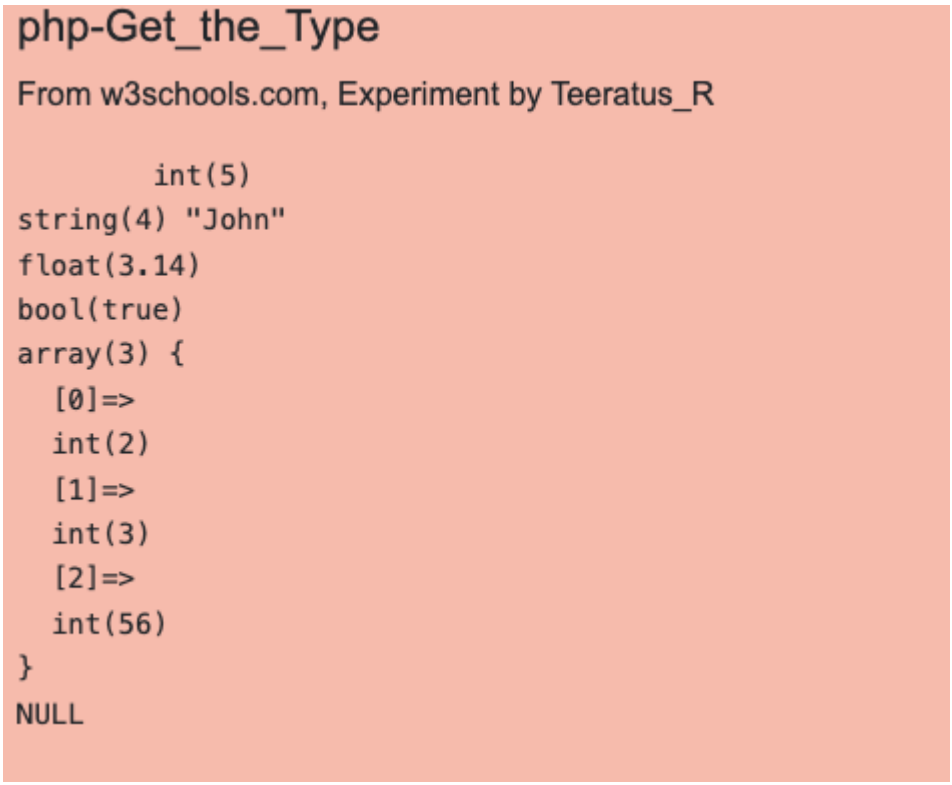
```
<h4>php-Output_Variables</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
    $txt = "W3Schools.com";
    echo "I love $txt!";
    echo "<br>";
    echo "<hr>";
    echo "I love " . $txt . "!";
    echo "<hr>";
    $x = 5;
    $y = 4;
    echo $x + $y;

?>
```

Example 2

Result View Example

PHP Get the Type

```
<h4>php-Get_the_Type</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<pre>
<?php
    var_dump(5);
    var_dump("John");
    var_dump(3.14);
    var_dump(true);
    var_dump([2, 3, 56]);
    var_dump(NULL);
?>

</pre>
```

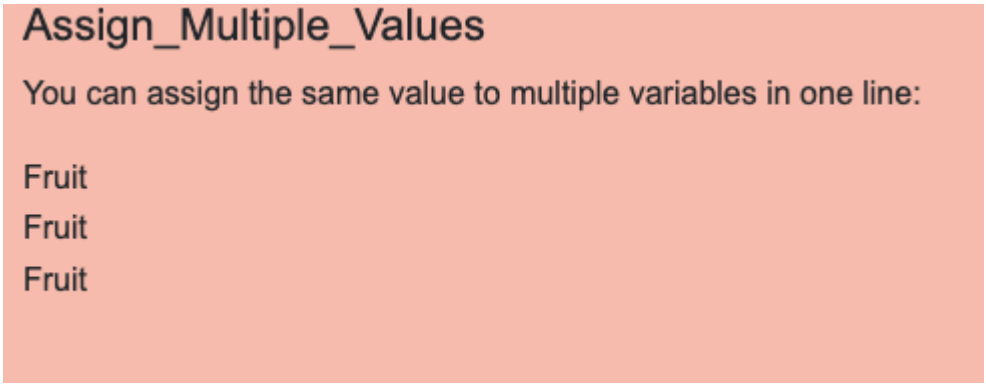


Example 3

Result View Example

PHP Assign Multiple Values

```
<h4>Assign_Multiple_Values</h4>
<p>You can assign the same value to multiple variables in one line:</p>
<?php
    $x = $y = $z = "Fruit";

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



Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Variables Scope

Back to ACP page

Table of Contents

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

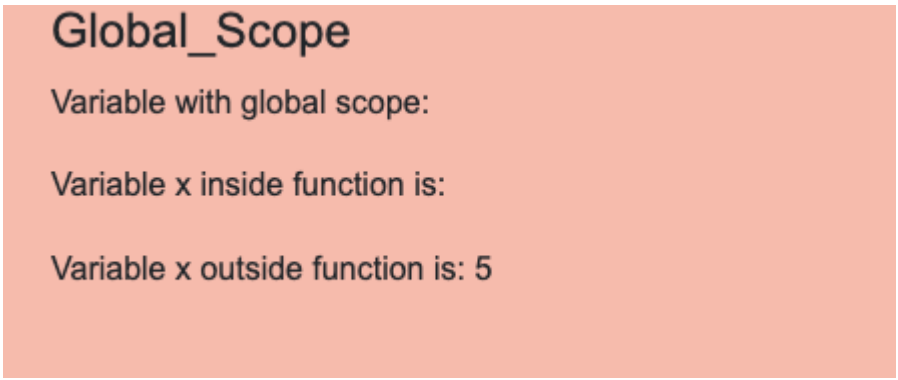
The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

PHP 1 Global Scope

```
<h4>Global_Scope</h4>
<p>Variable with global scope: </p>
<?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>";

?>
```



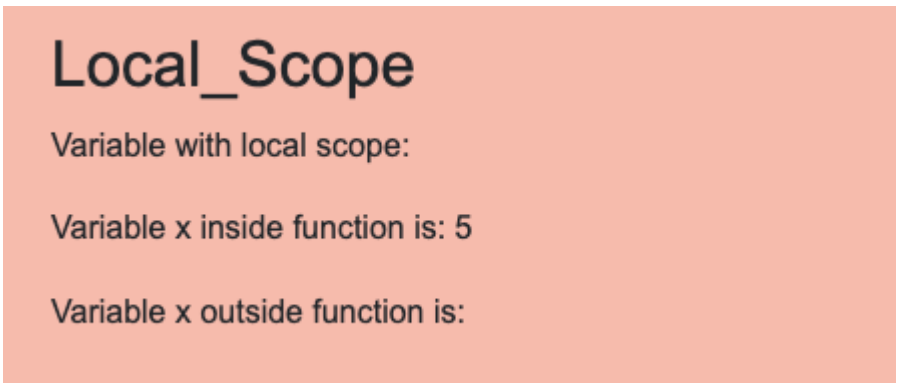
Example 1

Result View Example

PHP 2 Local Scope

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

```
<h2>Local_Scope</h2>
<p>Variable with local scope: </p>
<?php
function myTest() {
$x = 5; // local scope
echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```



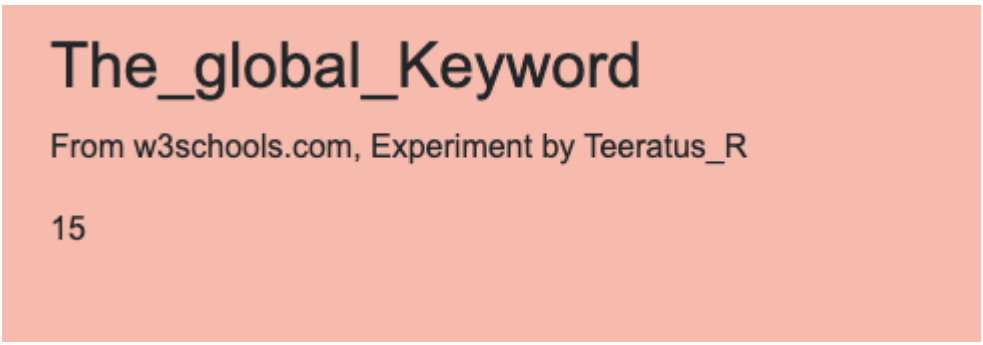
Example 2

Result View Example

PHP 3

Lorem ipsum dolor sit, amet consectetur adipisicing elit.

```
<h2>The_global_Keyword</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?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
?>
```



Example 3

Result View Example

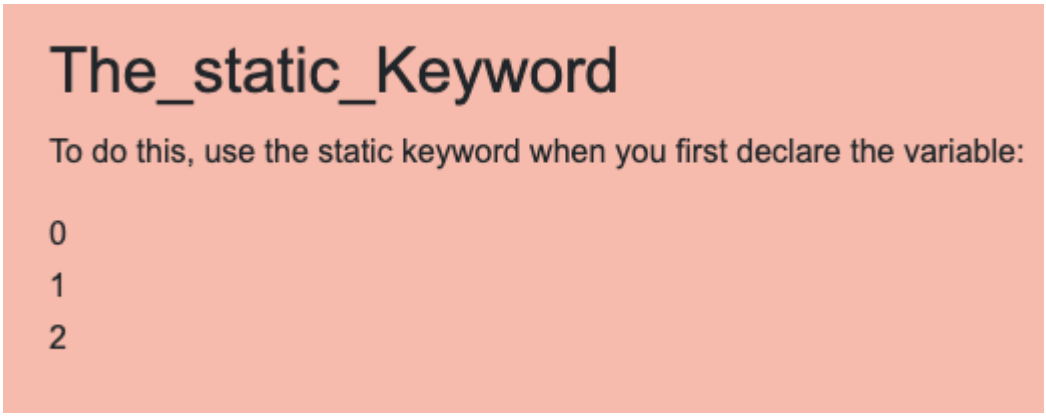
PHP 4 The static Keyword

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:

```
<h2>The_static_Keyword</h2>
<p>To do this, use the static keyword when you first declare the variable:
</p>
<?php
function myTest() {
static $x = 0;
echo $x;
$x++;
}

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

```
myTest();
?>
```



Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

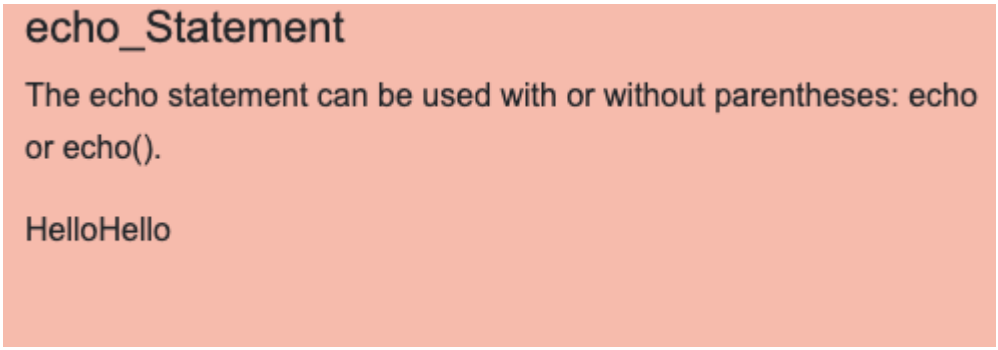
PHP echo and print

Table of Contents

PHP echo Statement

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

```
<h4>echo_Statement</h4>
<p>The echo statement can be used with or without parentheses: echo or
echo().</p>
<?php
echo "Hello";
//same as:
echo ("Hello");
?>
```



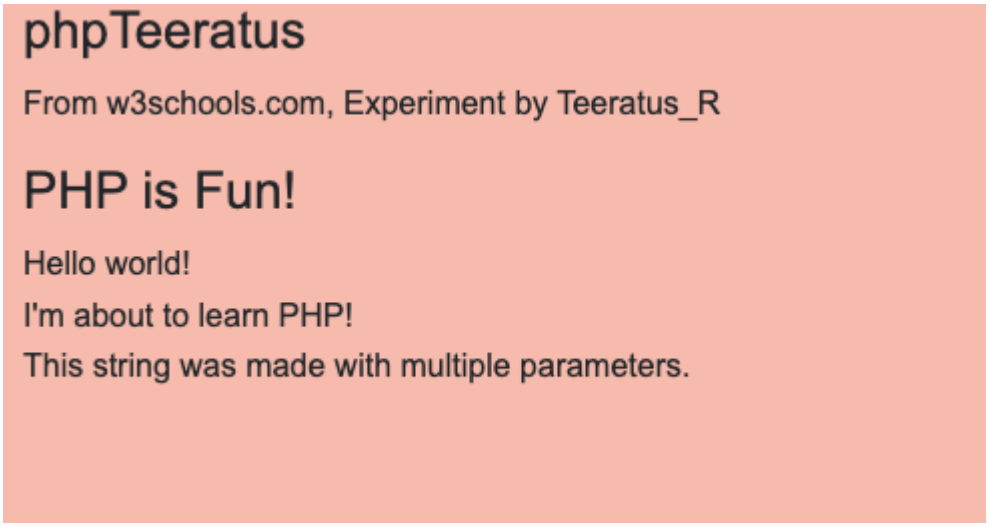
Example 1

Result View Example

PHP Display Text

In the example below, we use the PHP echo statement to output the text "Hello World!".

```
<h2>phpTeeratus</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?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.";
?>
```



Example 2

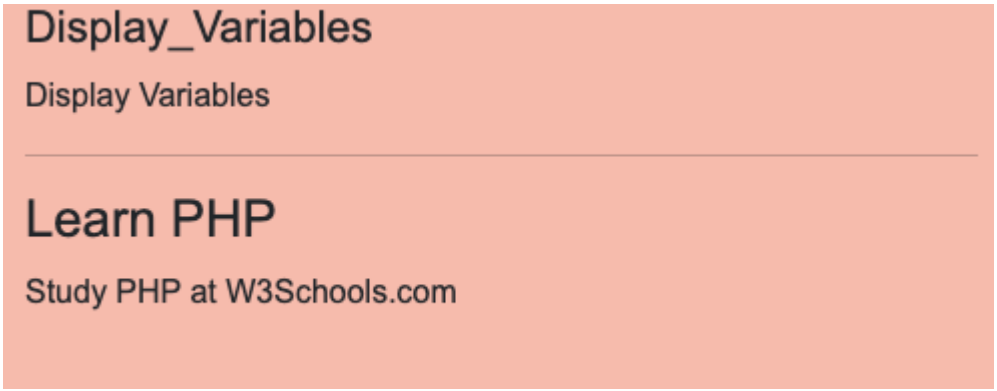
Result View Example

PHP Display Variables

In the example below, we use the PHP echo statement to output the text "Hello World!", but we use a variable to output the text:

```
<h4>Display_Variables</h4>
<p>Display Variables</p>
<hr>
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";

    echo "<h2>$txt1</h2>";
```

```
    echo "<p>Study PHP at $txt2</p>";
?>
```

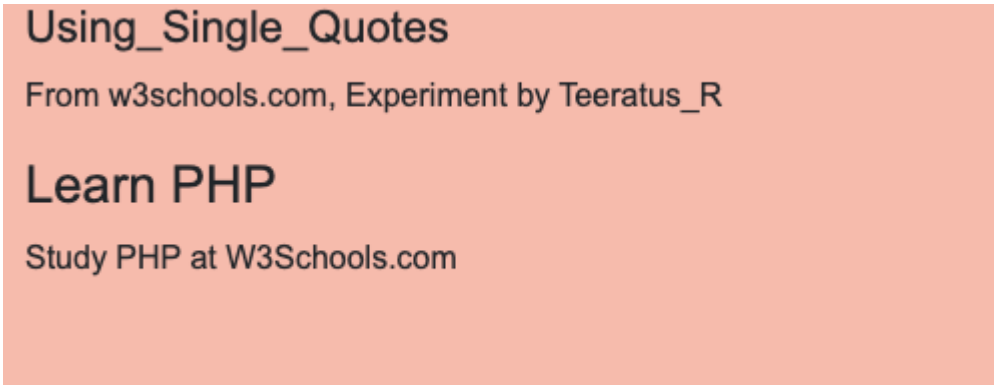


Example 3

Result View Example

PHP Using Single Quotes

```
<h4>Using_Single_Quotes</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";

    echo '<h2>' . $txt1 . '</h2>';
    echo '<p>Study PHP at ' . $txt2 . '</p>';
?>
```

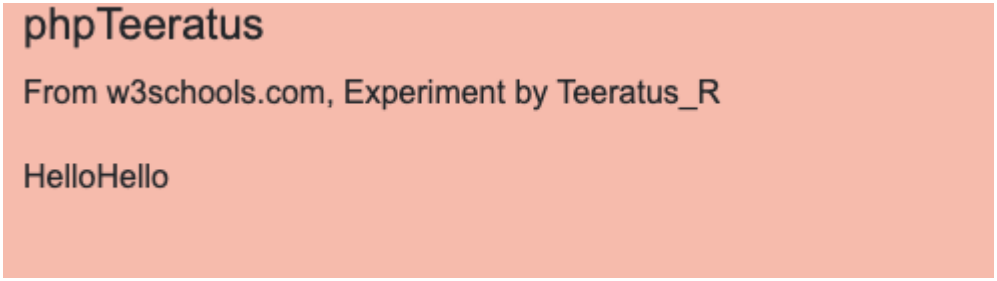


Example 4

Result View Example

PHP print Statement

```
<h4>phpTeeratus</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
    // The PHP print Statement
    print "Hello";
    //same as:
    print("Hello");

?>
```



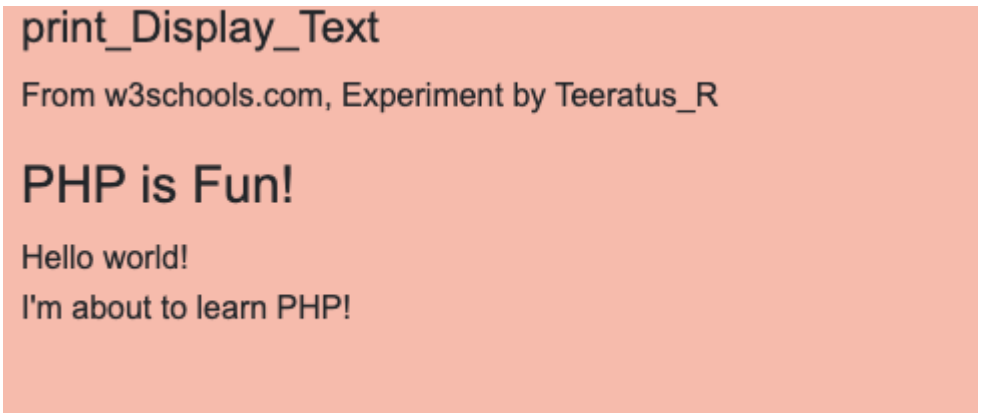
Example 5

Result View Example

PHP print Display Text

The following example shows how to output text with the print command

```
<h4>print_Display_Text</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
    print "<h2>PHP is Fun!</h2>";
    print "Hello world!<br>";
    print "I'm about to learn PHP!";
?>
```

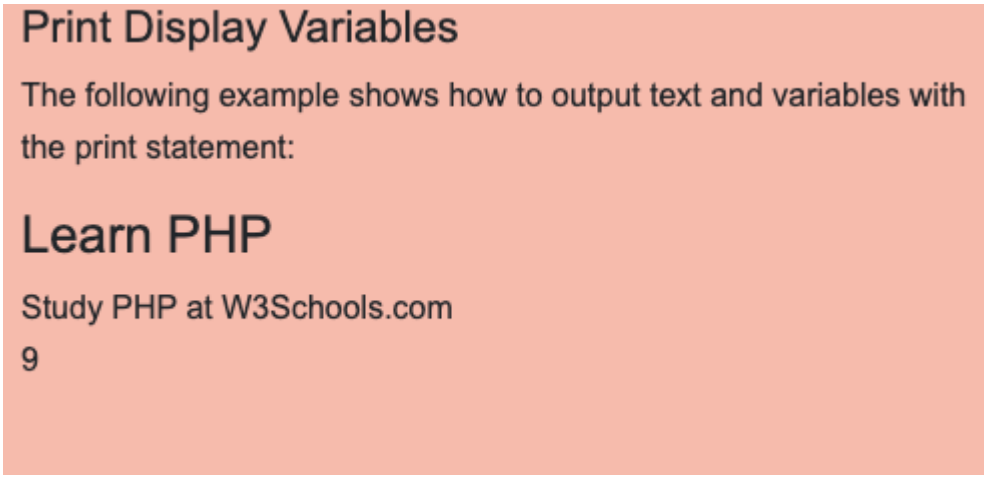


Example 6

Result View Example

PHP 7

```
<h4>Print Display Variables</h4>
<p>The following example shows how to output text and variables with the
print statement:</p>
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";
    $x = 5;
    $y = 4;

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



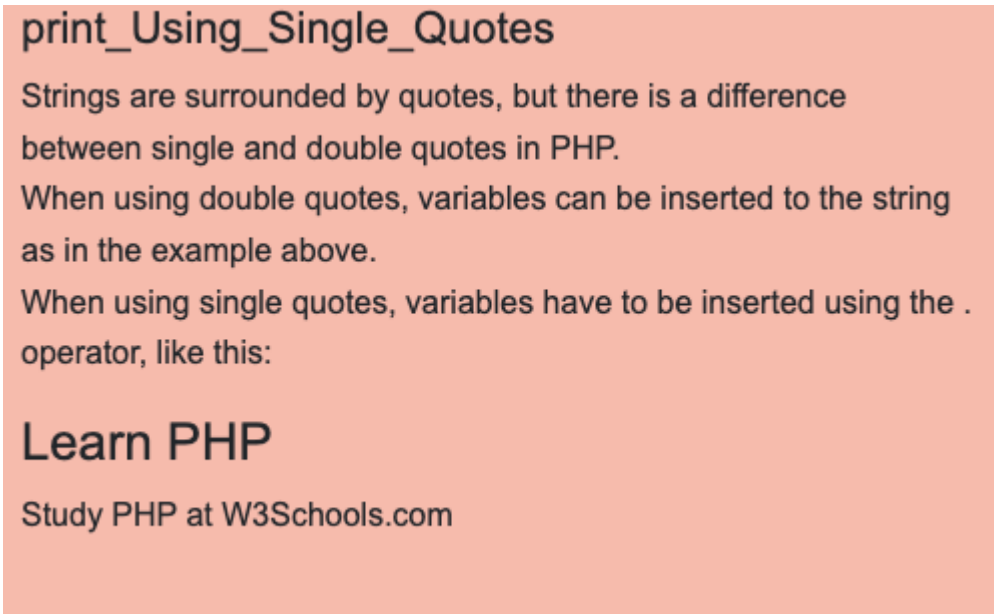
Example 7

Result View Example

PHP print Using Single Quotes

Strings are surrounded by quotes, but there is a difference between single and double quotes in PHP.
When using double quotes, variables can be inserted to the string as in the example above.
When using single quotes, variables have to be inserted using the . operator, like this:

```
<h4>print_Using_Single_Quotes</h4>

<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";

    print '<h2>' . $txt1 . '</h2>';
    print '<p>Study PHP at ' . $txt2 . '</p>';
?>
```



Example 8

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Data Types

Back to ACP page

Table of Contents

PHP Getting the Data Type - String -Integer - Float - Boolean - Array

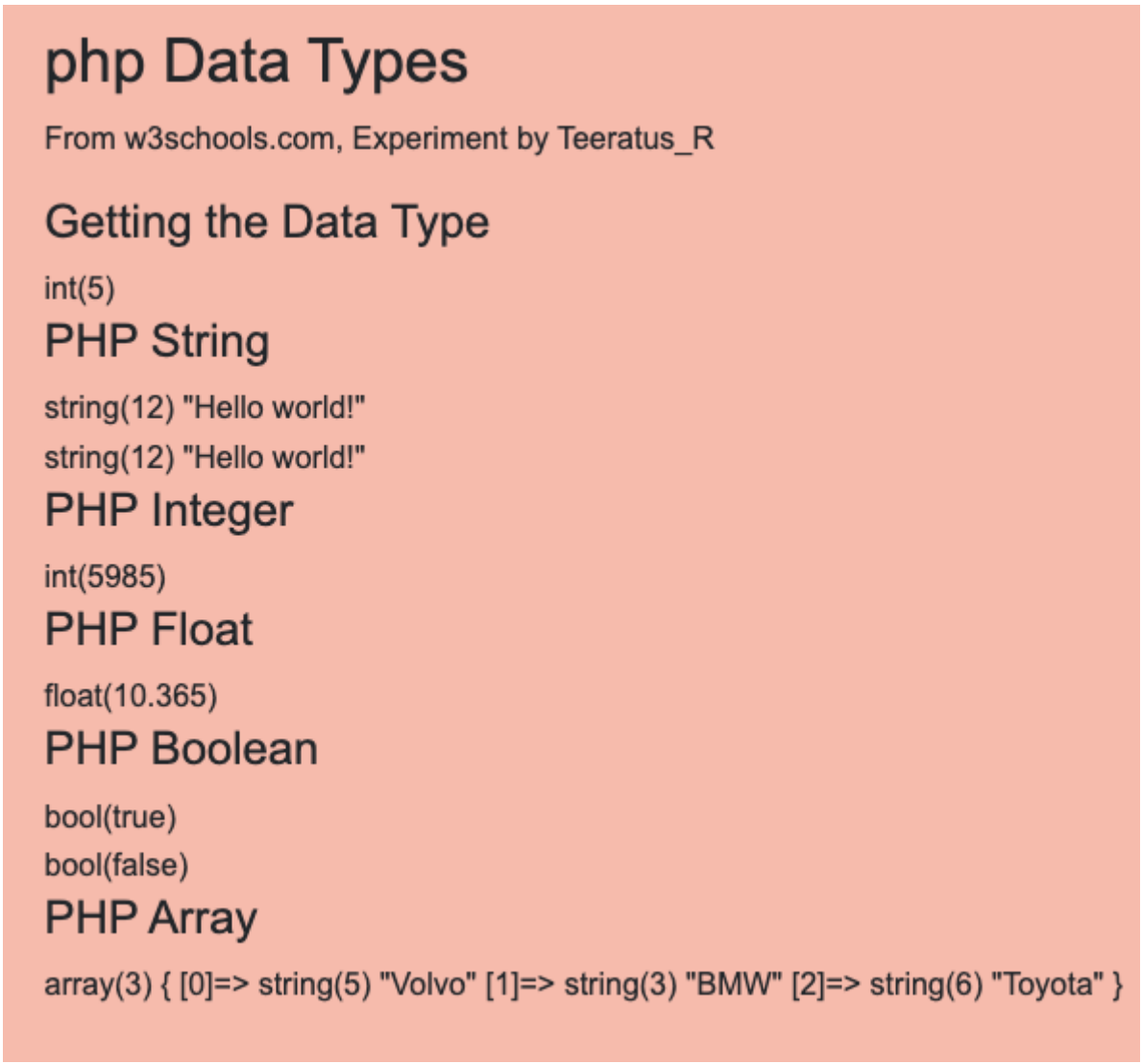
Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

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

```
<h2>php Data Types</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>

<h4>Getting the Data Type</h4>
<?php
// The PHP var_dump() function returns the data type and value:
$x = 5;
var_dump($x);
?>
<h4>PHP String</h4>
<?php
$x = "Hello world!";
$y = 'Hello world!';
var_dump($x);
```

```
echo "<br>";
var_dump($y);
?>
<h4>PHP Integer</h4>
<?php
$x = 5985;
var_dump($x);
?>
<h4>PHP Float</h4>
<?php
$x = 10.365;
var_dump($x);
?>
<h4>PHP Boolean</h4>
<?php
$x = true;
$y = false;
var_dump($x);
echo "<br>";
var_dump($y);
?>
<h4>PHP Array</h4>
<?php
$cars = array("Volvo", "BMW", "Toyota");
var_dump($cars);
?>
```



Example 1

Result View Example

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.

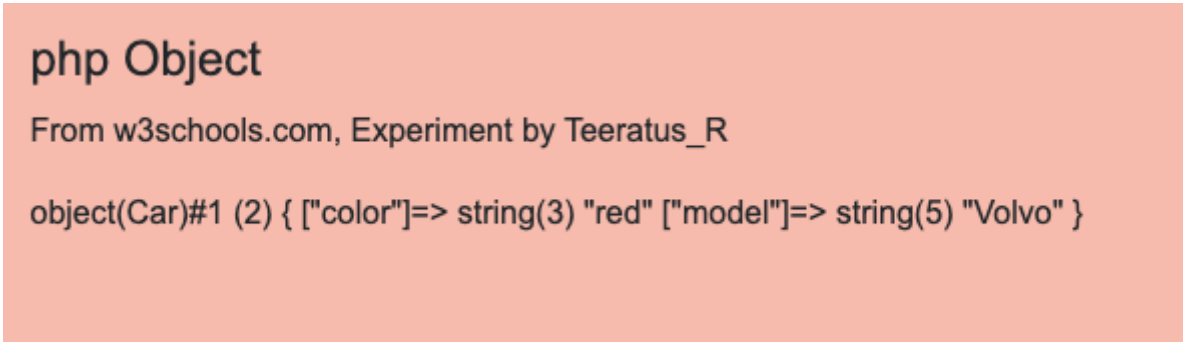
When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

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.

```
<h4>php Object</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>

<?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);

?>
```



Example 2

Result View Example

PHP NULL Value

Null is a special data type which can have only one value: NULL.

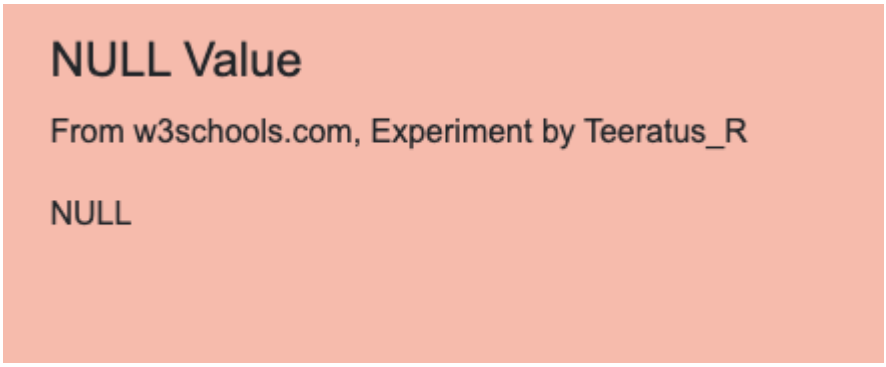
A variable of data type NULL is a variable that has no value assigned to it.

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:

```
<h4>NULL Value</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>

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

```
var_dump($x);
?>
```

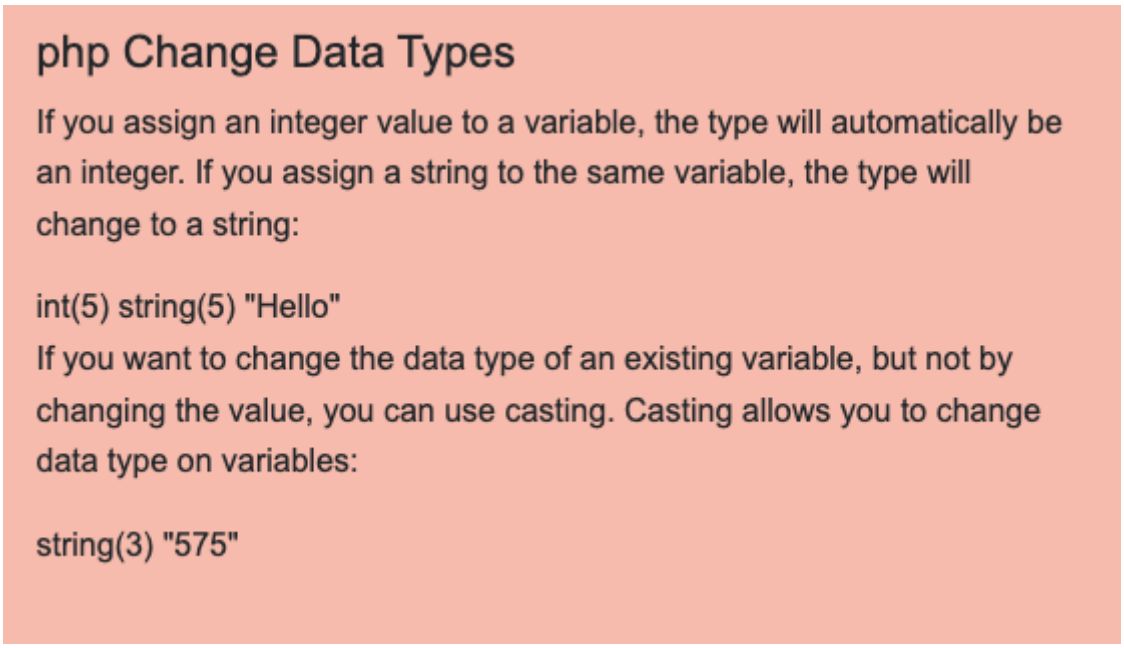


Example 3

Result View Example

PHP Change Data Type

```
<h4>php Change Data Types</h4>
        <p>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:</p>

        <?php
        $x = 5;
        var_dump($x);
        $x = "Hello";
        var_dump($x);
        ?>
        <p>If you want to change the data type of an existing variable,
but not by changing the value, you can use casting.

Casting allows you to change data type on variables:</p>
        <?php
        $x = 575;
        $x = (string) $x;
        var_dump($x);
        ?>
```



Example 4

Result View Example

PHP 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.

We will not talk about the resource type here, since it is an advanced topic.

Document

Document in project

You can [Download PDF](php-Data _Types.pdf) file.

Reference

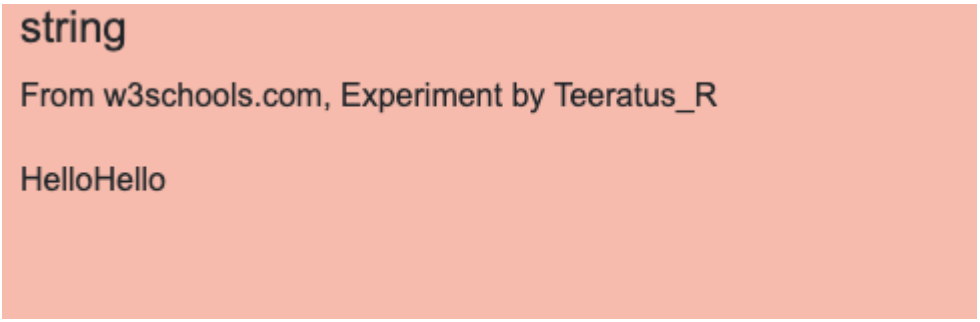
PHP String

Back to ACP page

Table of Contents

PHP String

```
<h4>string</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
echo "Hello";
echo 'Hello';
?>
```

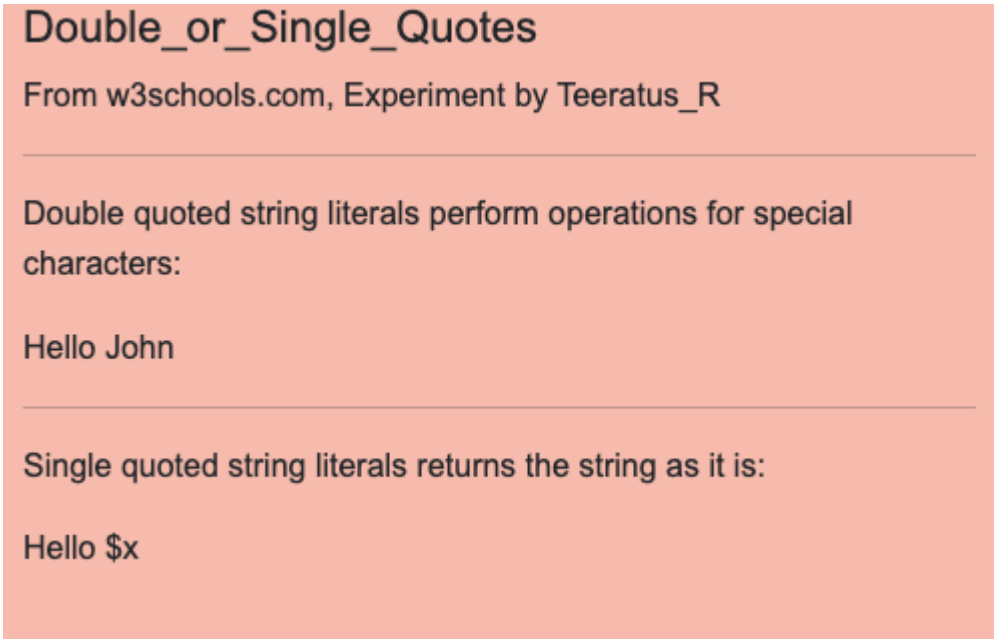


Example 1

Result View Example

PHP Double or Single Quotes

```
<h4>Double_or_Single_Quotes</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<hr>
<p>Double quoted string literals perform operations for special
characters:</p>
<?php
$x = "John";
echo "Hello $x";
?>
<hr>
<p>Single quoted string literals returns the string as it is:</p>
<?php
$x = "John";
echo 'Hello $x';
?>
```

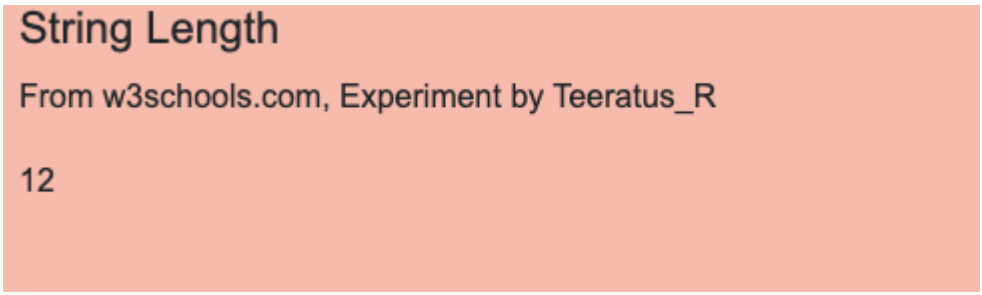


Example 2

Result [View Example](ex2-Double or Single Quotes/index.php)

PHP String Length

```
<h4>String Length</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
echo strlen("Hello world!");
?>
```

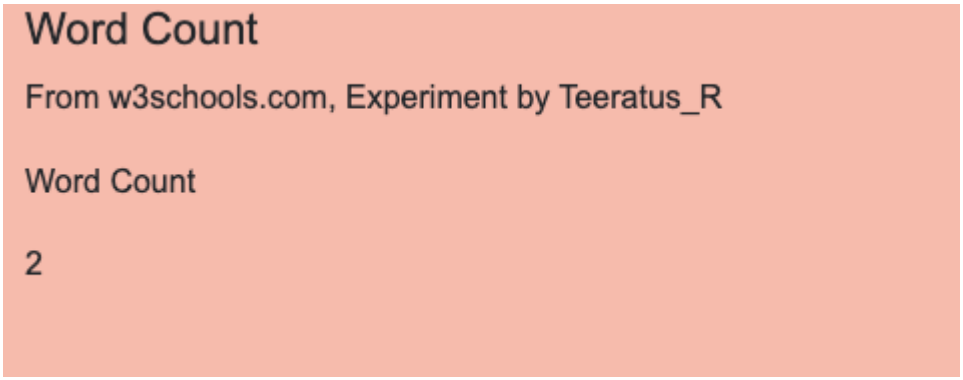


Example 3

Result View Example

PHP Word Count

```
<h4>Word Count</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<p>Word Count</p>
<?php
    echo str_word_count("Hello world!");
?>
```

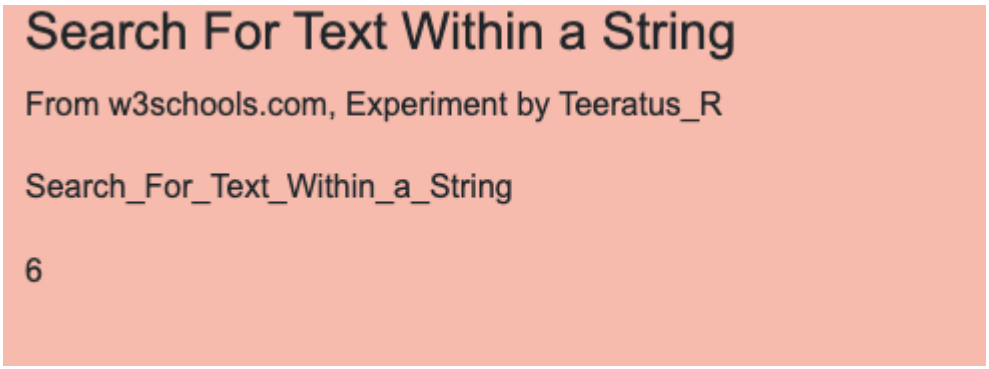


Example 4

Result View Example

PHP Search For Text Within a String

```
<h2>Search For Text Within a String</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<p>Search_For_Text_Within_a_String</p>
<?php
echo strpos("Hello world!", "world");
?>
```



Example 5

Result [View Example](ex5-Search For Text Within a String/index.php)

Document

Document in project

You can Download PDF file.

Reference

PHP Modify Strings

Back to ACP page

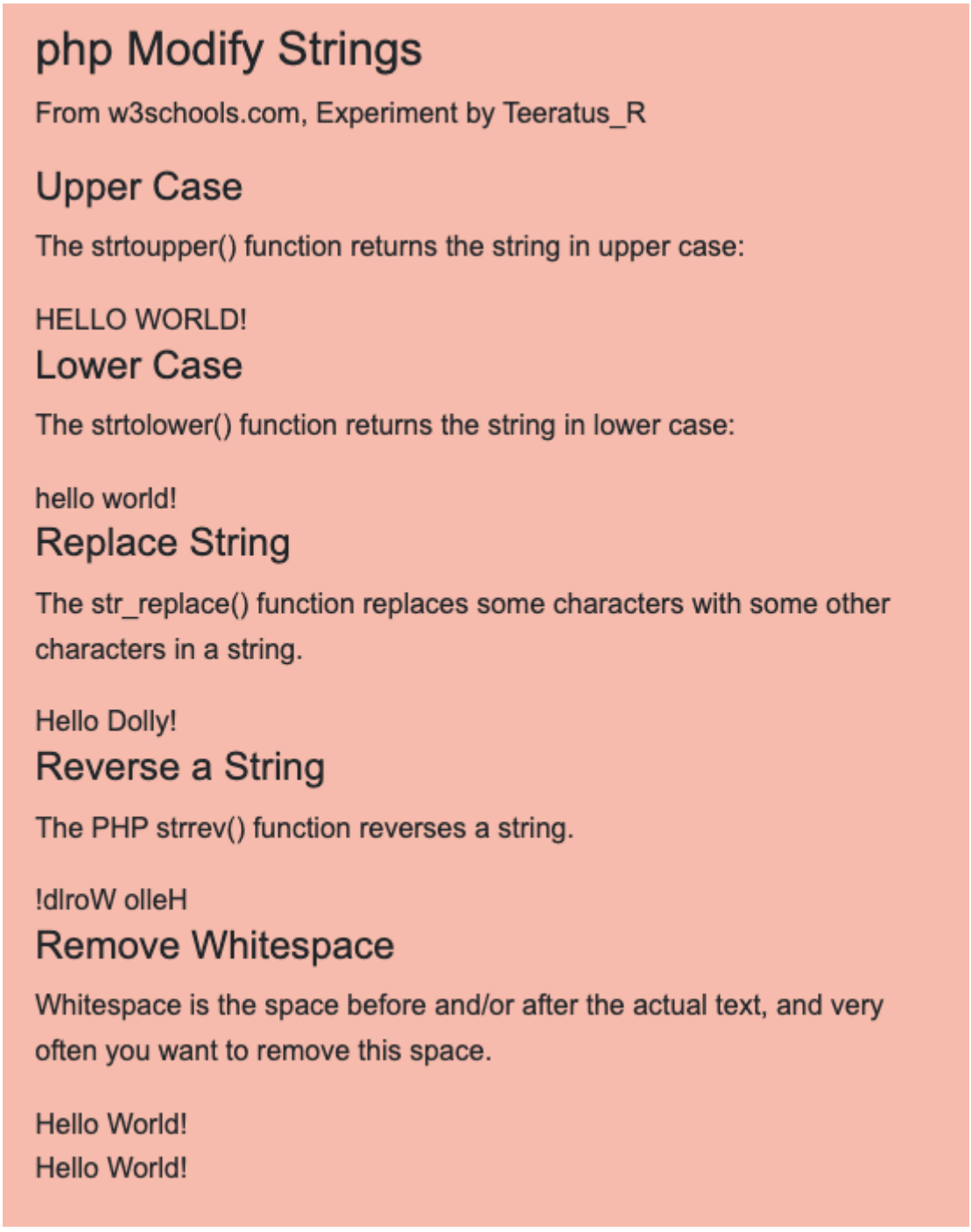
Table of Contents

PHP Modify Strings

PHP has a set of built-in functions that you can use to modify strings.

```
<h2>php Modify Strings</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<h4>Upper Case</h4>
<p>The strtoupper() function returns the string in upper case:</p>
<?php
$x = "Hello World!";
echo strtoupper($x);
?>
<h4>Lower Case</h4>
<p>The strtolower() function returns the string in lower case:</p>
<?php
$x = "Hello World!";
echo strtolower($x);
?>
<h4>Replace String</h4>
<p>The str_replace() function replaces some characters with some other
characters in a string.</p>
<?php
$x = "Hello World!";
echo str_replace("World", "Dolly", $x);
?>
<h4>Reverse a String</h4>
<p>The PHP strrev() function reverses a string.</p>
<?php
$x = "Hello World!";
echo strrev($x);
?>
<h4>Remove Whitespace</h4>
<p>Whitespace is the space before and/or after the actual text, and very
often you want to remove this space.</p>
<?php
```

```
$x = "  Hello   World!  ";
echo $x . "<br>";
echo trim($x);
?>
```



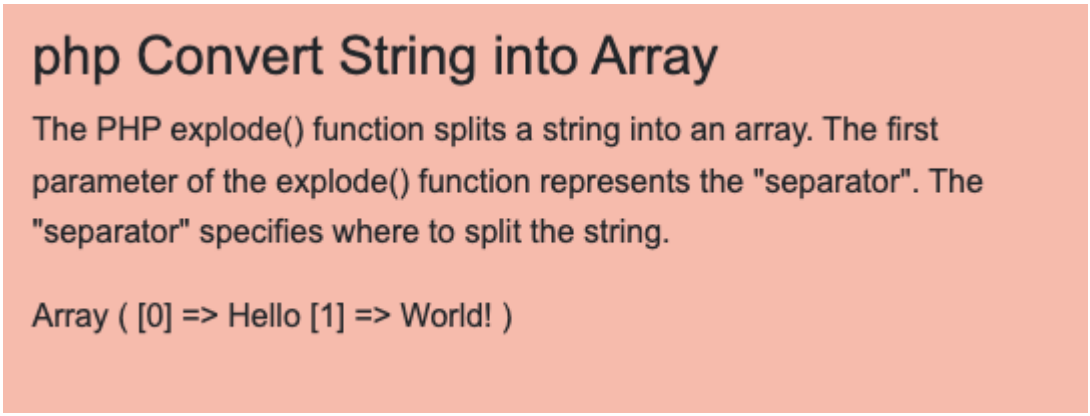
Example 1

Result View Example

PHP Convert String into Array

```
<h2>php Convert String into Array</h2>
<p>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.</p>
<?php
$x = "Hello World!";
$y = explode(" ", $x);

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

/*
Result:
Array ( [0] => Hello [1] => World! )
*/
?>
```



Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

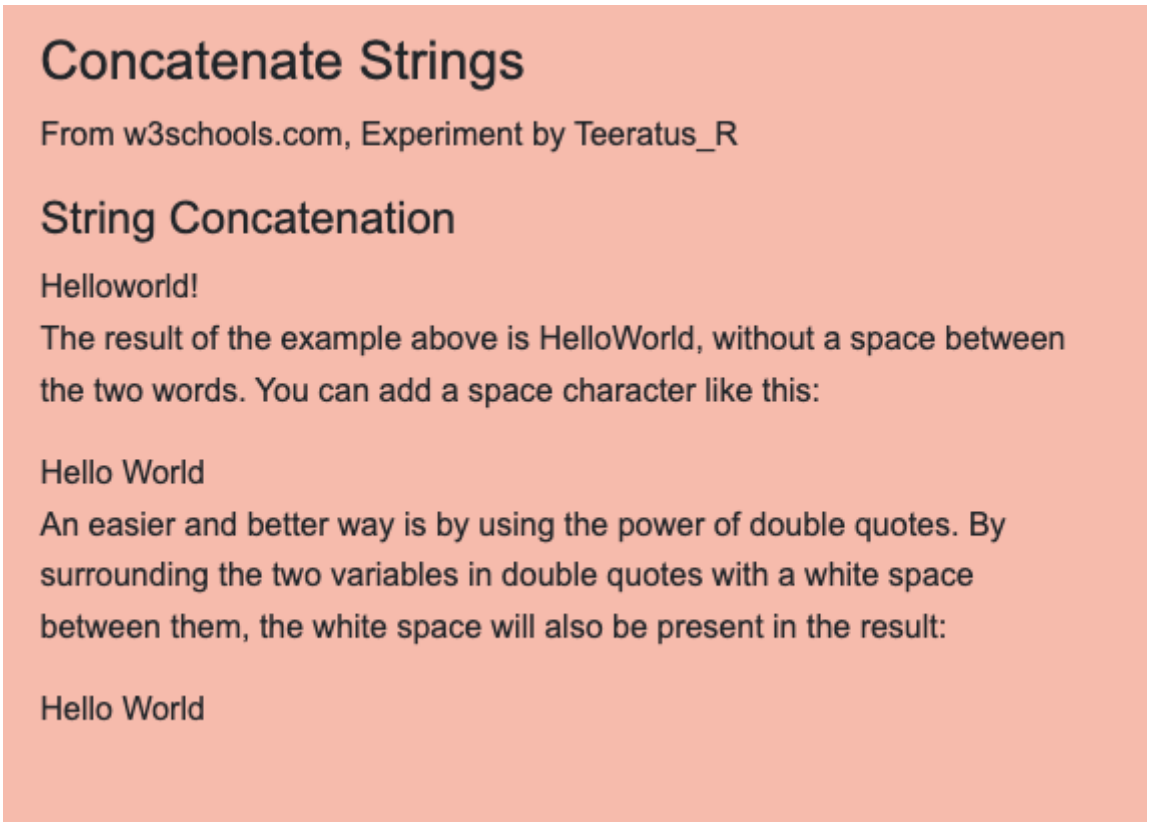
PHP Concatenate Strings

Table of Contents

PHP String Concatenation

To concatenate, or combine, two strings you can use the . operator:

```
<h2>Concatenate Strings</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>

<h4>String Concatenation</h4>
<?php
    $txt1 = "Hello";
    $txt2 = "world!";
    echo $txt1 . $txt2;
?>
<p>The result of the example above is HelloWorld, without a space between
the two words.
You can add a space character like this:</p>
<?php
$x = "Hello";
$y = "World";
$z = $x . " " . $y;
echo $z;
?>
<p>An easier and better way is by using the power of double quotes.

By surrounding the two variables in double quotes with a white space
between them, the white space will also be present in the result:</p>
<?php
$x = "Hello";
$y = "World";
$z = "$x $y";
echo $z;
?>
```



Example 1

Result View Example

Document

Document in project

You can Download PDF file.

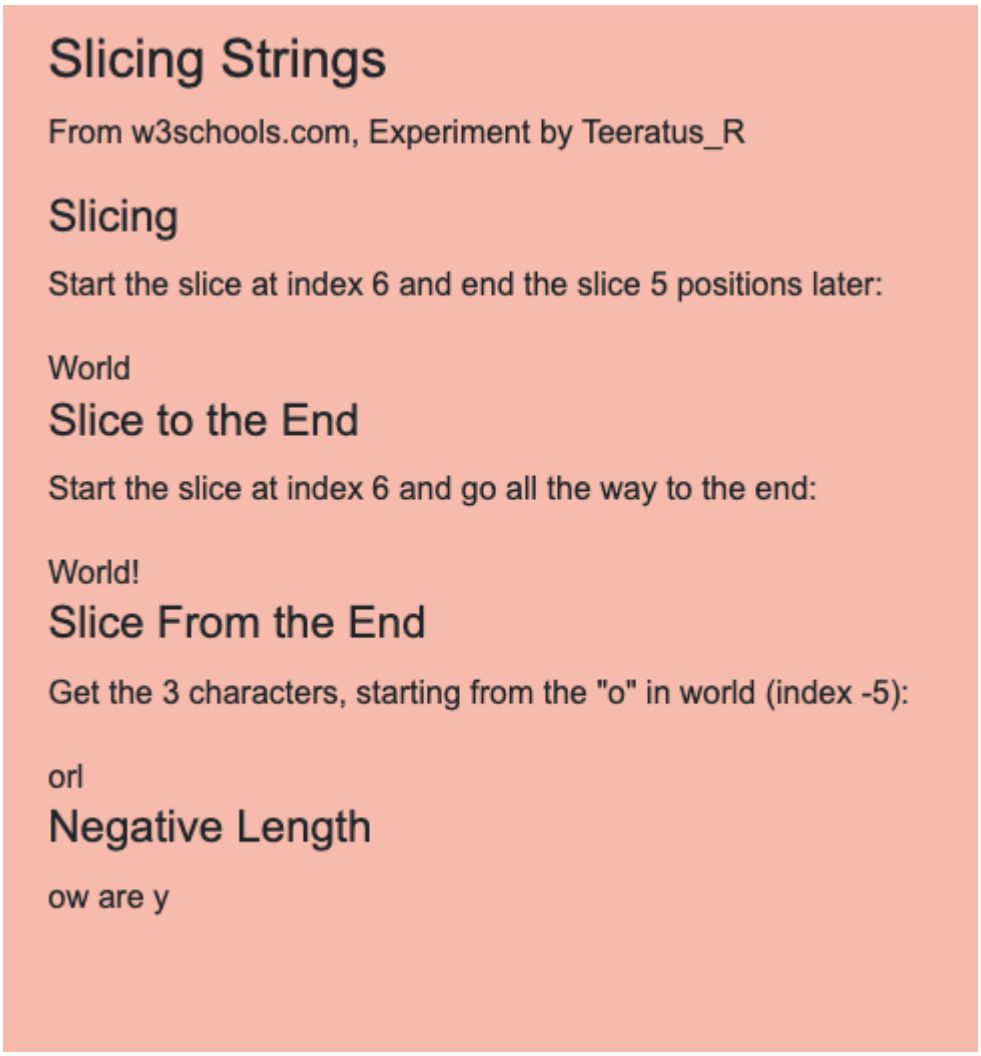
Reference

PHP Slicing Strings

Table of Contents

PHP Slicing Strings

```
<h2>Slicing Strings</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>

<h4>Slicing</h4>
<p>Start the slice at index 6 and end the slice 5 positions later:</p>
<?php
$x = "Hello World!";
echo substr($x, 6, 5);
?>
<h4>Slice to the End</h4>
<p>Start the slice at index 6 and go all the way to the end:</p>
<?php
$x = "Hello World!";
echo substr($x, 6);
?>
<h4>Slice From the End</h4>
<p>Get the 3 characters, starting from the "o" in world (index -5):</p>
<?php
$x = "Hello World!";
echo substr($x, -5, 3);
?>
<h4>Negative Length</h4>
<?php
$x = "Hi, how are you?";
echo substr($x, 5, -3);
?>
```



Example 1

Result View Example

Document

Document in project

You can Download PDF file.

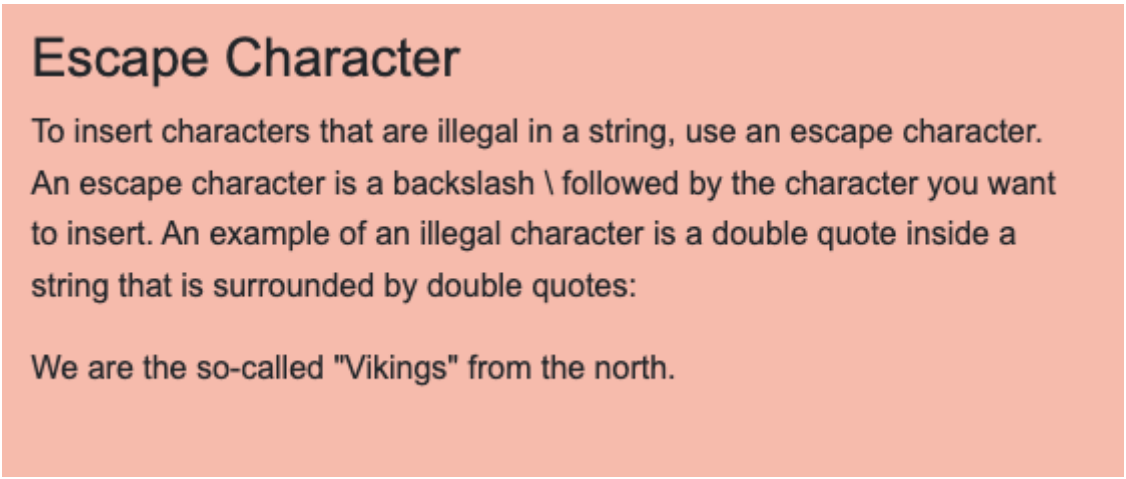
Reference

PHP Escape Characters

Table of Contents

PHP Escape Character

```
<h2>Escape Character</h2>
        <p>To insert characters that are illegal in a string, use an
escape character.
An escape character is a backslash \ followed by the character you want to
insert.
An example of an illegal character is a double quote inside a string that
is surrounded by double quotes:</p>
        <?php
            echo "We are the so-called \"Vikings\" from the north.";
        ?>
```



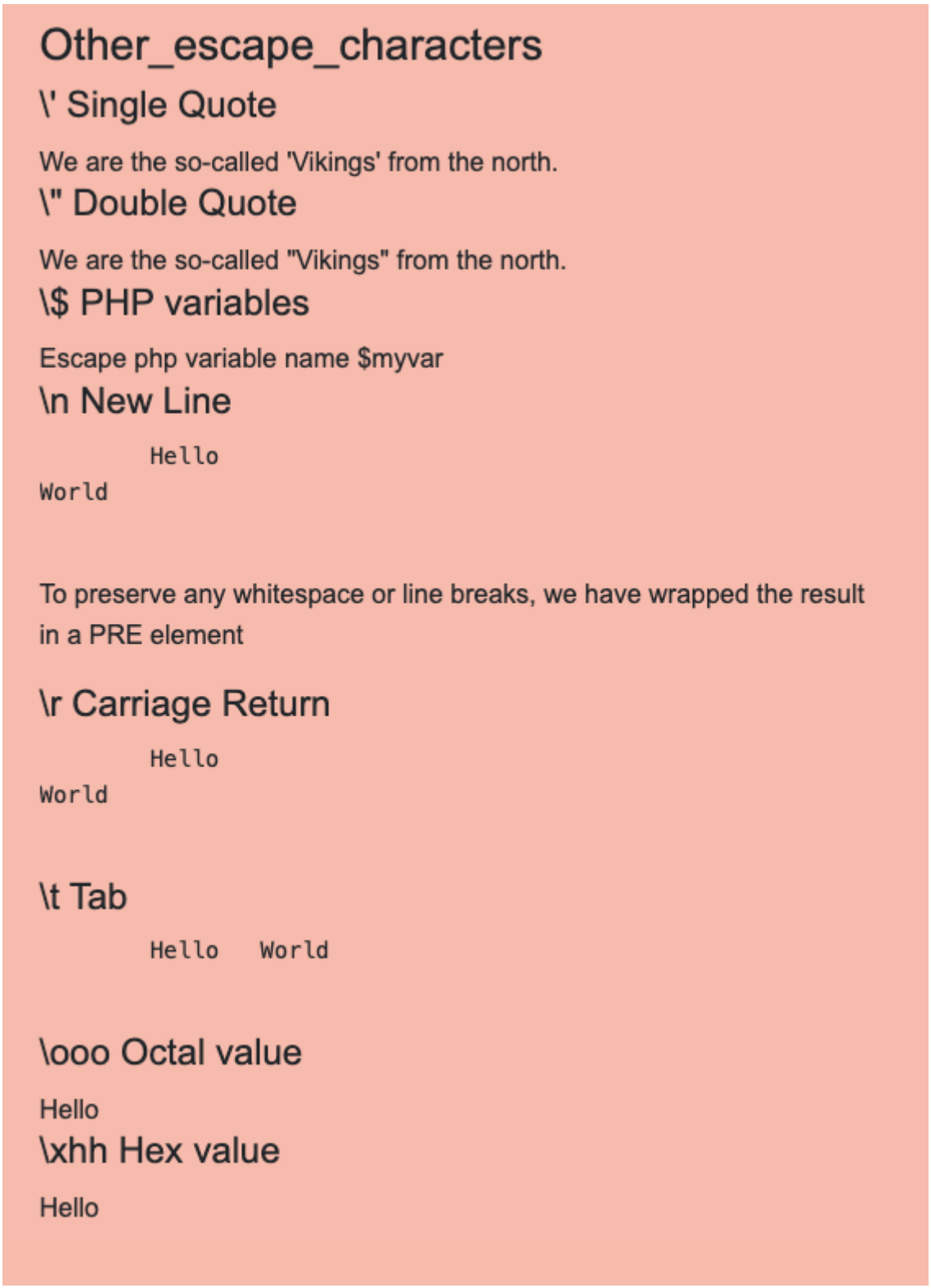
Example 1

Result View Example

PHP Other escape characters

```
<h2>Other_escape_characters</h2>
```

```
<h4>\' Single Quote</h4>
<?php
    $x = 'We are the so-called \'Vikings\' from the north.';
    echo $x;
?>
<h4>\" Double Quote</h4>
<?php
    $x = "We are the so-called \"Vikings\" from the north.";
    echo $x;
?>
<h4>\$ PHP variables</h4>
<?php
    $x = "Escape php variable name \$myvar";
    echo $x;
?>
<h4>\n New Line</h4>
<pre>
<?php
$x = "Hello\nWorld";
echo $x;
?>
</pre>

<p>To preserve any whitespace or line breaks, we have wrapped the result
in a PRE element</p>
<h4>\r Carriage Return</h4>
<pre>
<?php
$x = "Hello\rWorld";
echo $x;
?>
</pre>
<h4>\t Tab</h4>
<pre>
<?php
$x = "Hello\tWorld";
echo $x;
?>
</pre>
<h4>\ooo Octal value</h4>
<?php
    $x = "\110\145\154\154\157";
    echo $x;
?>
<h4>\xhh Hex value</h4>
<?php
    $x = "\x48\x65\x6c\x6c\x6f";
    echo $x;
?>
```



Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Numbers

Back to ACP page

Table of Contents

PHP Number

```
<h4>php-number</h2>
<p>There are three main numeric types in PHP: Integer, Float, Number
Strings</p>
<p>In addition, PHP has two more data types used for numbers:Infinity,
NaN</p>
<?php
    $a = 5;
    $b = 5.34;
    $c = "25";

    var_dump($a);       // Result: int(5)
    echo "<br>";
    var_dump($b);       // Result: float(5.34)
    echo "<br>";
    var_dump($c);       // Result: string(2) "25"
?>
```

php-number

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

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

Example 1

Result View Example

PHP 2

```
<h4>php-number-Integers</h2>
<p>An integer data type is a non-decimal number between -2147483648 and
2147483647 in 32 bit systems, and between -9223372036854775808 and
9223372036854775807 in 64 bit systems. A value greater (or lower) than
this, will be stored as float, because it exceeds the limit of an integer.
</p>
<?php
  // Check if the type of a variable is integer
  $x = 5985;
  var_dump(is_int($x));   // Result: bool(true)

  echo "<br>";

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

php-number-Integers

An integer data type is a non-decimal number between -2147483648 and 2147483647 in 32 bit systems, and between -9223372036854775808 and 9223372036854775807 in 64 bit systems. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.

bool(true)
bool(false)

Example 2

Result View Example

PHP Float

```
<h4>php-number-float</h2>
<p>A float is a number with a decimal point or a number</p>

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

php-number-float

A float is a number with a decimal point or a number

bool(true)

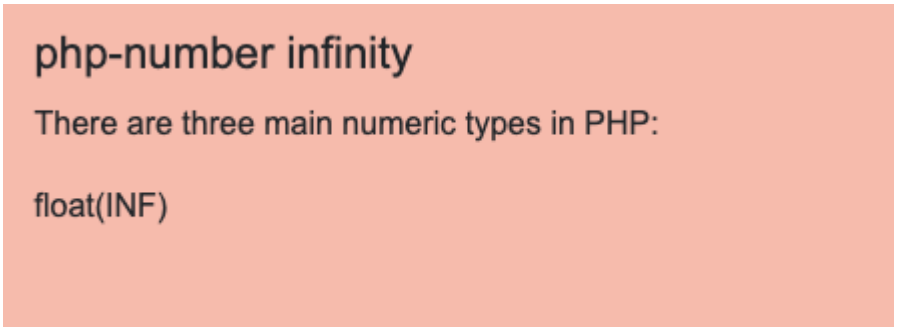
Example 3

Result View Example

PHP Infinity

```
<h4>php-number infinity</h4>
<p>There are three main numeric types in PHP:</p>

<?php
```

```
// Check if a numeric value is finite or infinite
$x = 1.9e411;
var_dump($x);   // Result: float(INF)
?>
```



Example 4

Result View Example

PHP NaN

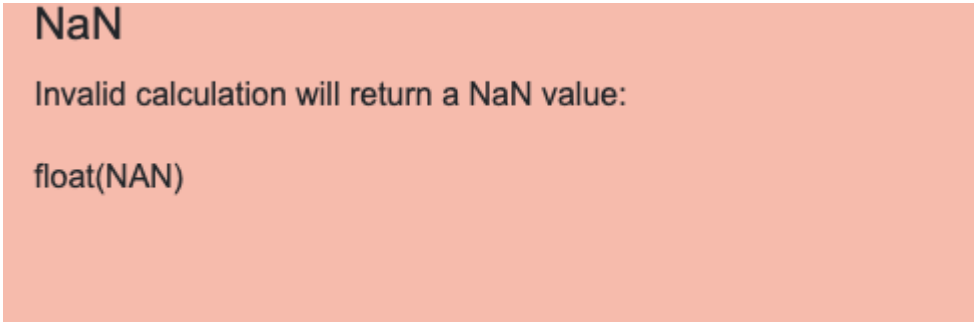
NaN stands for Not a Number.

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:

```
<h4>NaN</h4>
<p>Invalid calculation will return a NaN value:</p>

<?php
// Invalid calculation will return a NaN value
$x = acos(1.01);
var_dump($x);       // Result: float(NAN)
```



Example 5

Result View Example

PHP 1

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Odio porro similique eveniet tenetur voluptas at? Perferendis sunt temporibus fuga in?

Example 6

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Casting

Table of Contents

PHP Cast to String

```
<pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = "hello"; // String
$d = true;    // Boolean
$e = NULL;    // NULL

$a = (string) $a;
$b = (string) $b;
$c = (string) $c;
$d = (string) $d;
$e = (string) $e;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
```

```
var_dump($e);
?>
</pre>
```

Cast to String

To cast to string, use the (string) statement:

```
string(1) "5"
string(4) "5.34"
string(5) "hello"
string(1) "1"
string(0) ""
```

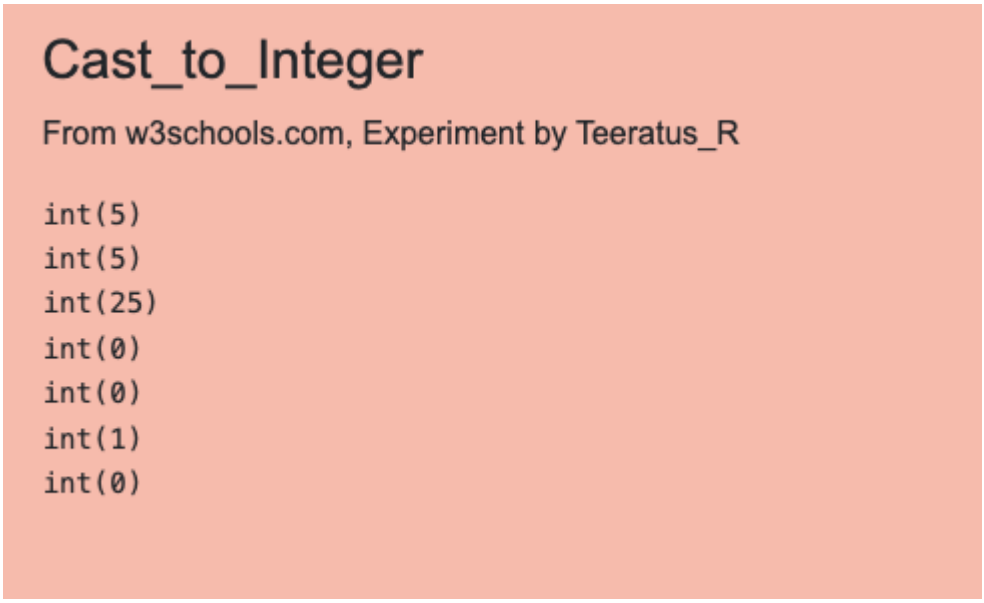
Example 1

Result View Example

PHP Cast to Integer

```
<h2>Cast_to_Integer</h2>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = "25 kilometers"; // String
$d = "kilometers 25"; // String
$e = "hello"; // String
$f = true;    // Boolean
$g = NULL;    // NULL

$a = (int) $a;
$b = (int) $b;
$c = (int) $c;
$d = (int) $d;
$e = (int) $e;
$f = (int) $f;
$g = (int) $g;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
var_dump($f);
```

```
var_dump($g);
?>
</pre>
```



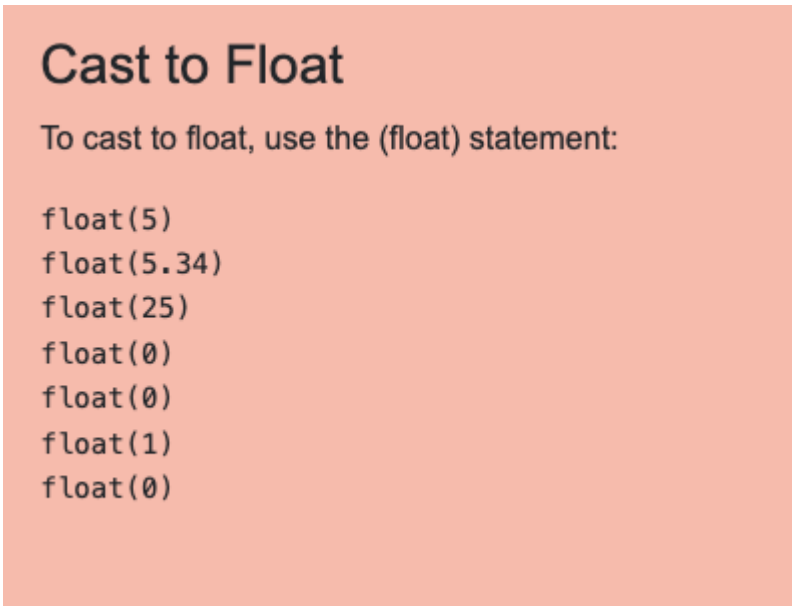
Example 2

Result View Example

PHP Cast to Float

```
<h2>Cast to Float</h2>
        <p>To cast to float, use the (float) statement: </p>
        <pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = "25 kilometers"; // String
$d = "kilometers 25"; // String
$e = "hello"; // String
$f = true;    // Boolean
$g = NULL;    // NULL

$a = (float) $a;
$b = (float) $b;
$c = (float) $c;
$d = (float) $d;
$e = (float) $e;
$f = (float) $f;
$g = (float) $g;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
```

```
var_dump($d);
var_dump($e);
var_dump($f);
var_dump($g);
?>
</pre>
```



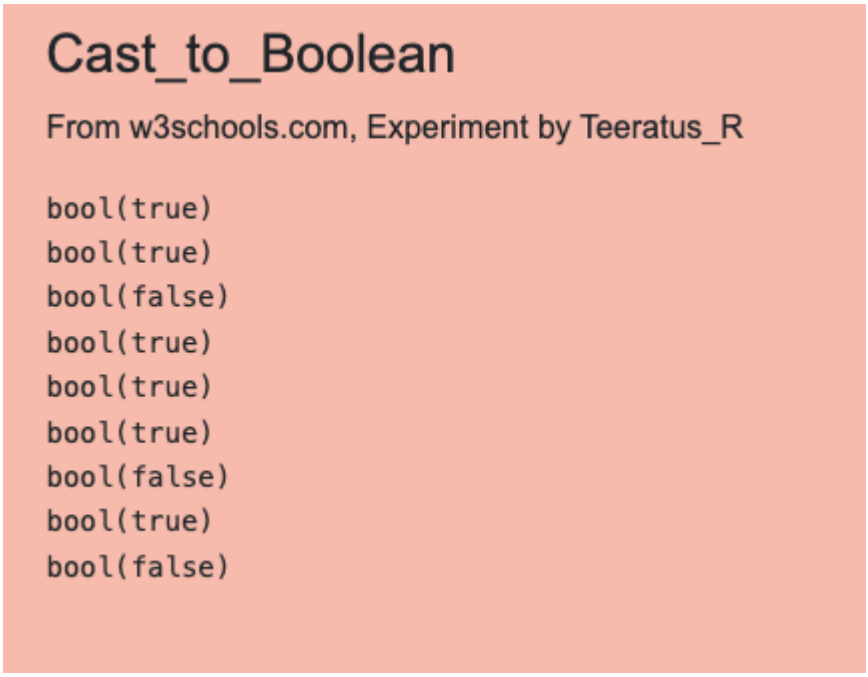
Example 3

Result View Example

PHP Cast to Boolean

```
<h2>Cast_to_Boolean</h2>
        <p>From w3schools.com, Experiment by Teeratus_R </p>
        <pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = 0;       // Integer
$d = -1;      // Integer
$e = 0.1;     // Float
$f = "hello"; // String
$g = "";      // String
$h = true;    // Boolean
$i = NULL;    // NULL

$a = (bool) $a;
$b = (bool) $b;
$c = (bool) $c;
$d = (bool) $d;
$e = (bool) $e;
$f = (bool) $f;
$g = (bool) $g;
```

```
$h = (bool) $h;
$i = (bool) $i;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
var_dump($f);
var_dump($g);
var_dump($h);
var_dump($i);
?>
</pre>
```



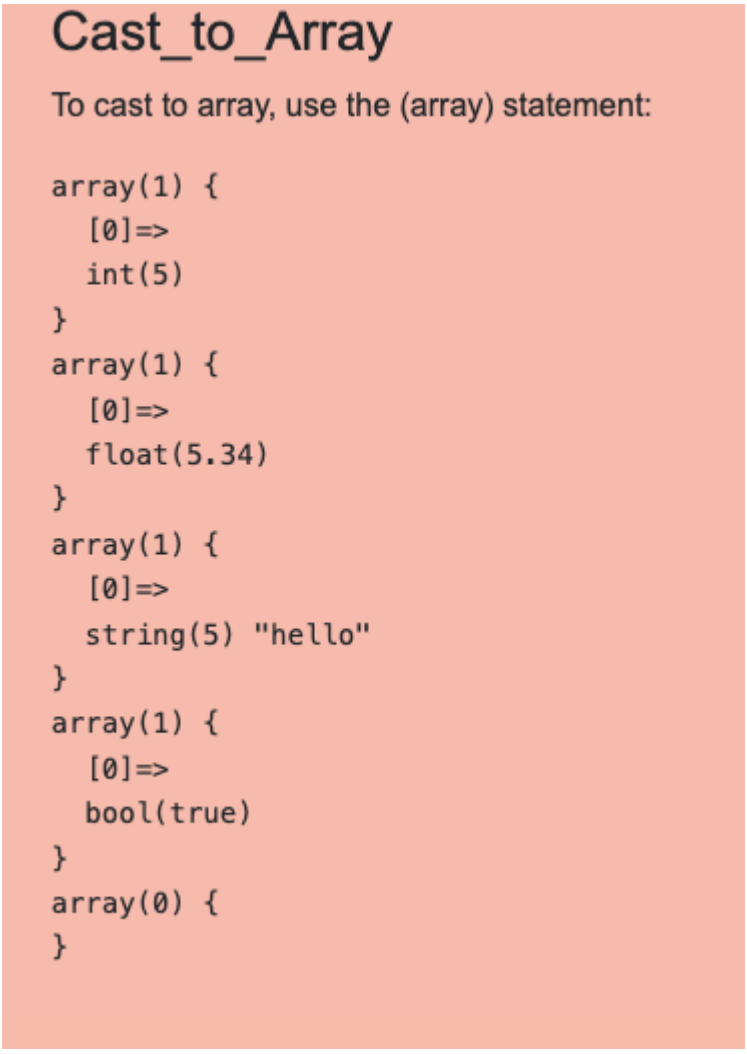
Example 4

Result View Example

PHP Cast to Array

```
<h2>Cast_to_Array</h2>
        <p>To cast to array, use the (array) statement:</p>
        <pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = "hello"; // String
$d = true;    // Boolean
$e = NULL;    // NULL
```

```
$a = (array) $a;
$b = (array) $b;
$c = (array) $c;
$d = (array) $d;
$e = (array) $e;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
?>
</pre>
```



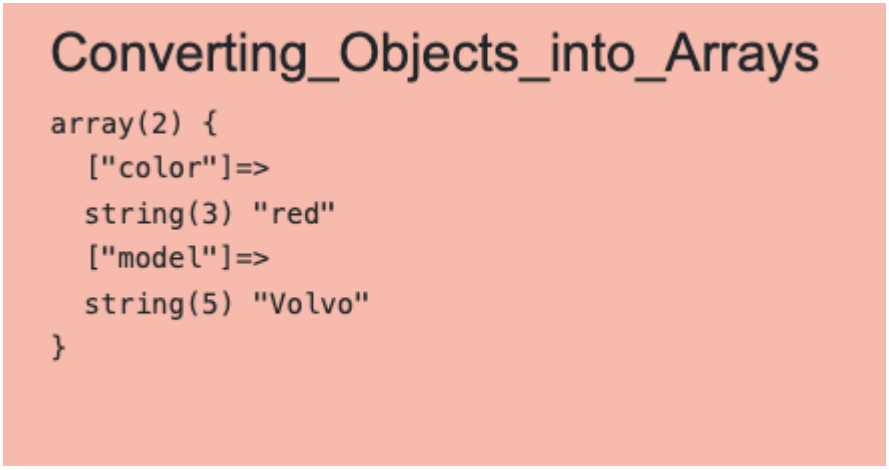
Example 5

Result View Example

PHP Converting Objects into Arrays

```
<h2>Converting_Objects_into_Arrays</h2>
```

```
<pre>
<?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");

$myCar = (array) $myCar;
var_dump($myCar);?>
</pre>
```



Example 6

Result View Example

PHP Cast to Object

```
<h2>Cast_to_Object</h2>
        <p>To cast to object, use the (object) statement: </p>
        <pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = "hello"; // String
$d = true;    // Boolean
$e = NULL;    // NULL

$a = (object) $a;
$b = (object) $b;
```

```
$c = (object) $c;
$d = (object) $d;
$e = (object) $e;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
?>
</pre>
```

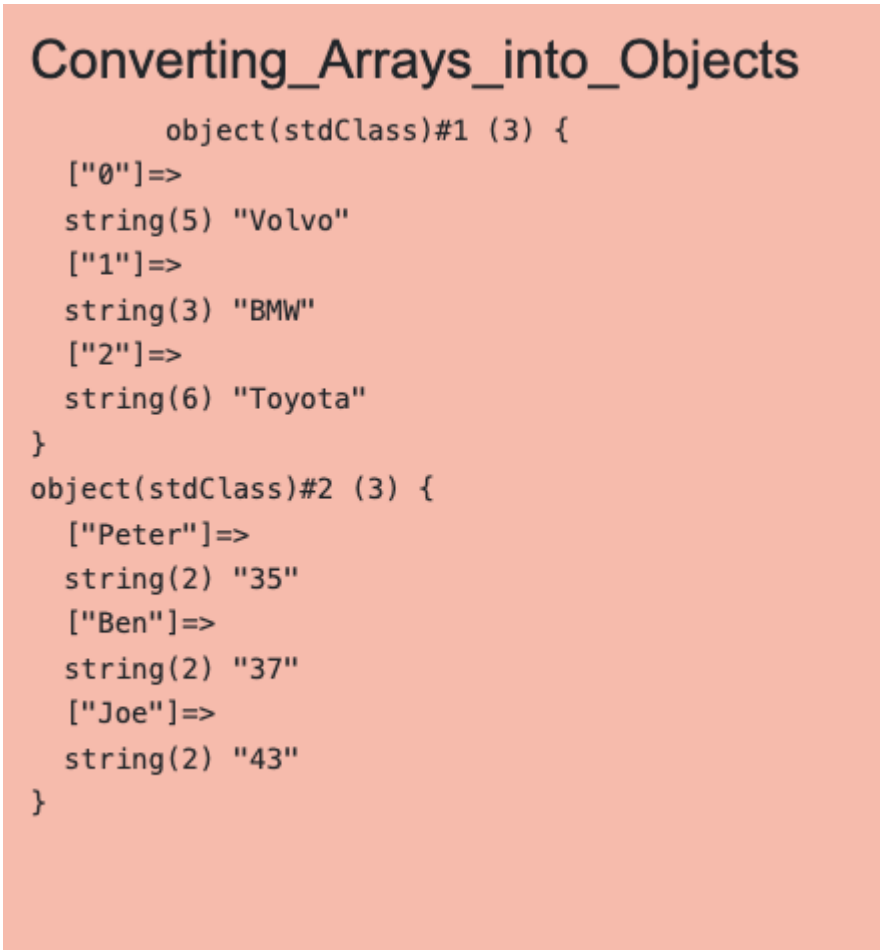
Example 7

Result View Example

PHP Converting Arrays into Objects

```
<h2>Converting_Arrays_into_Objects</h2>

<pre>
```

```
<?php
$a = array("Volvo", "BMW", "Toyota"); // indexed array
$b = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43"); // associative array

$a = (object) $a;
$b = (object) $b;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
?>
</pre>
```



Example 8

Result View Example

PHP Cast to NULL

```
<h2>Cast to NULL</h2>
<p>To cast to NULL, use the (unset) statement:</p>
<pre>
<?php
$a = 5;       // Integer
$b = 5.34;    // Float
```

```
$c = "hello"; // String
$d = true;    // Boolean
$e = NULL;    // NULL

unset($a);
unset($b);
unset($c);
unset($d);
unset($e);

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
?>
</pre>
```

Cast to NULL

To cast to NULL, use the (unset) statement:

```
Warning: Undefined variable $a in /Users/teeratus_r/GoogleDri
NULL

Warning: Undefined variable $b in /Users/teeratus_r/GoogleDri
NULL

Warning: Undefined variable $c in /Users/teeratus_r/GoogleDri
NULL

Warning: Undefined variable $d in /Users/teeratus_r/GoogleDri
NULL

Warning: Undefined variable $e in /Users/teeratus_r/GoogleDri
NULL
```

Example 9

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Math

Table of Contents

PHP Math

```
<h2>PHP Math</h2>
        <p>PHP has a set of math functions that allows you to perform
mathematical tasks on numbers.</p>

        <h4>PHP pi() Function</h4>
        <p>The pi() function returns the value of PI: </p>
        <?php
        echo(pi());
        ?>
        <h4>PHP min() and max() Functions</h4>
        <p>The min() and max() functions can be used to find the lowest or
highest value in a list of arguments: </p>
        <?php
        echo(min(0, 150, 30, 20, -8, -200));
        echo(max(0, 150, 30, 20, -8, -200));
        ?>
        <h4>PHP abs() Function</h4>
        <p>The abs() function returns the absolute (positive) value of a
number: </p>
        <?php
        echo(abs(-6.7));
        ?>
        <h4>PHP sqrt() Function</h4>
        <p>The sqrt() function returns the square root of a number: </p>
        <?php
        echo(sqrt(64));
        ?>
        <h4>PHP round() Function</h4>
        <p>The round() function rounds a floating-point number to its
nearest integer: </p>
        <?php
        echo(round(0.60));
        echo(round(0.49));
        ?>
        <h4>Random Numbers</h4>
        <p>The rand() function generates a random number: </p>
        <?php
        echo(rand());
```

```
?>
```

PHP Math

PHP has a set of math functions that allows you to perform mathematical tasks on numbers.

PHP pi() Function

The pi() function returns the value of PI:

3.1415926535898

PHP min() and max() Functions

The min() and max() functions can be used to find the lowest or highest value in a list of arguments:

-200150

PHP abs() Function

The abs() function returns the absolute (positive) value of a number:

6.7

PHP sqrt() Function

The sqrt() function returns the square root of a number:

8

PHP round() Function

The round() function rounds a floating-point number to its nearest integer:

10

Random Numbers

The rand() function generates a random number:

1018764240

Example 1

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Constants

Table of Contents

PHP 1 Create a PHP Constant

```
<h2>php Constants-Create_a_PHP_Constant</h2>
<p>To create a constant, use the define() function.</p>

<?php
    // case-sensitive constant name
    define("GREETING", "Welcome to W3Schools.com!");
    echo GREETING;
?>
```

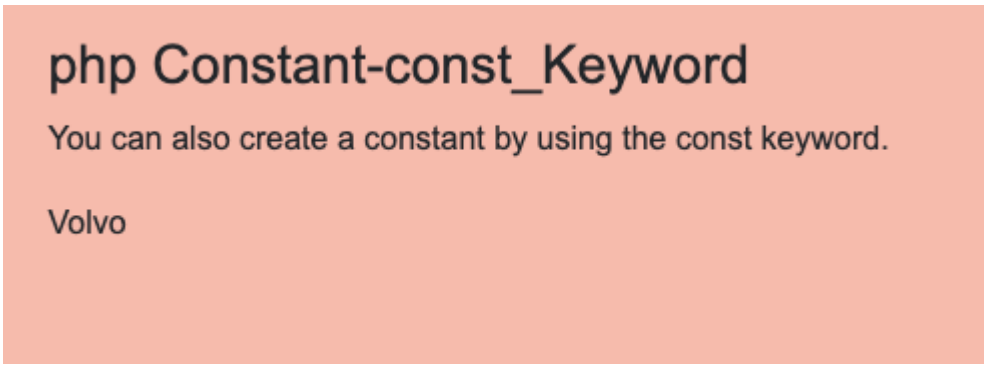
Example 1

Result View Example

PHP 2 const Keyword

```
<h2>php Constant-const_Keyword</h2>
<p>You can also create a constant by using the const keyword.</p>
```

```php
<?php
    // Create a case-sensitive constant with the const keyword:
    const MYCAR = "Volvo";

    echo MYCAR;
?>
```

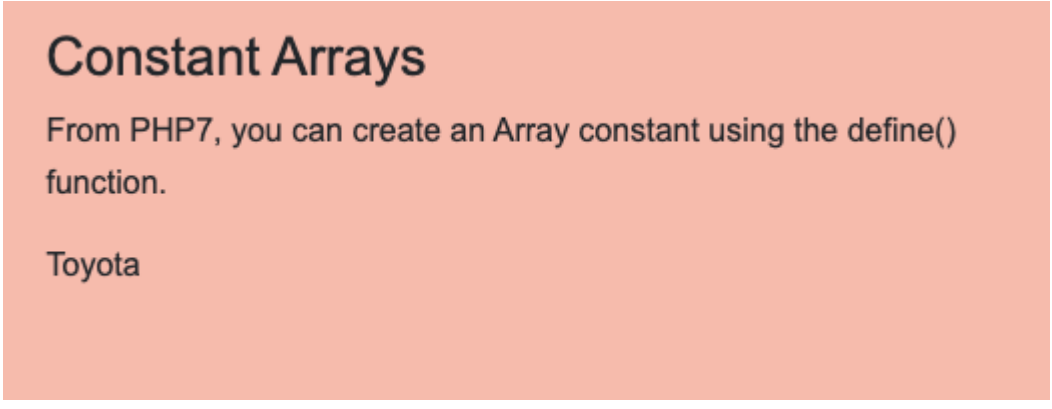


Example 2

Result View Example

PHP 3 Constant Arrays

```php
<h2>Constant Arrays</h2>
<p>From PHP7, you can create an Array constant using the define()
function.</p>

<?php
    define("cars", [
        "Alfa Romeo",
        "BMW",
        "Toyota"
    ]);
    echo cars[2];
?>
```

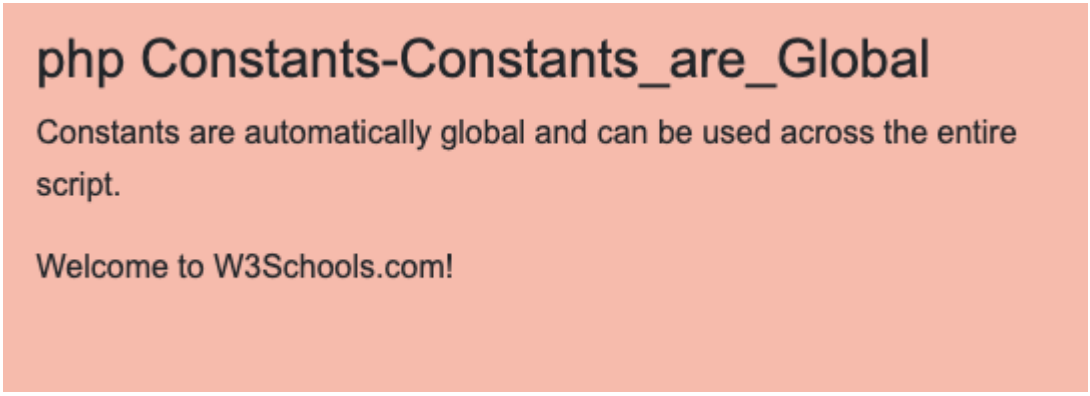


Example 3

Result View Example

PHP 4 Constants are Global

```
<h2>php Constants-Constants_are_Global</h2>
<p>Constants are automatically global and can be used across the entire
script.</p>

<?php
// This example uses a constant inside a function, even if it is defined
outside the function:

    define("GREETING", "Welcome to W3Schools.com!");

    function myTest() {
    echo GREETING;
    }

    myTest();
?>
```



Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Magic Constants

Table of Contents

Magic Constants

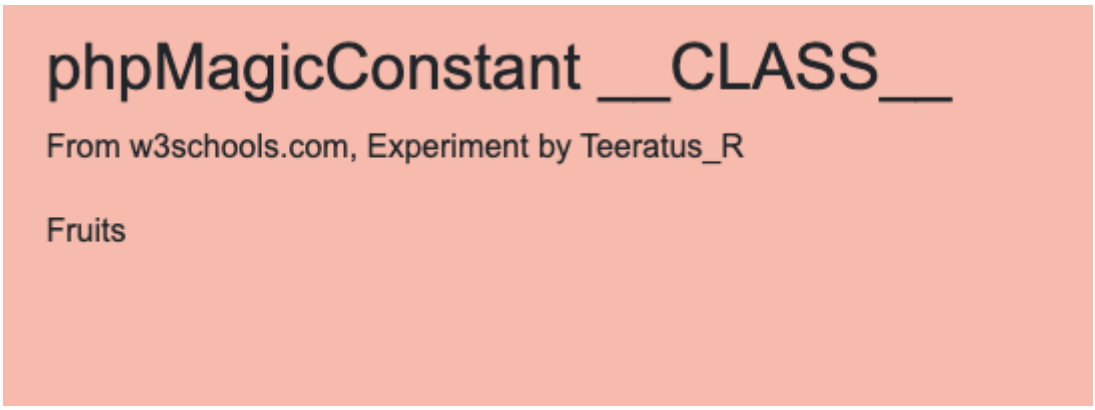
Here are the magic constants, with descriptions and examples:

Constant	Description
`__CLASS__`	If used inside a class, the class name 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.
`__FILE__`	The full path and filename of the file. If used inside an include, the name of the included file is returned.
`__FUNCTION__`	The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
`__LINE__`	The current line number of the file.
`__METHOD__`	The class method name.
`__NAMESPACE__`	The name of the current namespace. (Added in PHP 5.3.0)

PHP 1 MagicConstant CLASS

```
<h2>phpMagicConstant __CLASS__ </h2>
<p>If used inside a class, the class name is returned </p>
```

```php
<?php
class Fruits
{
    public function myValue()
    {
        return __CLASS__;
    }
}
$kiwi = new Fruits();
echo $kiwi->myValue();
?>
```

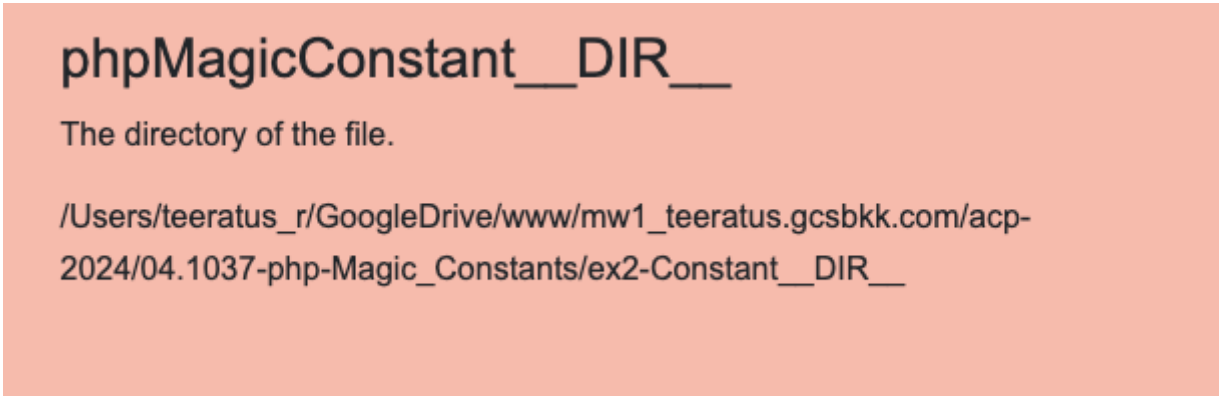


Example 1

Result View Example

PHP 2 MagicConstant DIR

```php
<h2>phpMagicConstant__DIR__</h2>
<p>The directory of the file.</p>

<?php
    echo __DIR__;
?>
```



Example 2

Result View Example

PHP 3 MagicConstant FILE

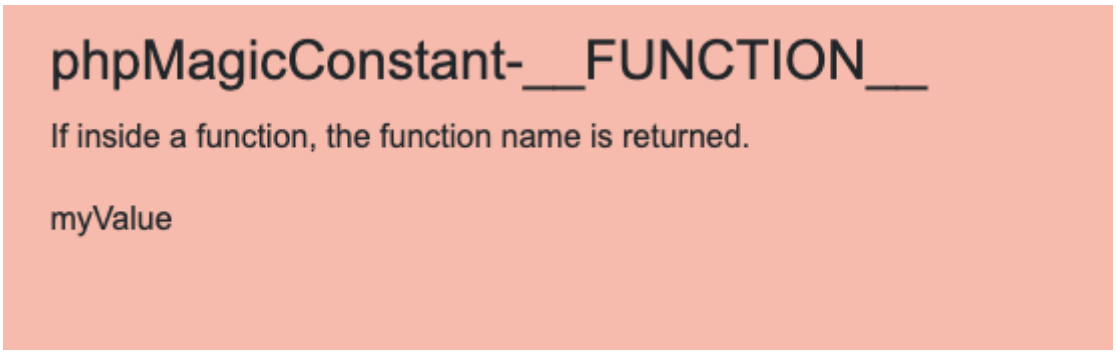
```
<h2>phpMagicConstant-__FILE__</h2>
<p>The file name including the full path.</p>

<?php
    echo __FILE__;
?>
```

Example 3

Result View Example

PHP 4 MagicConstant FUNCTION

```
<h2>phpMagicConstant-__FUNCTION__</h2>
<p>If inside a function, the function name is returned.</p>

<?php
    function myValue(){
        return __FUNCTION__;
    }
    echo myValue();
?>
```

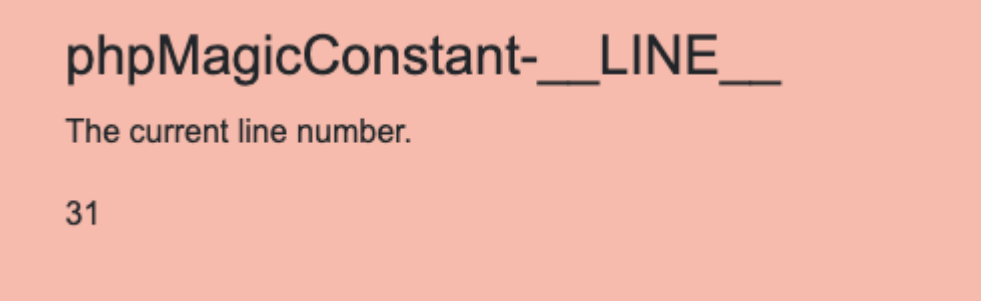


Example 4

Result View Example

PHP 5 MagicConstant LINE

```
<h2>phpMagicConstant-__LINE__</h2>
<p>The current line number.</p>

<?php
    echo __LINE__;
?>
```



Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP-Operators

Back to ACP page

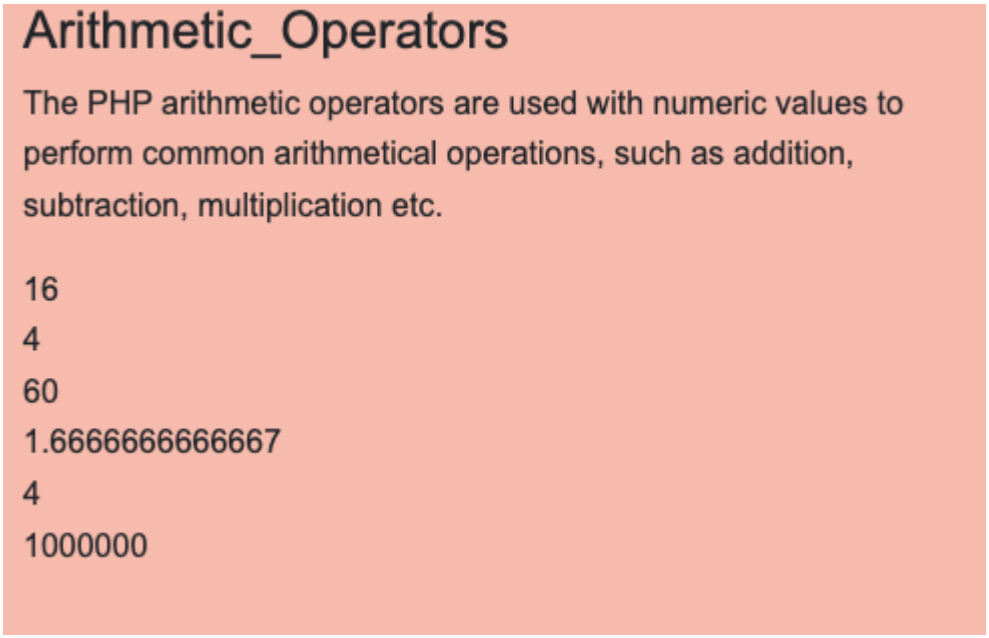
Table of Contents

PHP Arithmetic Operators

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

```php
<?php
    $x = 10;
    $y = 6;

    // Addition
    echo $x + $y; // Output: 16
    echo "<br>";

    // Subtraction
    echo $x - $y; // Output: 4
    echo "<br>";

    // Multiplication
    echo $x * $y; // Output: 60
    echo "<br>";
```

```
    // Division
    echo $x / $y; // Output: 1.6666666666667
    echo "<br>";

    // Modulus
    echo $x % $y; // Output: 4
    echo "<br>";

    // Exponentiation
    echo $x ** $y; // Output: 1000000
    echo "<br>";

?>
```



Example 1

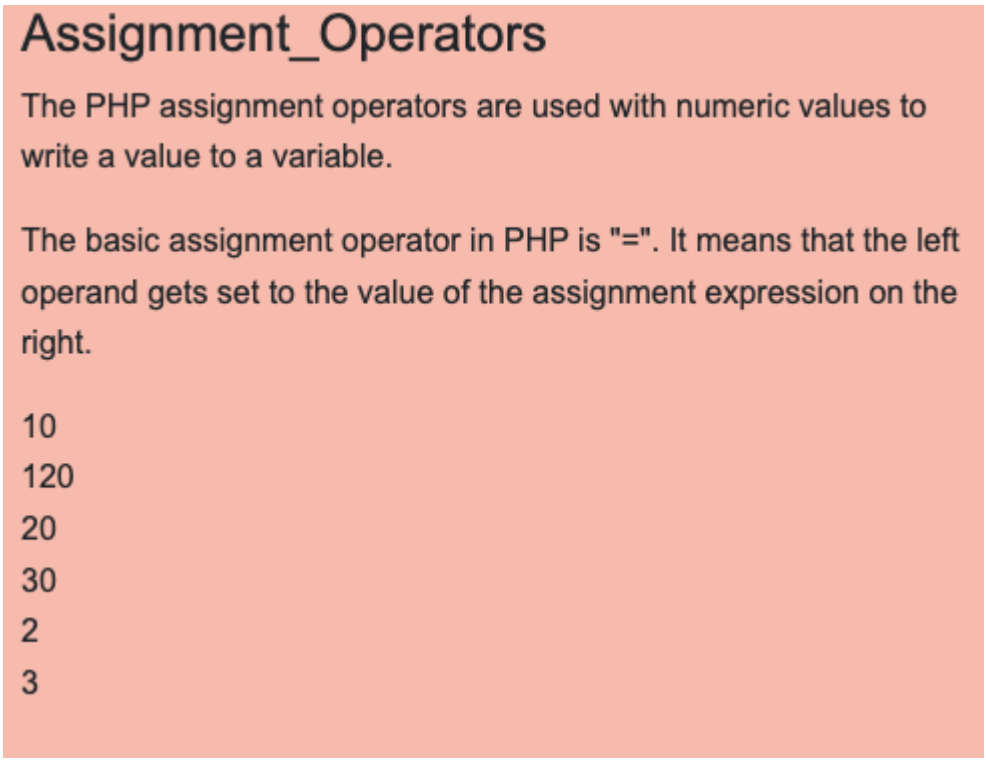
Result View Example

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

```
<?php
  $x = 10;    // Assignment
  echo $x;    // Output: 10

  echo "<br>";

  $x = 20;
  $x += 100;  // Addition
  echo $x;    // Output: 120


  echo "<br>";
```

```php
    $x = 50;
    $x -= 30;   // Subtraction
    echo $x;    // Output: 20

    echo "<br>";

    $x = 5;
    $x *= 6;    // Multiplication
    echo $x;    // Output: 30

    echo "<br>";

    $x = 10;
    $x /= 5;    // Division
    echo $x;    // Output: 2

    echo "<br>";

    $x = 15;
    $x %= 4;    // Modulus
    echo $x;    // Output: 3
?>
```



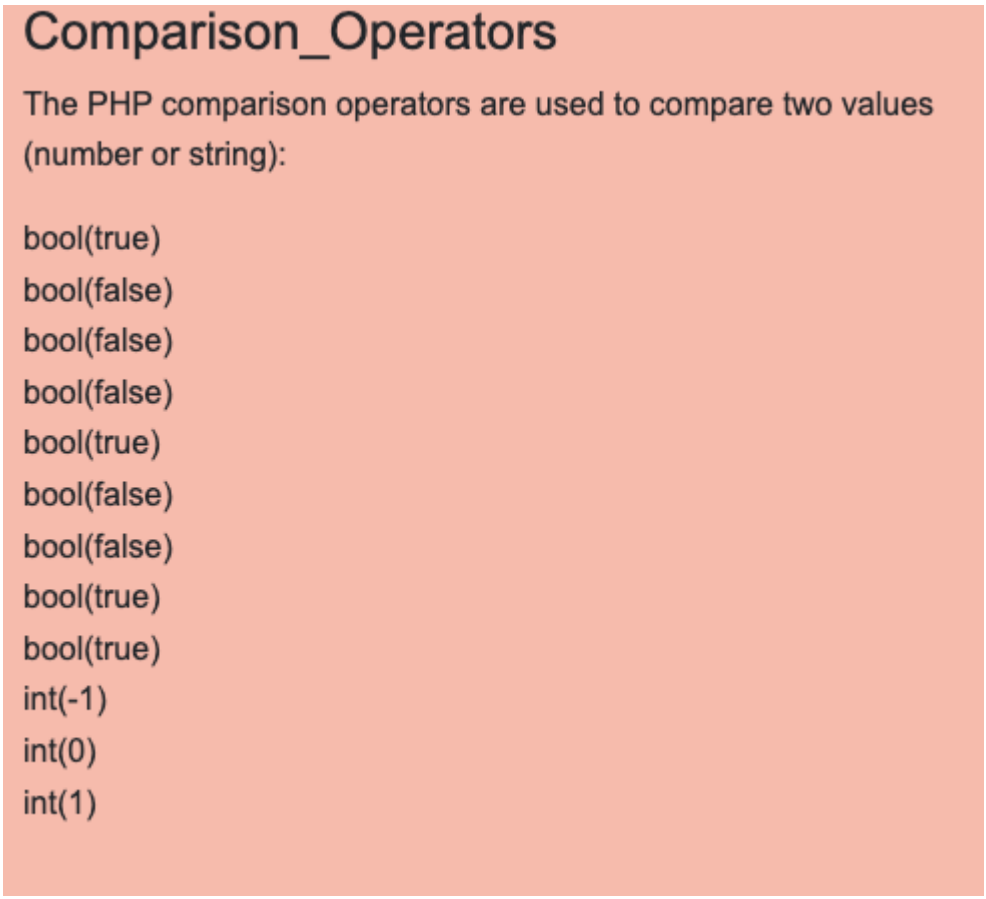
Example 2

Result View Example

PHP Comparison Operators

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

```
<h2>Comparison_Operators</h2>
<p>The PHP comparison operators are used to compare two values (number or
string): </p>
<?php
    $x = 100;
    $y = "100";

    var_dump($x == $y); // Equal: returns true because values are equal
    echo "<br>";
    var_dump($x === $y); // Identical: returns false because types are not
equal
    echo "<br>";
    var_dump($x != $y); // Not equal: returns false because values are
equal
    echo "<br>";
    var_dump($x <> $y); // Not equal: returns false because values are
equal
    echo "<br>";
    var_dump($x !== $y); // Not identical: returns true because types are
not equal
    echo "<br>";

    var_dump($x > $y); // Greater than: returns false because $x is not
greater than $y
    echo "<br>";
    var_dump($x < $y); // Less than: returns false because $x is not less
than $y
    echo "<br>";

    var_dump($x <= $y); // Less than or equal to: returns true because $x
is less than or equal to $y
    echo "<br>";
    var_dump($x >= $y); // Greater than or equal to: returns true because
$x is greater than or equal to $y
    echo "<br>";

    $x = 5;
    $y = 10;
    var_dump($x <=> $y); // Spaceship: returns -1 because $x is less than
$y
    echo "<br>";
    $x = 10;
    $y = 10;
    var_dump($x <=> $y); // Spaceship: returns 0 because values are equal
    echo "<br>";
    $x = 15;
    $y = 10;
    var_dump($x <=> $y); // Spaceship: returns +1 because $x is greater
than $y
    echo "<br>";

?>
```



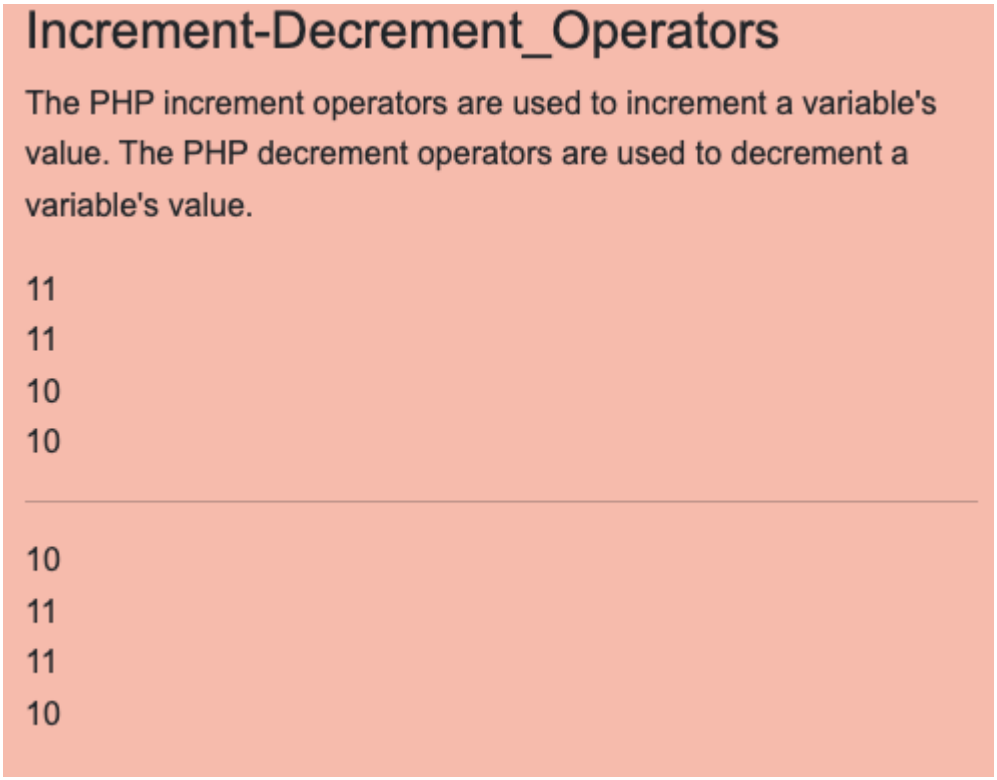
Example 3

Result View Example

PHP Increment-Decrement Operators

```
<h2>Increment-Decrement_Operators</h2>
<p>The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.</p>

<?php
    $x = 10;
    echo ++$x; // Output: 11
    echo "<br>";
    echo $x; // Output: 11
    echo "<br>";
    echo --$x; // Output: 10
    echo "<br>";
    echo $x; // Output: 10
    echo "<hr>";
    $x = 10;
    echo $x++; // Output: 10
    echo "<br>";
    echo $x; // Output: 11
    echo "<br>";
    echo $x--; // Output: 11
```

```
    echo "<br>";
    echo $x; // Output: 10
?>
```



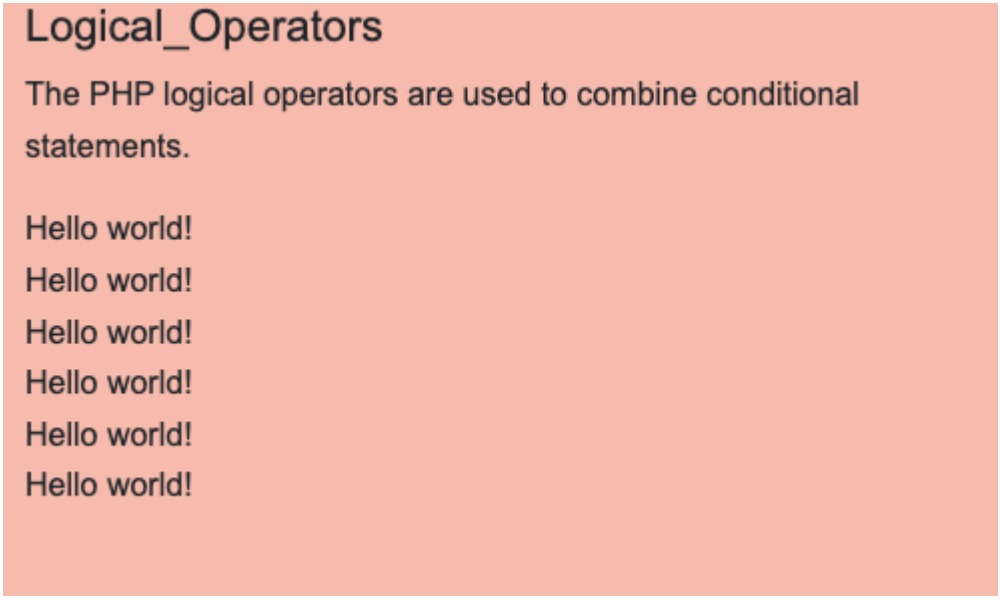
Example 4

Result View Example

PHP Logical Operators

```
<h4>Logical_Operators</h4>
<p>The PHP logical operators are used to combine conditional statements.
</p>
<?php
    $x = 100;
    $y = 50;

    // and operator
    if ($x == 100 and $y == 50) {
        echo "Hello world!"."<br>";
    }

    // or operator
    if ($x == 100 or $y == 80) {
        echo "Hello world!"."<br>";
    }

    // xor operator
    if ($x == 100 xor $y == 80) {
        echo "Hello world!"."<br>";
```

```
    }

    // not operator
    if ($x !== 90) {
        echo "Hello world!" ."<br>";
    }

    // && operator
    if ($x == 100 && $y == 50) {
        echo "Hello world!" ."<br>";
    }

    // || operator
    if ($x == 100 || $y == 80) {
        echo "Hello world!" ."<br>";
    }
?>
```



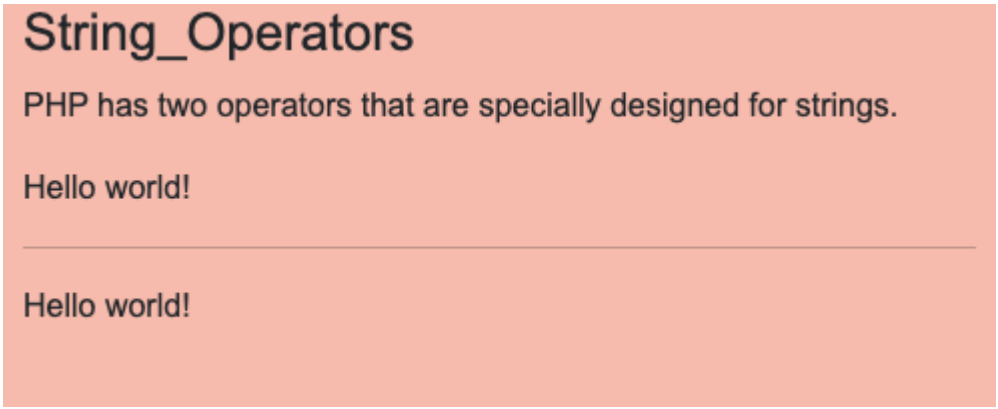
Example 5

Result View Example

PHP String Operators

```
<h2>String_Operators</h2>
<p>PHP has two operators that are specially designed for strings. </p>
<?php
    $txt1 = "Hello";
    $txt2 = " world!";
    // Concatenation
    echo $txt1 . $txt2;
    echo "<hr>";
    // Concatenation assignment
    $txt1 .= $txt2;
```

```
    echo $txt1;
?>
```



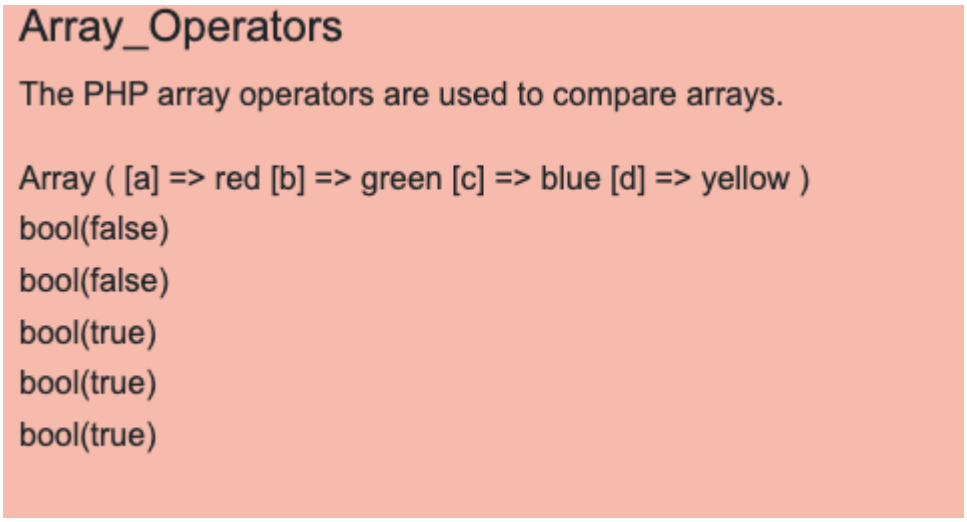
Example 6

Result View Example

Array Operators

Lorem ipsum dolor sit, amet consectetur adipisicing elit.

```
<h4>Array_Operators</h4>
<p>The PHP array operators are used to compare arrays. </p>
<?php
    $x = array("a" => "red", "b" => "green");
    $y = array("c" => "blue", "d" => "yellow");
    // $y = array("a" => "red", "b" => "green");
    // union
    print_r($x + $y); // union of $x and $y
    echo "<br>";
    // equality
    var_dump($x == $y); // Returns true if $x and $y have the same
key/value pairs
    echo "<br>";
    // identity
    var_dump($x === $y); // Returns true if $x and $y have the same
key/value pairs in the same order and of the same types
    echo "<br>";
    // inequality
    var_dump($x != $y); // Returns true if $x is not equal to $y
    echo "<br>";
    // inequality
    var_dump($x <> $y); // Returns true if $x is not equal to $y
    echo "<br>";
    // non-identity
    var_dump($x !== $y); // Returns true if $x is not identical to $y
    echo "<br>";
?>
```

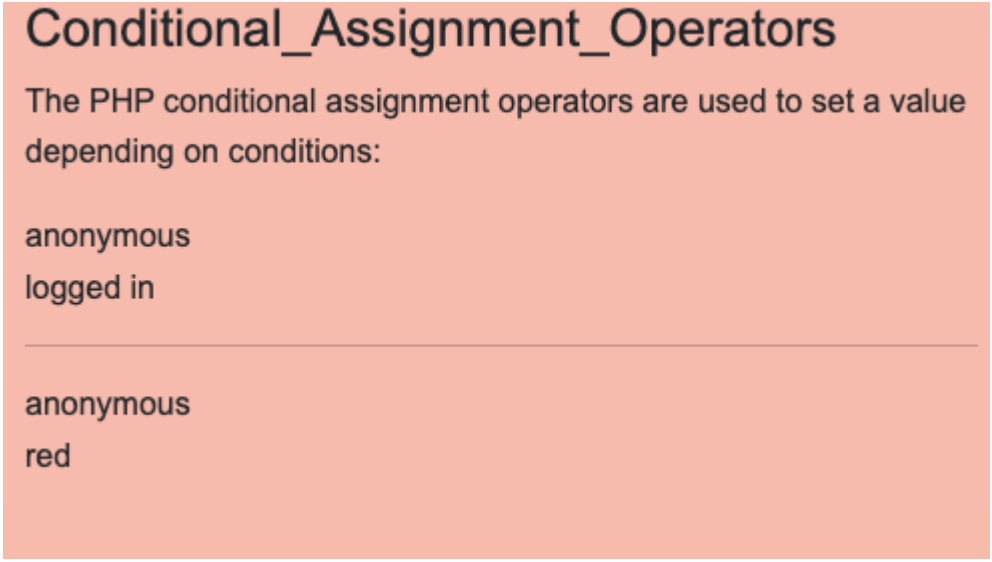


Example 7

Result View Example

PHP Conditional Assignment Operators

```
<h2>Conditional_Assignment_Operators</h2>
<p>The PHP conditional assignment operators are used to set a value
depending on conditions:</p>

<?php
    // if empty($user) = TRUE, set $status = "anonymous"
    echo $status = (empty($user)) ? "anonymous" : "logged in";
    echo("<br>");

    $user = "John Doe";
    // if empty($user) = FALSE, set $status = "logged in"
    echo $status = (empty($user)) ? "anonymous" : "logged in";

    echo("<hr>");
    // variable $user is the value of $_GET['user']
    // and 'anonymous' if it does not exist
    echo $user = $_GET["user"] ?? "anonymous";
    echo("<br>");

    // variable $color is "red" if $color does not exist or is null
    echo $color = $color ?? "red";

?>
```



Example 8

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP if Statement

Table of Contents

PHP if Statement

The `if` statement is used to execute some code if a condition is `true`.

```
<h2>if Statement</h2>
<p>The if statement executes some code if one condition is true.</p>
<?php
$t = 14;

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

Example 1

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP if Operators

Table of Contents

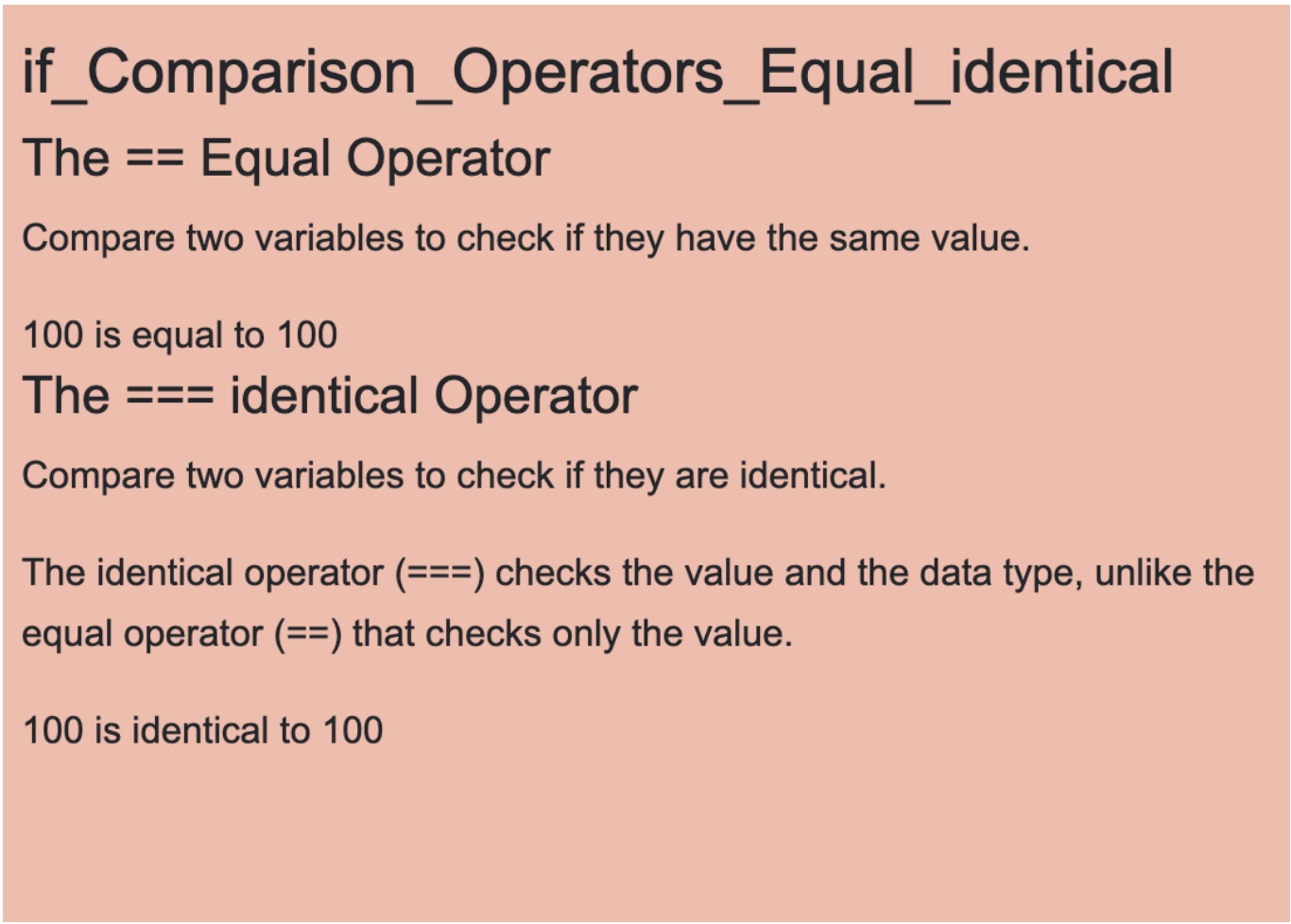
PHP 11 Comparison Operators Equal - identical

To compare two values, we need to use a comparison operator. Here are the PHP comparison operators to use in if statements:

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
>	Greater than	$x > $y	True if $x is greater than $y
<	Less than	$x < $y	True if $x is less than $y
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

```
<h2>if_Comparison_Operators_Equal_identical</h2>
<h4>The == Equal Operator</h4>
<p>Compare two variables to check if they have the same value.</p>
<?php
$x = 100;
$y = 100;
if ($x == $y) {
  echo "$x is equal to $y";
}
?>
<h4>The === identical Operator</h4>
```

```
<p>Compare two variables to check if they are identical.</p>
<p>The identical operator (===) checks the value and the data type, unlike
the equal operator (==) that checks only the value.</p>
<?php
$x = 100;
$y = 100;
if ($x === $y) {
  echo "$x is identical to $y";
}
?>
```



Example 11

Result View Example

PHP 12 Comparison Operators Not equal - NOT identical

Operator	Name	Example	Result
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type

```
<h2>if Comparison Operators Not equal - NOT identical</h2>

<h4>The != Not equal Operator</h4>
<p>Compare two variables and write a message if they don't have the same
value.</p>
<?php
$x = 100;
$y = 50;
if ($x != $y) {
  echo "$x is not equal to $y";
}
?>
<h4>The <> Not equal Operator</h1>
<p>Compare two variables and write a message if they don't have the same
value.</p>
<?php
$x = 100;
$y = 50;
if ($x <> $y) {
  echo "$x is not equal to $y";
}
?>
<h4>The !== NOT identical Operator</h4>
<p>Compare two variables and write a message if they are NOT identical.
</p>
<p>The not identical operator (!==) checks the value and the data type,
unlike the not equal operator (!=) that checks only the value.</p>
<?php
$x = 100;
$y = 50;
if ($x !== $y) {
  echo "$x is not identical to $y";
}
?>
```

if Comparison Operators Not equal - NOT identical

The != Not equal Operator

Compare two variables and write a message if they don't have the same value.

100 is not equal to 50

The <> Not equal Operator

Compare two variables and write a message if they don't have the same value.

100 is not equal to 50

The !== NOT identical Operator

Compare two variables and write a message if they are NOT identical.

The not identical operator (!==) checks the value and the data type, unlike the not equal operator (!=) that checks only the value.

100 is not identical to 50

Example 12

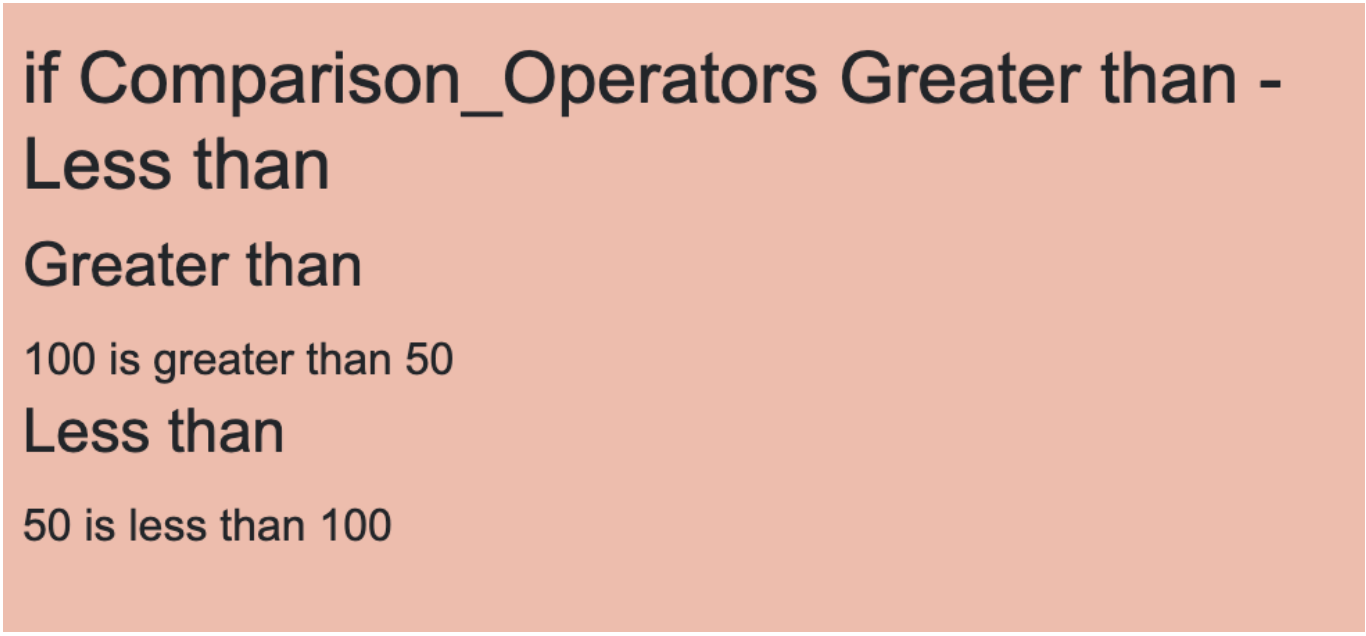
Result View Example

PHP 13 Comparison Operators Greater than - Less than

Operator	Name	Example	Result
>	Greater than	$x > $y	True if $x is greater than $y
<	Less than	$x < $y	True if $x is less than $y

```
<h2>if Comparison_Operators Greater than – Less than </h2>
```

```
<h4>Greater than</h4>
<?php
    $x = 100;
$y = 50;

if ($x > $y) {
  echo "$x is greater than $y";
}
?>

<h4>Less than</h4>
<?php
    $x = 100;
$y = 50;

if ($y < $x) {
  echo "$y is less than $x";
}
```



Example 13

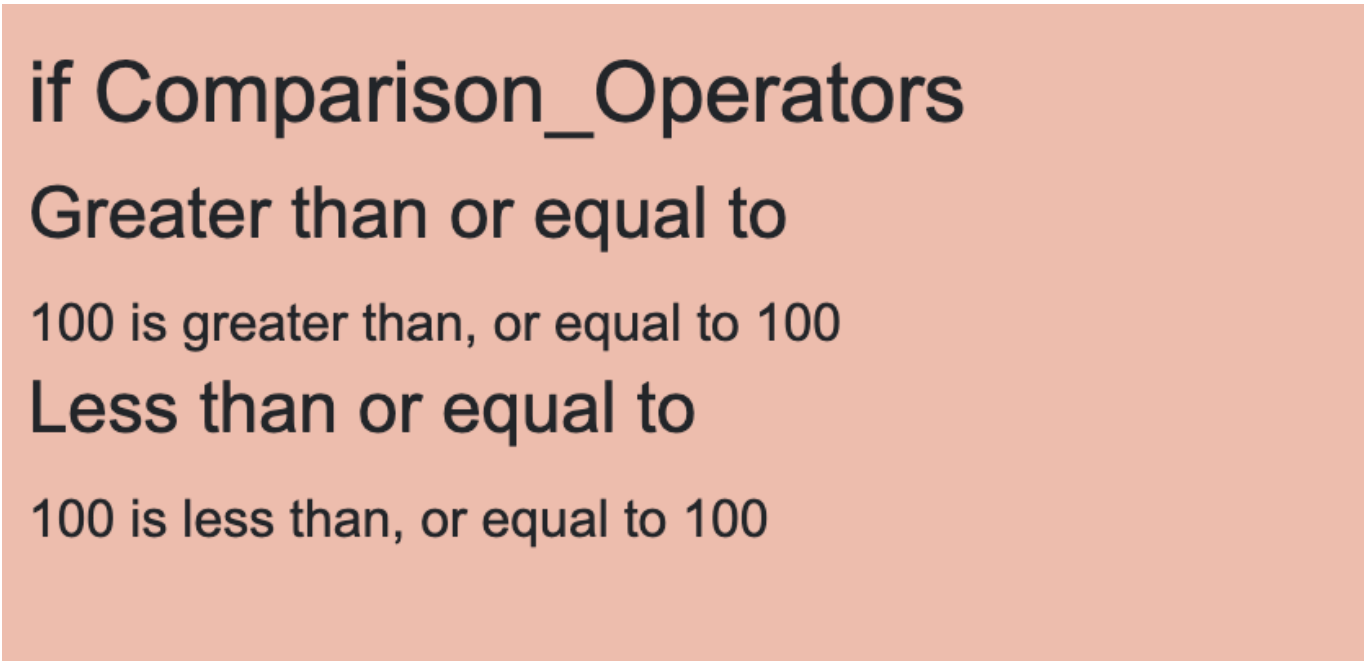
Result View Example

Greater_than_or_equal_to_-_Less_than_or_equal_to

PHP 14 Comparison Operators Greater than or equal to - Less than or equal to

Operator	Name	Example	Result
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

```
<h4>Greater than or equal to</h4>
<?php
$x = 100;
$y = 100;
if ($x >= $y) {
  echo "$x is greater than, or equal to $y";
}
?>

<h4>Less than or equal to</h4>
<?php
$x = 100;
$y = 100;
if ($y <= $x) {
  echo "$y is less than, or equal to $x";
}
?>
```



Example 14

Result View Example

PHP 21 Logical Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
&&	And	$x && $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
||	Or	$x || $y	True if either $x or $y is true

Operator	Name	Example	Result
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
!	Not	!$x	True if $x is not true

```
 <h2>Logical Operators</h2>
        <p>To check more than one condition, we can use logical
operators</p>

        <h4>and - && And</h4>
        <?php
        $x = 100;
        $y = 50;

        if ($x == 100 and $y == 50) {
            echo "Hello world! and 1";
        }
        if ($x == 100 && $y == 50) {
            echo "Hello world! and 2";
        }
        ?>
        <h4>or - || Or</h4>
        <?php
        $x = 100;
        $y = 50;

        if ($x == 100 or $y == 80) {
            echo "Hello world! or 1";
        }
        if ($x == 100 || $y == 80) {
            echo "Hello world! or 2";
        }
        ?>
        <h4>xor - Xor</h4>
        <?php
        $x = 100;
        $y = 50;
        if ($x == 100 xor $y == 80) {
            echo "Hello world! Xor";
        }
        ?>
        <h4>! Not</h4>
        <?php
        $x = 100;

        if (!($x == 90)) {
            echo "Hello world! Not";
        }
        ?>
```

Document

Document in project

You can Download PDF file.

Reference

PHP if else Statements

Table of Contents

PHP 1 if else Statement

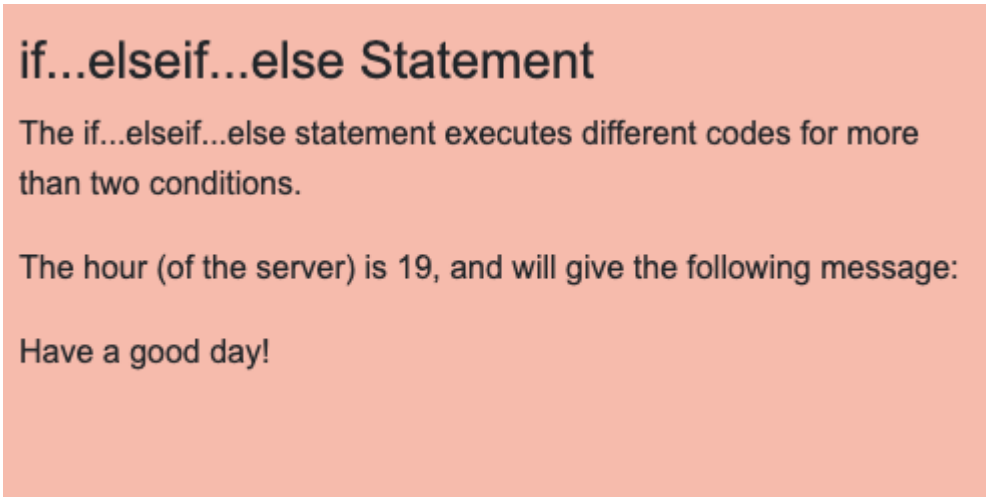
```
<h2>if...else_Statement</h2>
<p>The if...else statement executes some code if a condition is true and
another code if that condition is false.</p>
<?php
$t = date("H");

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

Example 1

Result View Example

PHP 2 if elseif else Statement

```
<h2>if...elseif...else Statement</h2>
<p>The if...elseif...else statement executes different codes for more than
two conditions. </p>
<?php
// $t = date("H");
$t = 19;
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!";
}
?>
```



Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

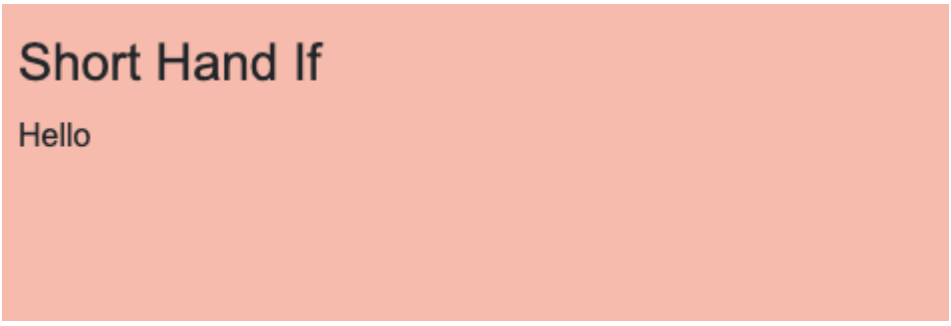
PHP Shorthand if Statements

Back to ACP page

Table of Contents

PHP 1 Short Hand If

```
<h2>Short Hand If</h2>
<?php
$a = 5;
// $b= "Hi";
if ($a < 10) $b = "Hello";
echo $b
?>
```



Example 1

Result View Example

PHP 2 Short Hand If Else

```
<h2>Short Hand If Else</h2>
<p> </p>
<?php
$a = 13;
$b = $a < 10 ? "Hello" : "Good Bye";
echo $b;
?>
```

Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

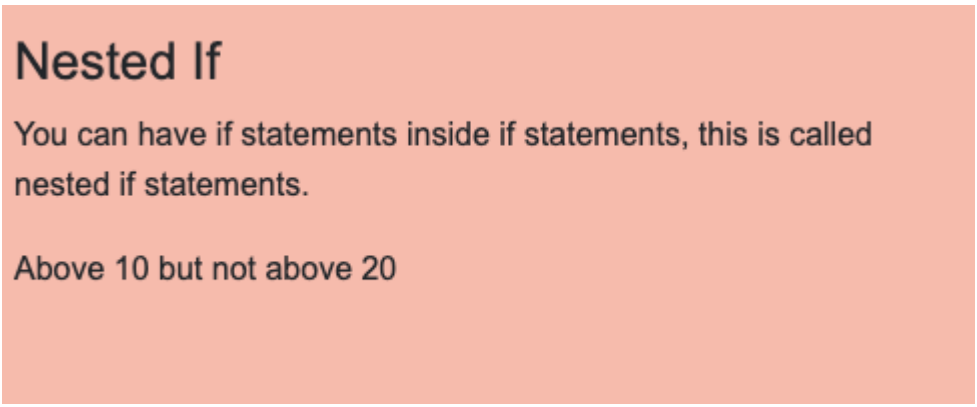
PHP Nested if Statement

Back to ACP page

Table of Contents

PHP 1 Nested If

```
<h2>Nested If</h2>
        <p>You can have if statements inside if statements, this is called
nested if statements.</p>
        <?php
$a = 13;

if ($a > 10) {
  echo "Above 10";
  if ($a > 20) {
    echo " and also above 20";
  } else {
    echo " but not above 20";
  }
}
?>
```



Example 1

Result View Example

Document

Document in project

You can Download PDF file.

Reference

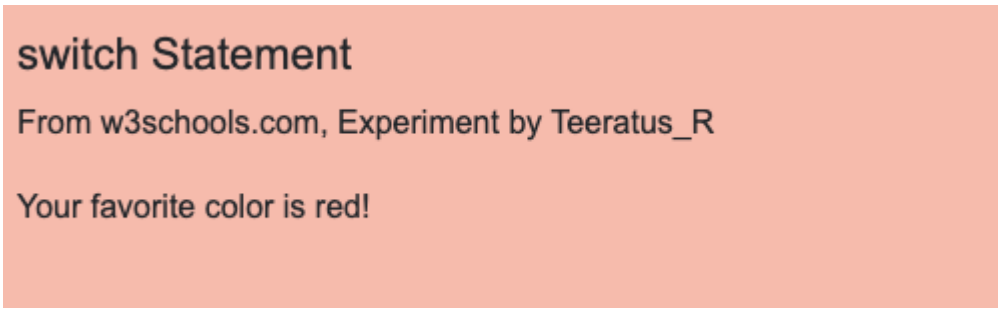
PHP switch Statement

Back to ACP page

Table of Contents

PHP 1 switch Statement

```
<h4>switch Statement</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?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!";
}
?>
```



Example 1

Result View Example

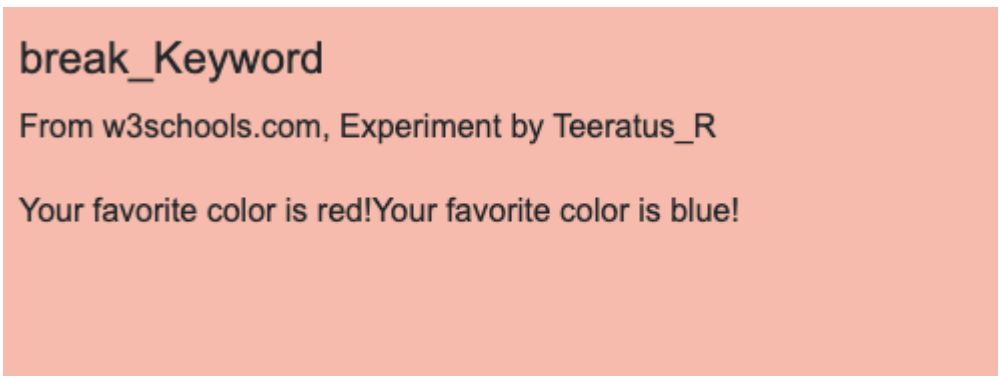
PHP 2 break Keyword

When PHP reaches a break keyword, it breaks out of the switch block.

This will stop the execution of more code, and no more cases are tested.

The last block does not need a break, the block breaks (ends) there anyway.

```
<h4>break_Keyword</h4>
<p>From w3schools.com, Experiment by Teeratus_R </p>
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
    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!";
}
?>
```



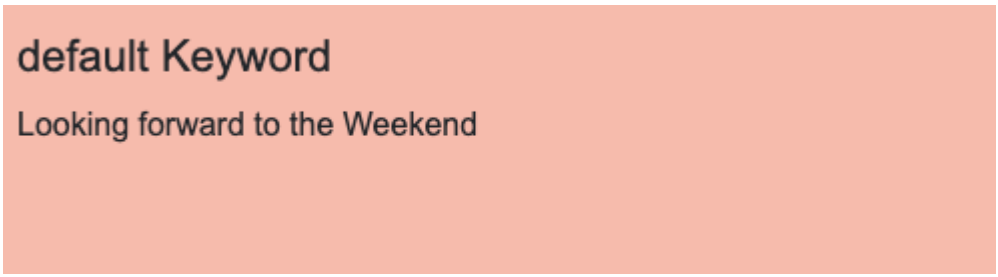
Example 2

Result View Example

PHP 3 default Keyword

The default keyword specifies the code to run if there is no case match:

```
<h4>default Keyword</h4>
<?php
$d = 4;

switch ($d) {
    case 6:
        echo "Today is Saturday";
        break;
    case 0:
        echo "Today is Sunday";
        break;
    default:
        echo "Looking forward to the Weekend";
}
?>
```



Example 3

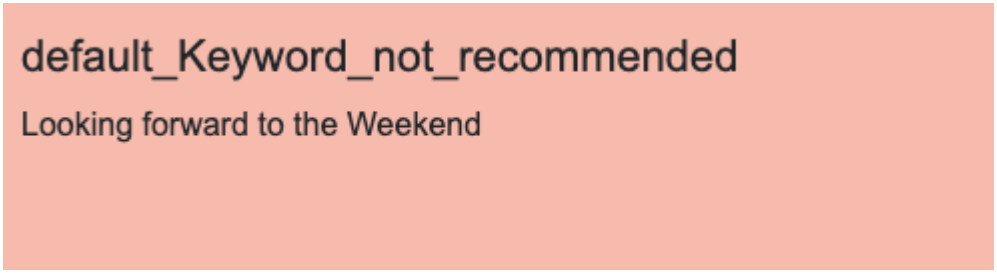
Result View Example

PHP 4 default Keyword - not recommended

Putting the default block elsewhere than at the end of the switch block is allowed, but not recommended.

```
<h4>default_Keyword_not_recommended</h4>
<?php
$d = 4;

switch ($d) {
    default:
        echo "Looking forward to the Weekend";
        break;
    case 6:
        echo "Today is Saturday";
```

```
        break;
    case 0:
        echo "Today is Sunday";
}
?>
```



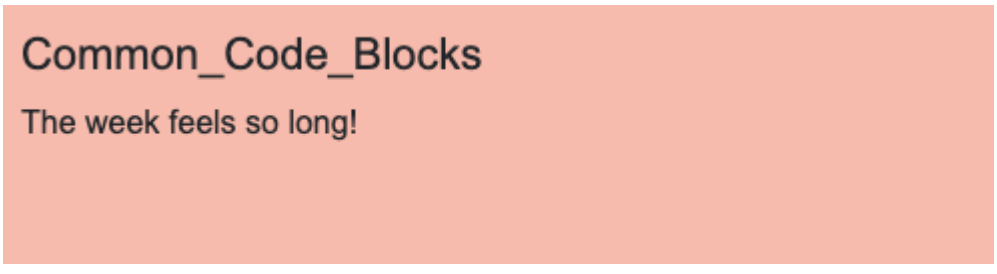
Example 4

Result View Example

PHP 5 Common Code Blocks

If you want multiple cases to use the same code block, you can specify the cases like this:

```
<h4>Common_Code_Blocks</h4>
<?php
$d = 3;

switch ($d) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        echo "The week feels so long!";
        break;
    case 6:
    case 0:
        echo "Weekends are the best!";
        break;
    default:
        echo "Something went wrong";
}
?>
```



Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

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 The following chapters will explain and give examples of each loop type.

Document

Document in project

You can Download PDF file.

Refference

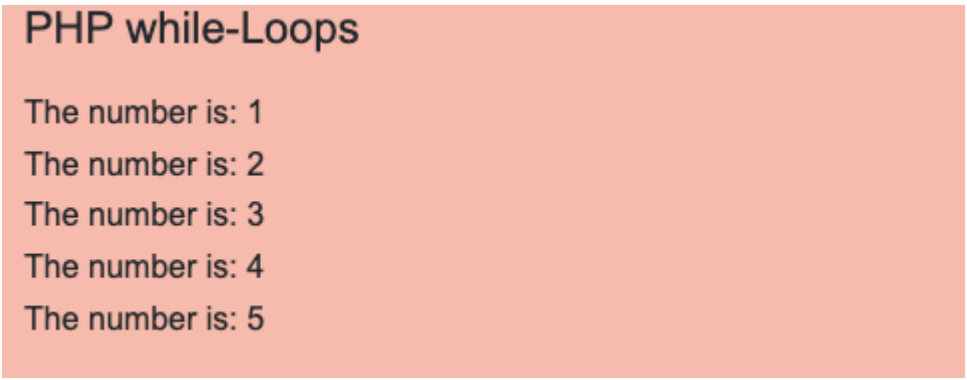
PHP while loops

Table of Contents

PHP while loops

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

```
<h4>PHP while-Loops </h4>
<p></p>
<?php
// while loop
    $x = 1;
    while($x <= 5) {
        echo "The number is: $x <br>";
        $x++;
    }
?>
```



Example 1

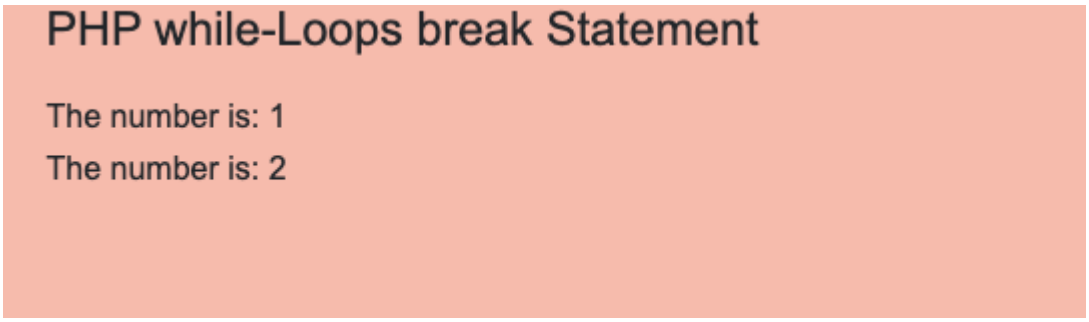
Result View Example

PHP break statement

With the break statement we can stop the loop even if the condition is still true:

```
<h4>PHP while-Loops break Statement</h4>
<p>With the break statement we can stop the loop even if the condition is
still true:</p>
<?php
// while loop break Statement
$i = 1;
while ($i <= 6) {
    if ($i == 3) {
        break;
    }
    echo "The number is: $i <br>";
    $i++;
}
```

?>



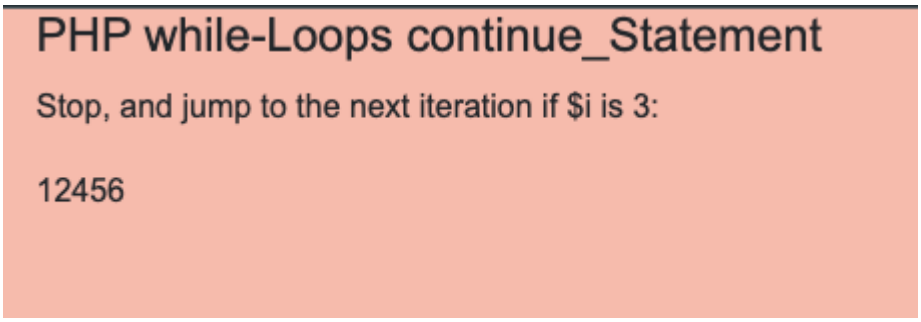
Example 2

Result View Example

PHP continue_Statement

With the continue statement we can stop the current iteration, and continue with the next:

```
<h4>PHP while-Loops continue_Statement</h4>
<p>Stop, and jump to the next iteration if $i is 3:</p>
<?php
// while loop continue_Statement
$i = 0;
while ($i < 6) {
  $i++;
  if ($i == 3) continue;
  echo $i;
}
?>
```

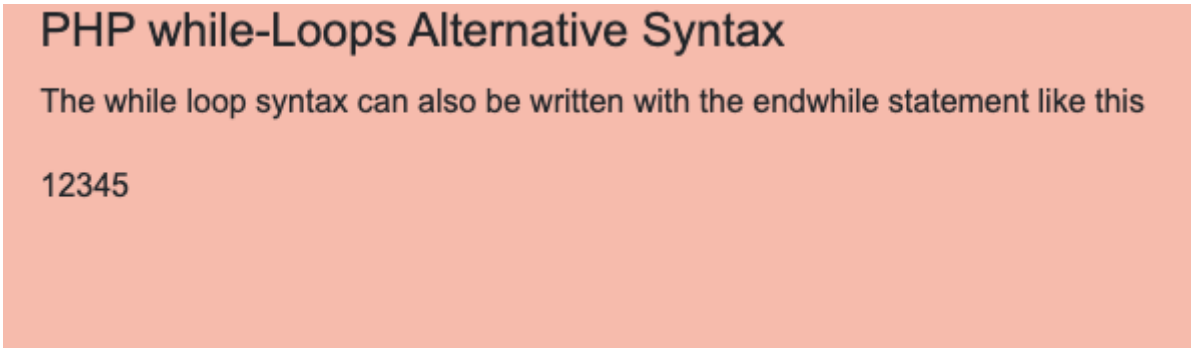


Example 3

Result View Example

PHP Alternative Syntax

```
<h4>PHP while-Loops Alternative Syntax</h4>
<p>The while loop syntax can also be written with the endwhile statement
like this</p>
<?php
// while loop Alternative Syntax
$i = 1;
while ($i < 6):
echo $i;
$i++;
endwhile;
?>
```



Example 4

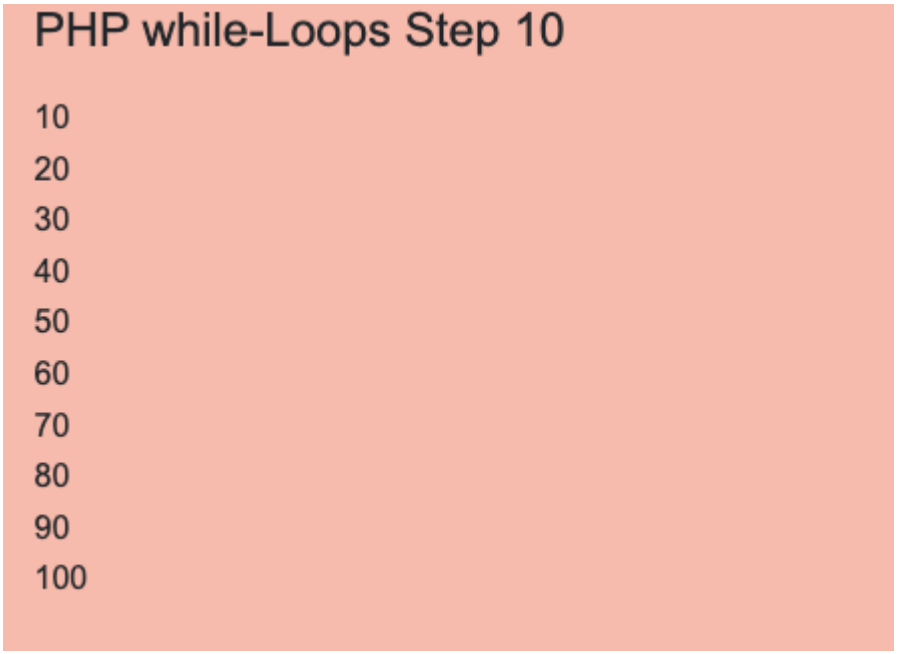
Result View Example

PHP Step 10

If you want the while loop count to 100, but only by each 10, you can increase the counter by 10 instead 1 in each iteration:

```
// while loop step 10
$i = 0;
```

```php
while ($i < 100) {
$i+=10;
echo "$i<br>";
}
```



Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

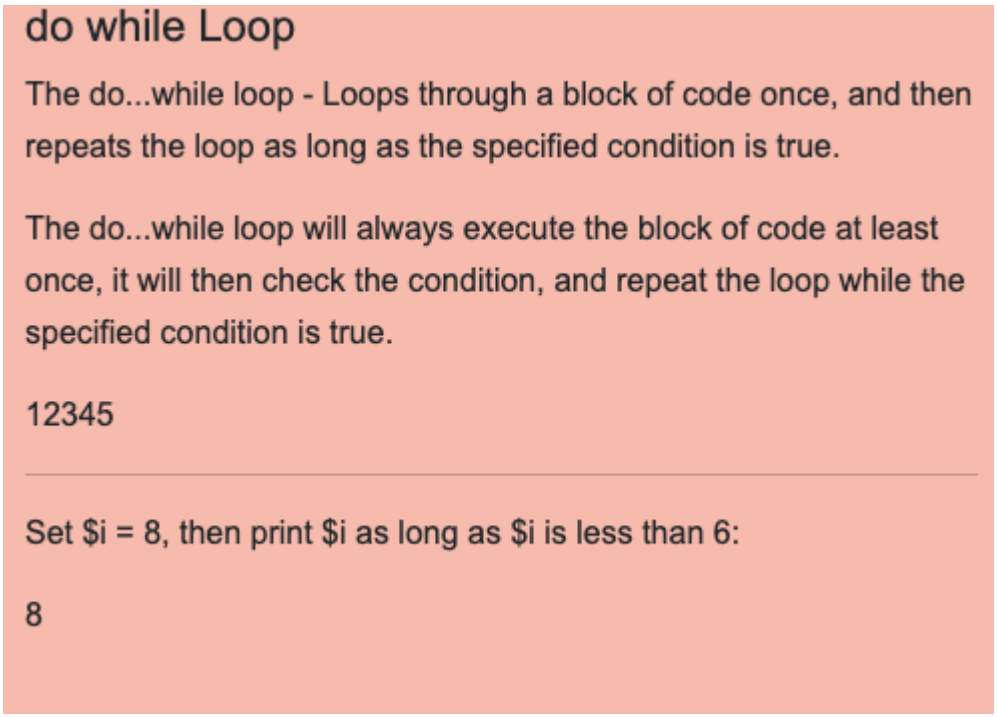
PHP do while Loop

Table of Contents

PHP do while Loop

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

```
<h4>do while Loop</h4>
<p>The do...while loop – Loops through a block of code once, and then
repeats the loop as long as the specified condition is true.</p>
<p>The do...while loop will always execute the block of code at least
once, it will then check the condition, and repeat the loop while the
specified condition is true.</p>
<?php
    $i = 1;

    do {
      echo $i;
      $i++;
    } while ($i < 6);
?>
<hr>
<p>Set $i = 8, then print $i as long as $i is less than 6:</p>
<?php
$i = 8;

do {
    echo $i;
    $i++;
} while ($i < 6);
?>
```



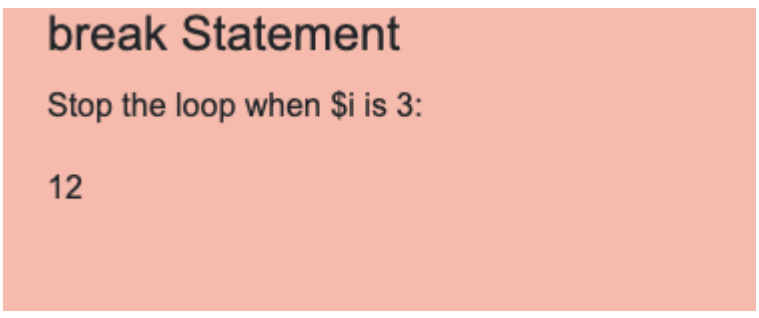
Example 1

Result View Example

PHP break Statement

With the break statement we can stop the loop even if the condition is still true:

```
<h4>break Statement</h4>
<p>Stop the loop when $i is 3:</p>
<?php
    $i = 1;

    do {
      if ($i == 3) break;
      echo $i;
      $i++;
    } while ($i < 6);
?>
```



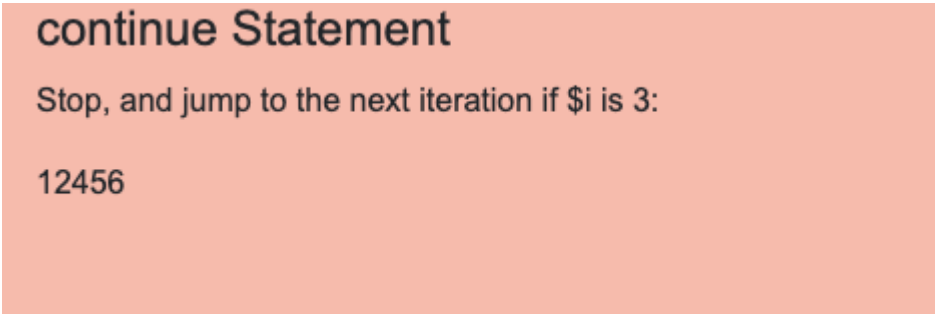
Example 2

Result View Example

PHP continue Statement

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
<h4>continue Statement</h4>
<p>Stop, and jump to the next iteration if $i is 3:</p>
<?php
    $i = 0;

    do {
      $i++;
      if ($i == 3) continue;
      echo $i;
    } while ($i < 6);
?>
```



Example 3

Result View Example

Document

Document in project

You can Download PDF file.

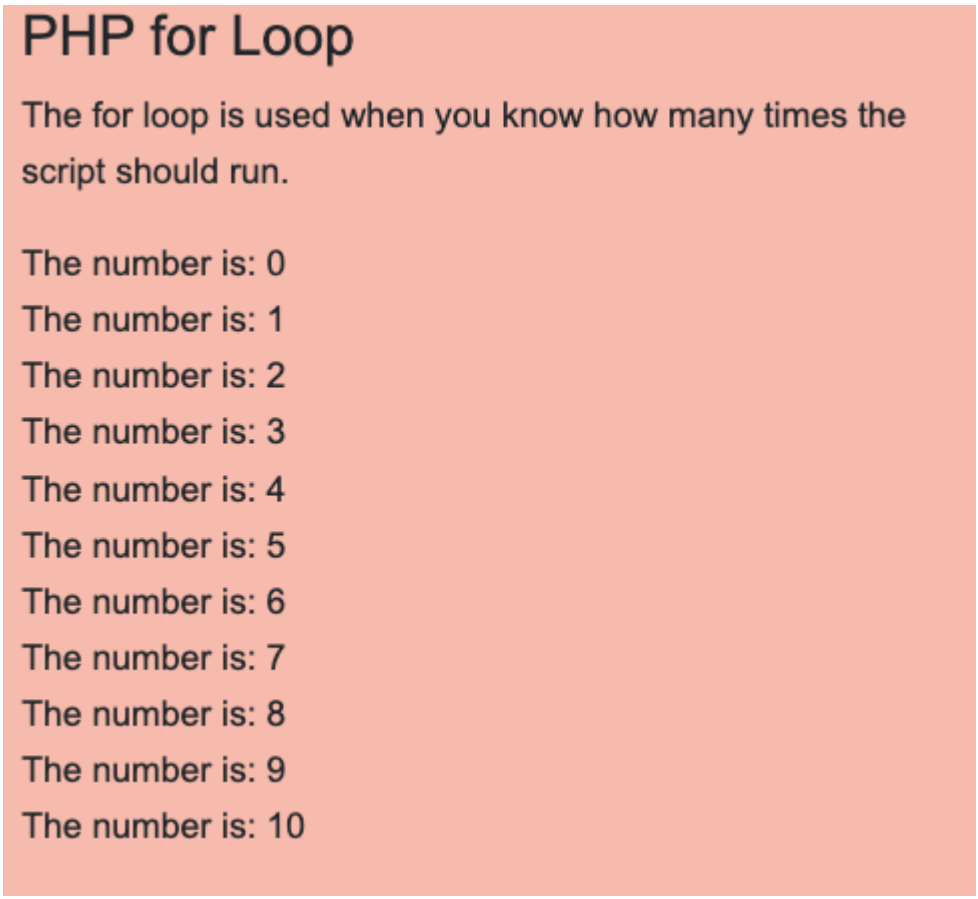
Reference

PHP for_Loop

Table of Contents

PHP for_Loop

```
<h2>PHP for Loop</h1>
<p>The for loop is used when you know how many times the script should
run.</p>

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

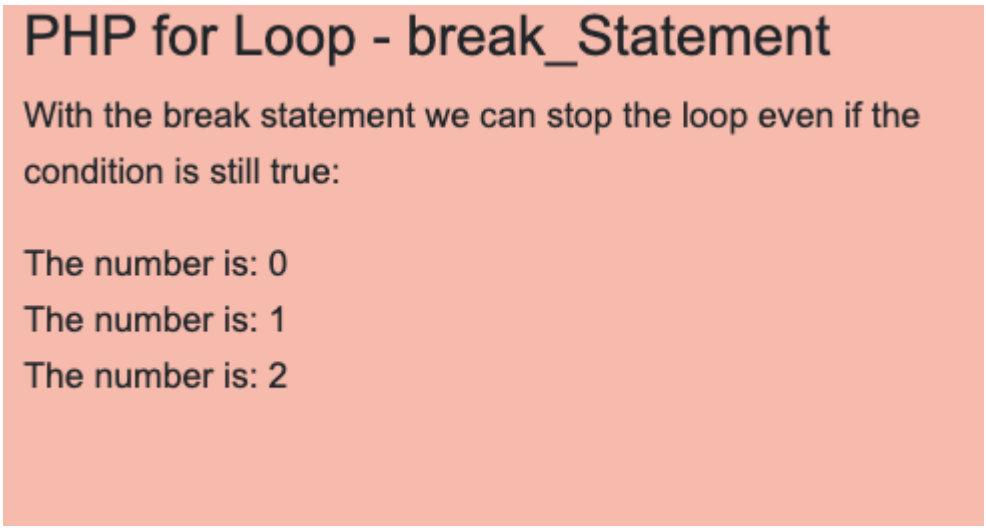


Example 1

Result View Example

PHP 2 break_Statement

```
<h2>PHP for Loop - break_Statement</h1>
<p>With the break statement we can stop the loop even if the condition is
still true:</p>

<?php
    for ($x = 0; $x <= 10; $x++) {
        if ($x == 3) break;
        echo "The number is: $x <br>";
    }
?>
```

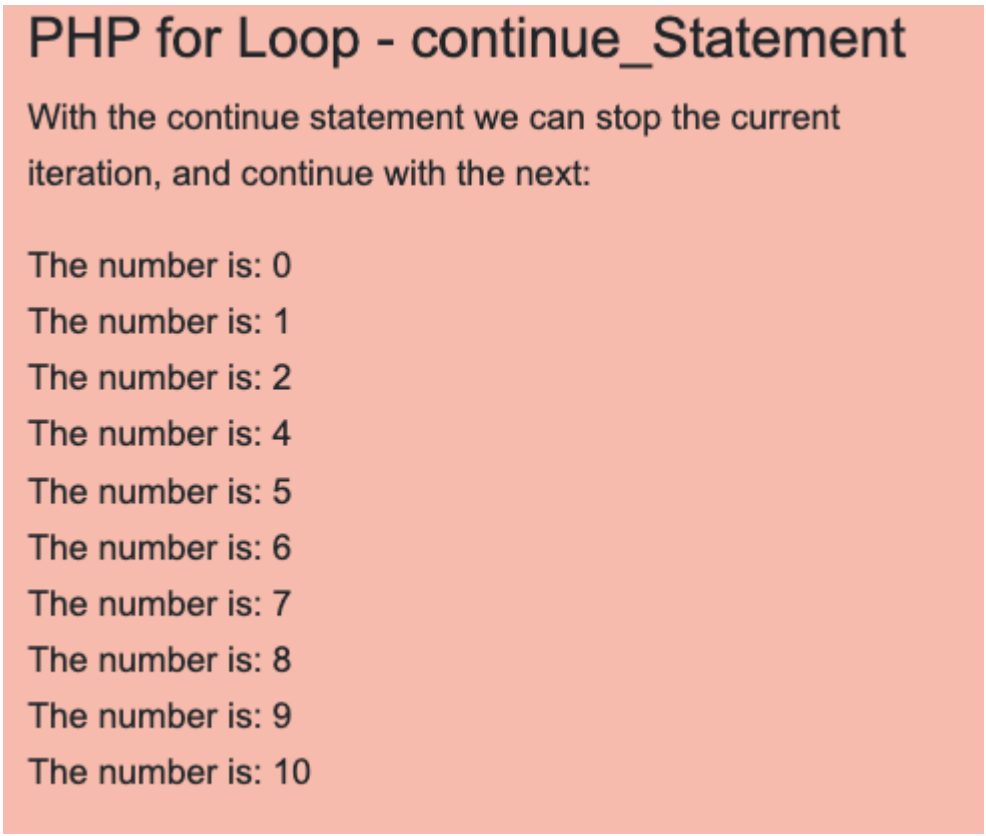


Example 2

Result View Example

PHP 3 continue_Statement

```
<h2>PHP for Loop – continue_Statement</h1>
<p>With the continue statement we can stop the current iteration, and
continue with the next:</p>

<?php
    for ($x = 0; $x <= 10; $x++) {
        if ($x == 3) continue;
        echo "The number is: $x <br>";
    }
?>
```

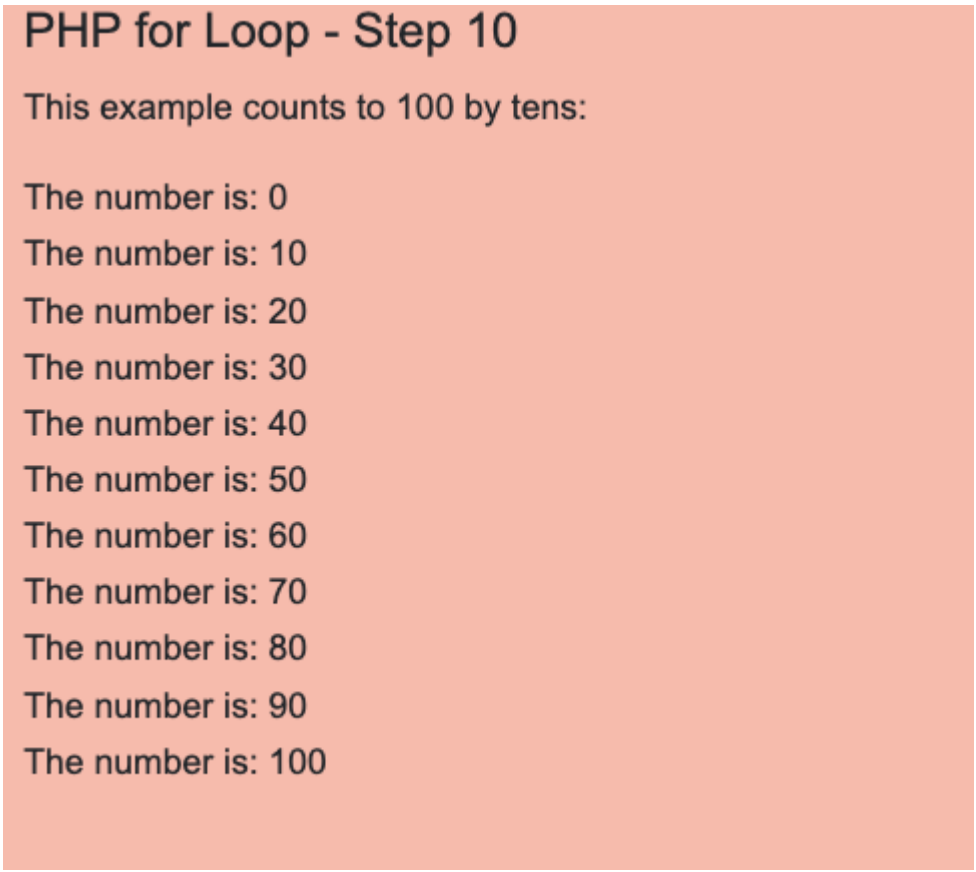


Example 3

Result View Example

PHP 4 Step_10

```
<h4>PHP for Loop - Step 10</h4>
<p>This example counts to 100 by tens:</p>

<?php
    for ($x = 0; $x <= 100; $x+=10) {
    echo "The number is: $x <br>";
}
?>
```



Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

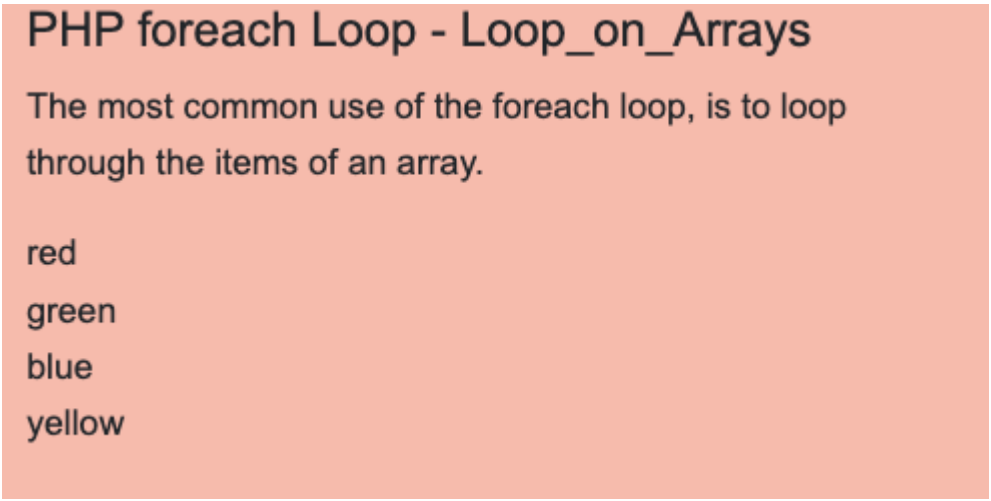
php-foreach_Loop

Table of Contents

Loop_on_Arrays

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

```
  <h4>PHP foreach Loop - Loop_on_Arrays</h4>
  <p>The most common use of the foreach loop, is to loop through the items
of an array.</p>

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

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

  ?>
```



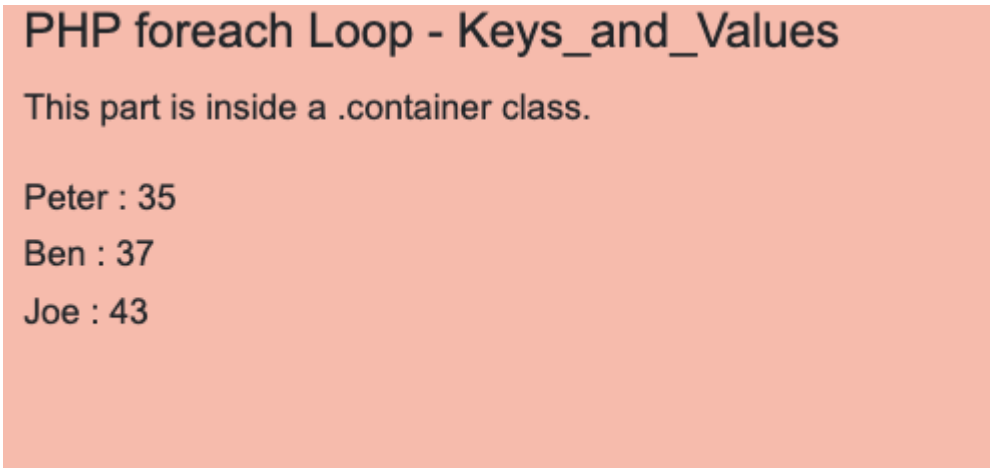
Example 1

Result View Example

Keys_and_Values

Print both the key and the value from the $members array:

```
<h4>PHP foreach Loop - Keys_and_Values</h4>
<p>Print both the key and the value from the $members array:</p>

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

    foreach($members as $x => $x_value) {
      echo "Key=" . $x . ", Value=" . $x_value;
      echo "<br>";
    }

?>
```



Example 2

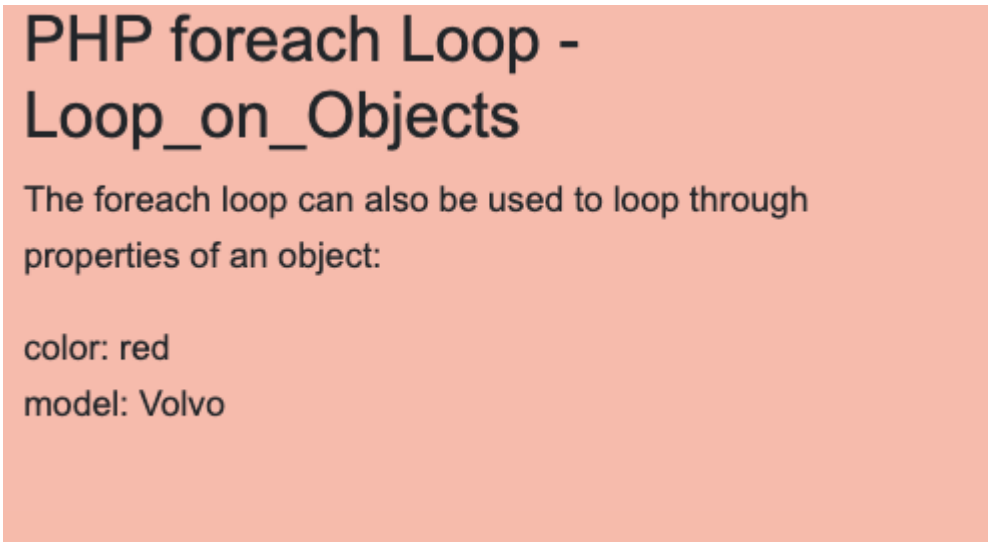
Result View Example

foreach Loop on Objects

The foreach loop can also be used to loop through properties of an object:

```
  <h4>PHP foreach Loop - foreach Loop on Objects</h4>
  <p>The foreach loop can also be used to loop through properties of an
object:</p>

  <?php
      class Car {
          public $color;
          public $model;
          public function __construct($color, $model) {
            $this->color = $color;
            $this->model = $model;
          }
```

```
        }

        $myCar = new Car("red", "Volvo");

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

  ?>
```



Example 3

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Functions

Back to ACP page

Table of Contents

PHP Built-in Functions

PHP 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.

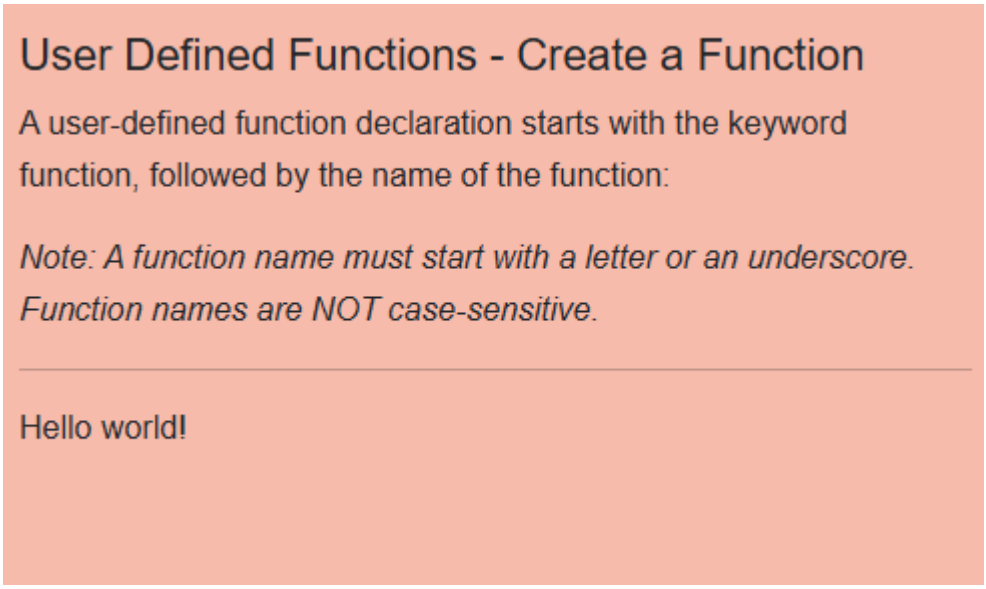
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

PHP 1 Create a Function - Call a Function

A user-defined function declaration starts with the keyword function, followed by the name of the function:

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

```
<h4>User Defined Functions - Create a Function</h4>
<p>A user-defined function declaration starts with the keyword function,
followed by the name of the function:</p>
<i>Note: A function name must start with a letter or an underscore.
Function names are NOT case-sensitive.</i>
<hr>
<?php
    // Call the function
    _myMessage();


    // Create a function
    function _myMessage() {
        echo "Hello world!";
    }
?>
```



Example 1

Result View Example

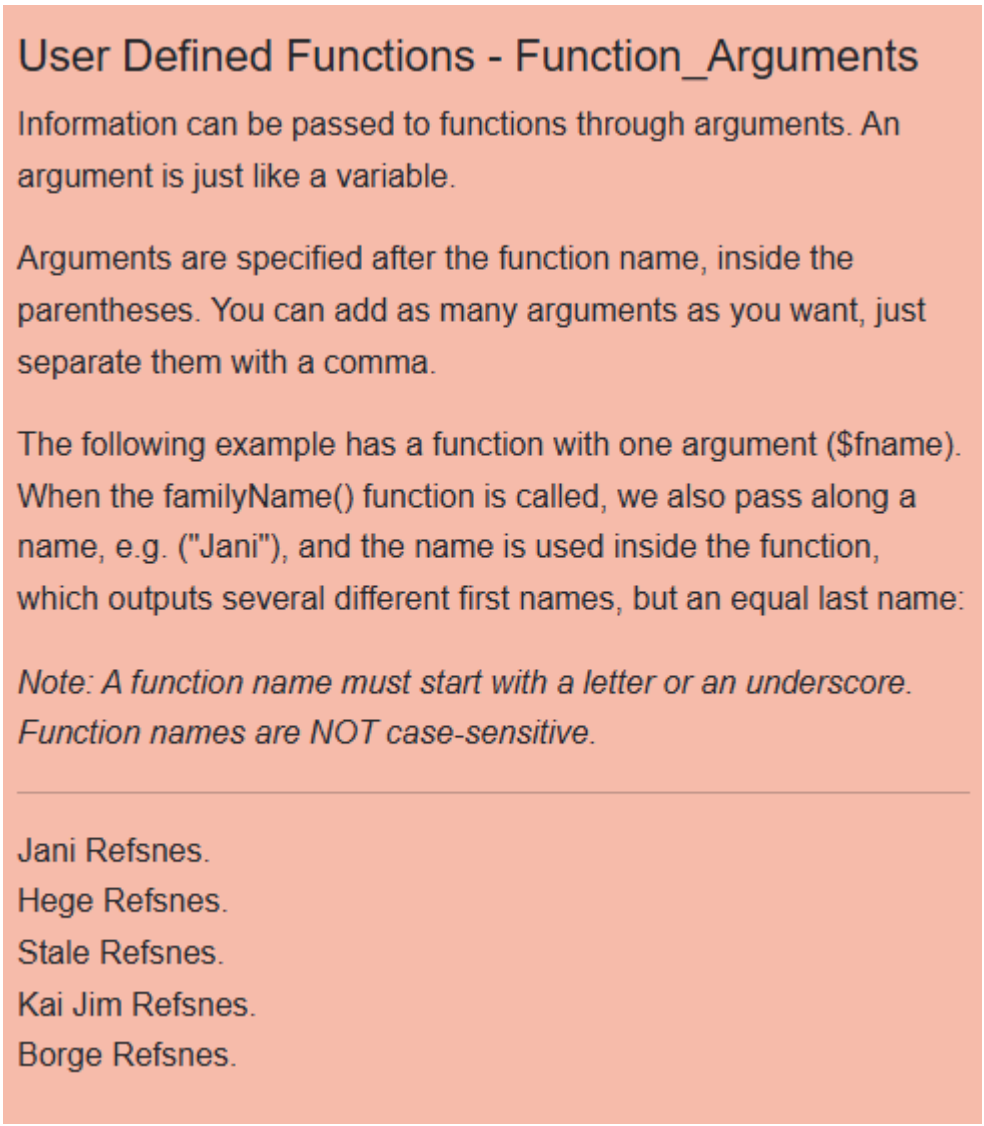
PHP 2 Function Arguments

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.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name, e.g. ("Jani"), and the name is used inside the function, which outputs several different first names, but an equal last name:

```
<h4>User Defined Functions - Function_Arguments</h4>
<i>Note: A function name must start with a letter or an underscore.
Function names are NOT case-sensitive.</i>
<hr>

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

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



User Defined Functions - 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.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name, e.g. ("Jani"), and the name is used inside the function, which outputs several different first names, but an equal last name:

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

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

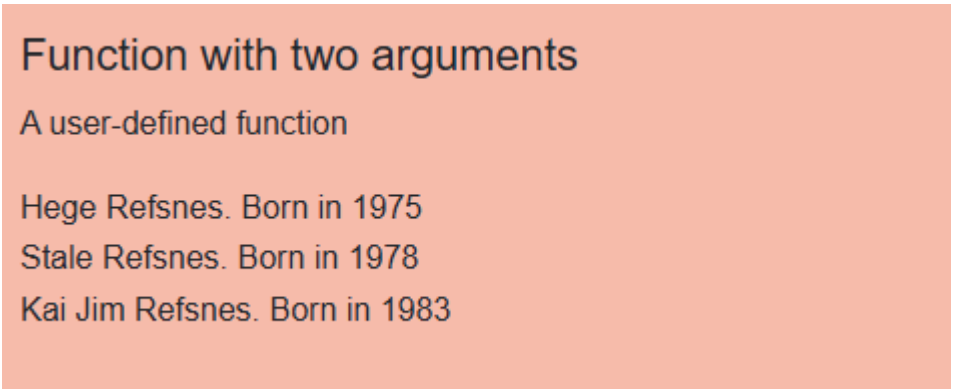
Example 2

Result View Example

PHP 3 function with two arguments

The following example has a function with two arguments ($fname, $year):

```
<h4>Function with two arguments</h4>
<p>A user-defined function</p>
<?php
function familyName($fname, $year)
{
    echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

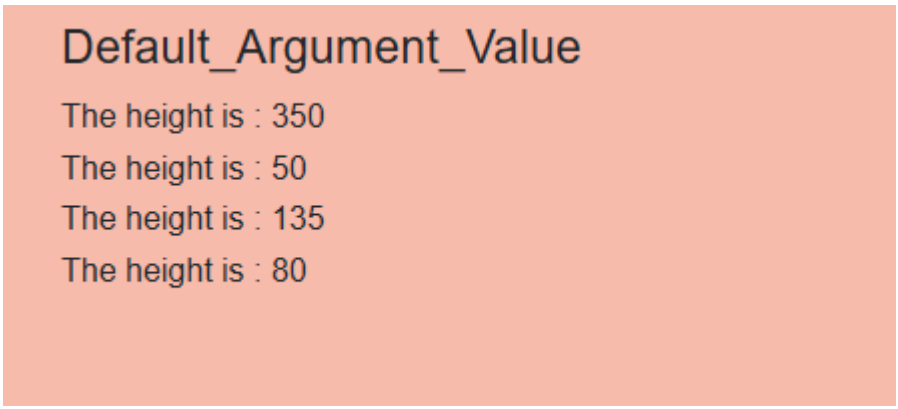


Example 3

Result View Example

PHP 4 Default Argument Value

```
<h4>Default_Argument_Value</h4>
<?php
function setHeight($minheight = 50)
{
    echo "The height is : $minheight <br>";
}

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



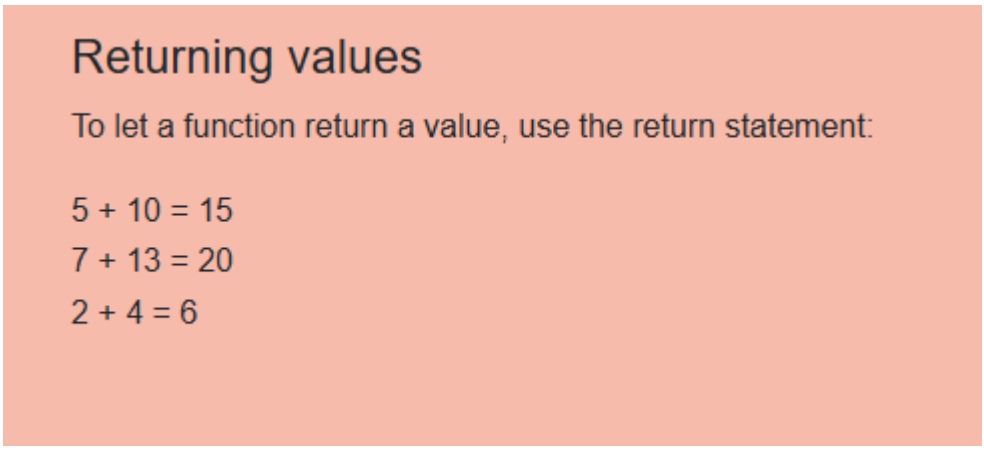
Example 4

Result View Example

PHP 5 Functions - Returning values

To let a function return a value, use the return statement:

```
<h4>Returning values</h4>
<p>To let a function return a value, use the return statement:</p>
<?php
function sum($x, $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);
?>
```



Example 5

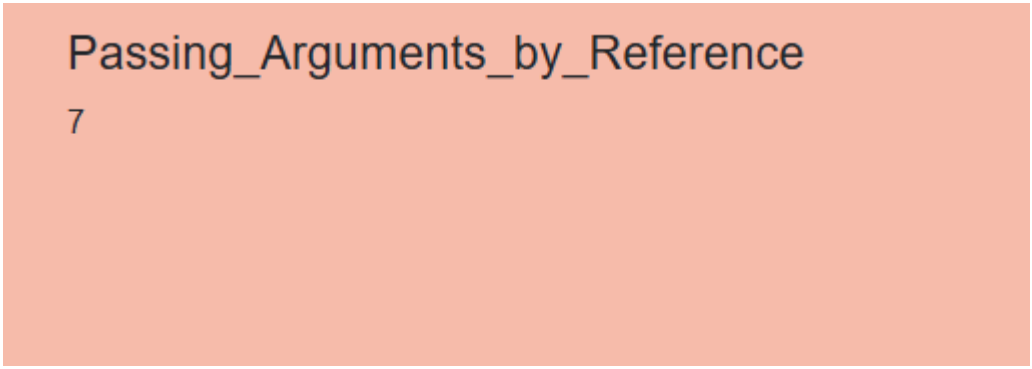
Result View Example

PHP 6 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:

```
<h4>Passing_Arguments_by_Reference</h4>
<?php
function add_five(&$value)
{
    $value += 5;
}

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



Example 6

Result View Example

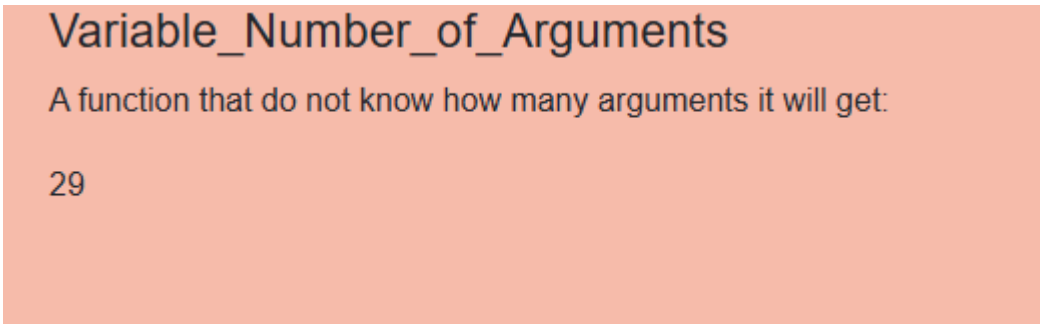
PHP 7 Variable Number of Arguments

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.

```
<h4>Variable_Number_of_Arguments</h4>
<p>A function that do not know how many arguments it will get:</p>
<?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;
?>
```

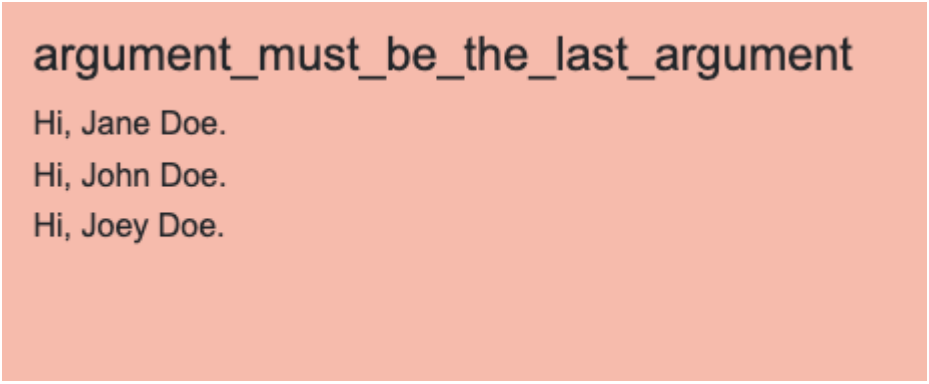


Example 7

Result View Example

PHP 8 argument must be the last argument

```
<h4>argument_must_be_the_last_argument</h4>

<?php
    function myFamily($lastname, ...$firstname) {
    // function myFamily(...$firstname, $lastname) { // Error
        $txt = "";
        $len = count($firstname);
        for($i = 0; $i < $len; $i++) {
          $txt = $txt."Hi, $firstname[$i] $lastname.<br>";
        }
        return $txt;
      }

      $a = myFamily("Doe", "Jane", "John", "Joey");
      echo $a;
?>
```



Example 8

Result View Example

PHP 9 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
declare(strict_types=1); // strict requirement


/**
 * Move the declare(strict_types=1);
 * statement to the very top of the PHP file, before any other code.
 */
?>

<?php

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
?>
```

Loosely_Typed_Language

Note: 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:

10

Example 9

Result View Example

PHP 10 Return Type Declarations

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


/**
 * Move the declare(strict_types=1);
 * statement to the very top of the PHP file, before any other code.
 */
?>

<h4>Return_Type_Declarations</h4>

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

Return_Type_Declarations

6

Example 10

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Arrays

Back to ACP page

Table of Contents

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

Working With Arrays

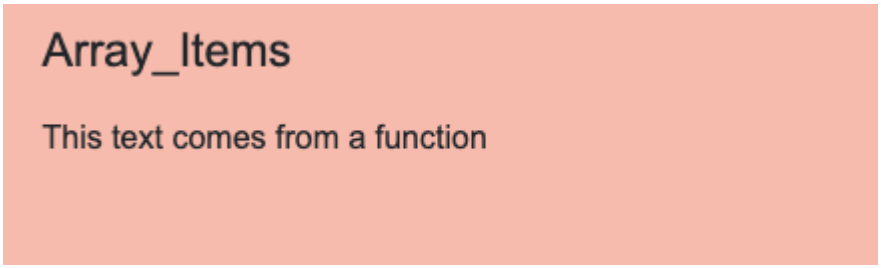
In this tutorial you will learn how to work with arrays, including:

- Create Arrays
- Access Arrays
- Update Arrays
- Add Array Items
- Remove Array Items
- Sort Arrays

PHP 1 Array Items

```
<h4>Array_Items</h4>
<p></p>
<?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]();
?>
```

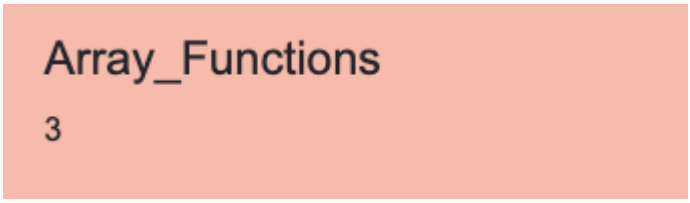


Example 1

Result View Example

PHP 2 Array Functions

```
<h4>Array_Functions</h4>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```



Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

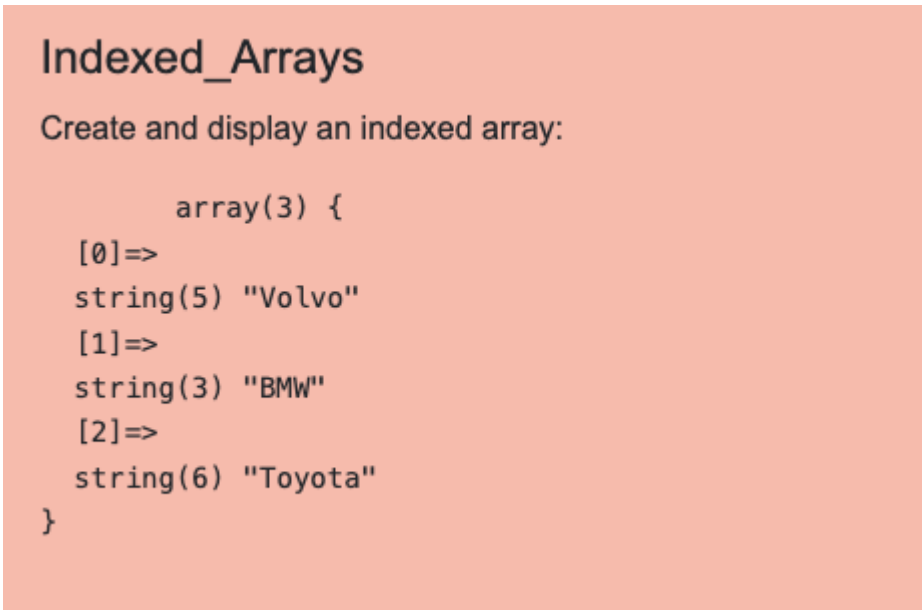
PHP Indexed Arrays

Back to ACP page

Table of Contents

PHP 1 Indexed Arrays

```
<h4>Indexed_Arrays</h4>
<p>Create and display an indexed array:</p>
<pre>
<?php
$cars = array("Volvo", "BMW", "Toyota");
var_dump($cars);
?>
</pre>
```



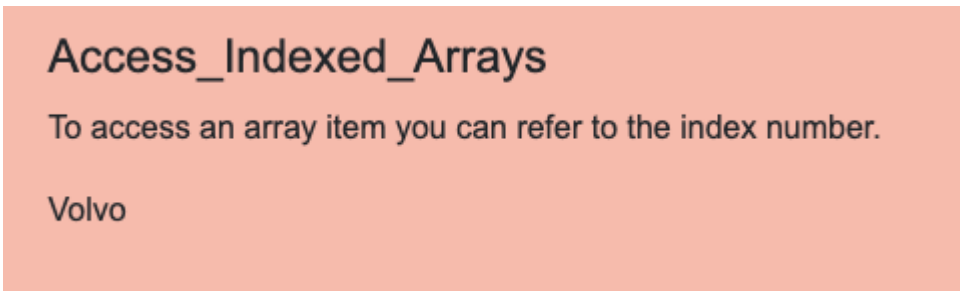
Example 1

Result View Example

PHP 2 Access Indexed Arrays

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

```
<h4>Access_Indexed_Arrays</h4>
<p>To access an array item you can refer to the index number.</p>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo $cars[0];
?>
```



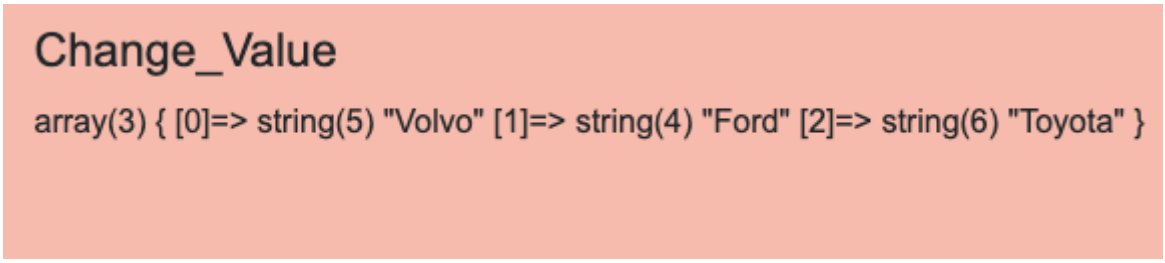
Example 2

Result View Example

PHP 3 Change Value

To change the value of an array item, use the index number:

```
<h4>Change_Value</h4>
<?php
$cars = array("Volvo", "BMW", "Toyota");
$cars[1] = "Ford";
var_dump($cars);
?>
```



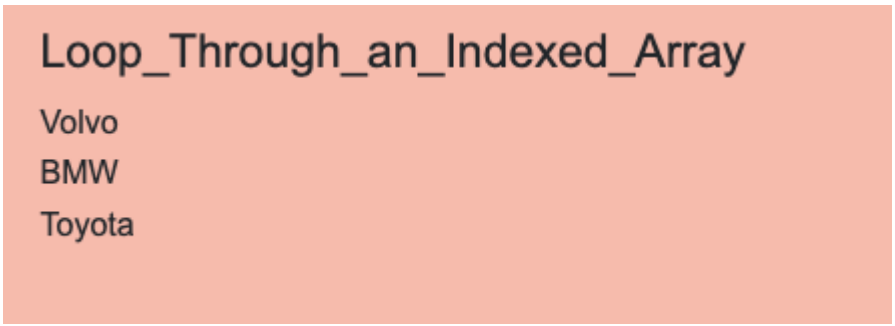
Example 3

Result View Example

PHP 4 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:

```
<h4>Loop_Through_an_Indexed_Array</h4>
<?php
$cars = array("Volvo", "BMW", "Toyota");
foreach ($cars as $x) {
    echo "$x <br>";
}
?>
```



Example 4

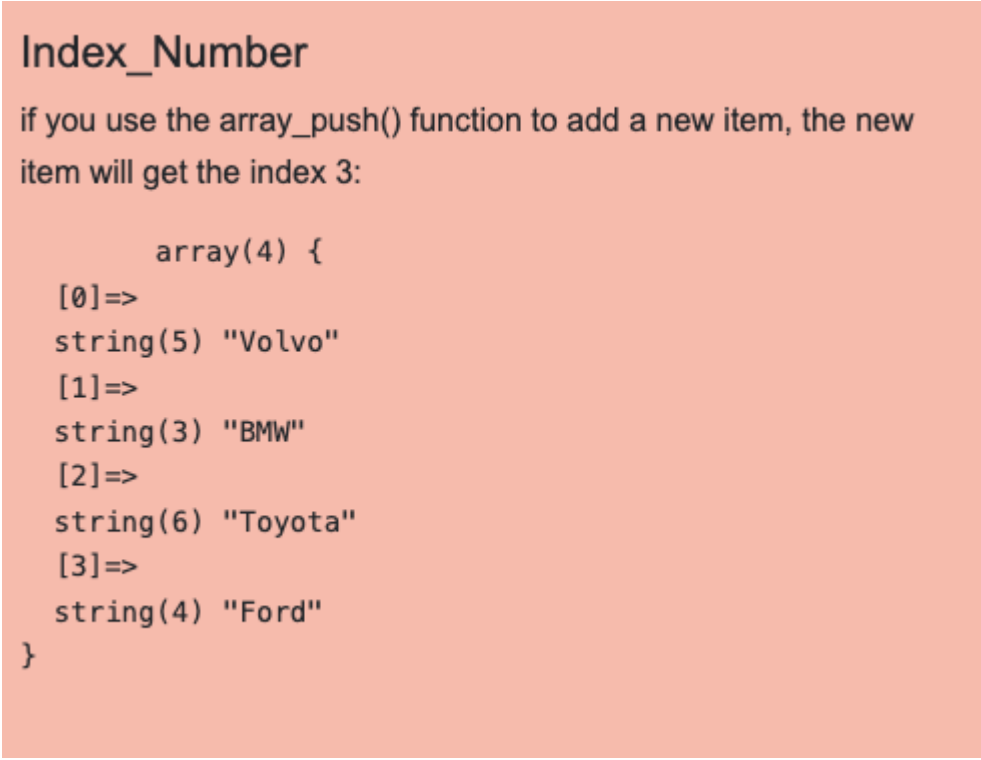
Result View Example

PHP 5 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.

```
<h4>Index_Number</h4>
<p>if you use the array_push() function to add a new item, the new item
will get the index 3:</p>
<pre>
<?php
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";

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



Example 5

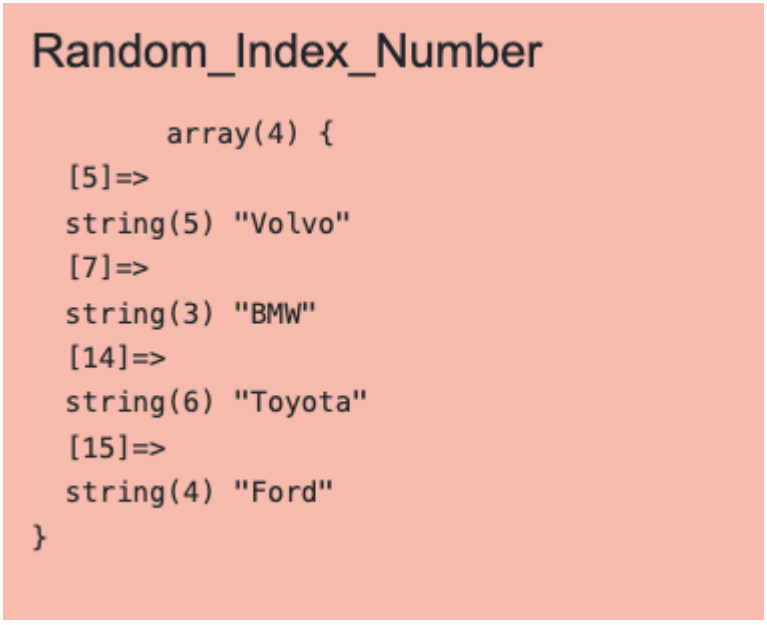
Result View Example

PHP 6 Random Index Number

If you assign a new value without an index number, PHP will automatically assign one for you.

```
<h4>Random_Index_Number</h4>
<pre>
<?php
$cars[5] = "Volvo";
$cars[7] = "BMW";
```

```
$cars[14] = "Toyota";

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



Example 6

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Associative Arrays

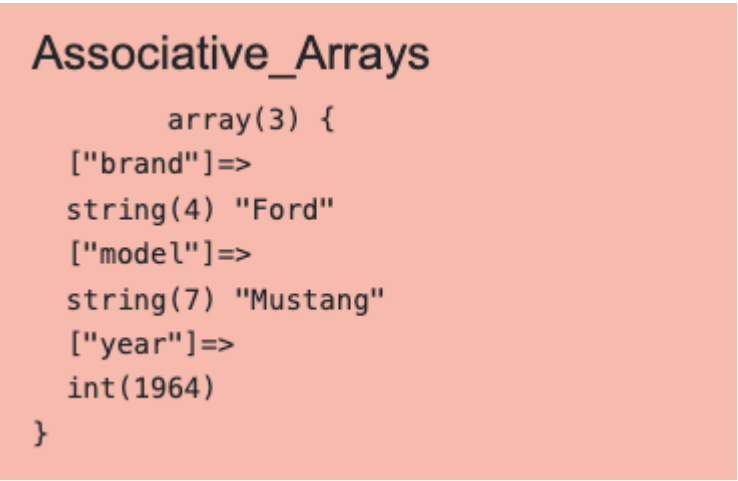
Back to ACP page

Table of Contents

PHP 1 Associative_Arrays

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

```
<h4>Associative_Arrays</h4>
<pre>
<?php
$car = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);
var_dump($car);
?>
</pre>
```



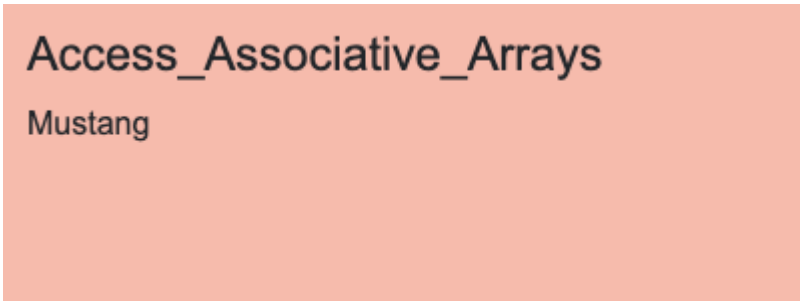
Example 1

Result View Example

PHP 2 Access Associative Arrays

You can access the values of the array using the key name.

```
<h4>Access_Associative_Arrays</h4>
<?php
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);
echo $car["model"];
?>
```



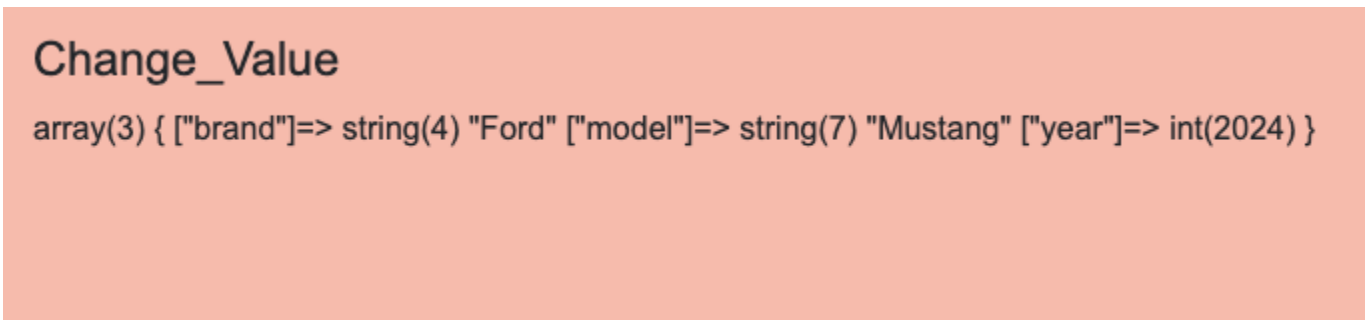
Example 2

Result View Example

PHP 3 Change Value

You can change the value of a specific element by referring to the key name.

```
<h4>Change_Value</h4>
<?php
$car = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);
$car["year"] = 2024;
var_dump($car);
?>
```



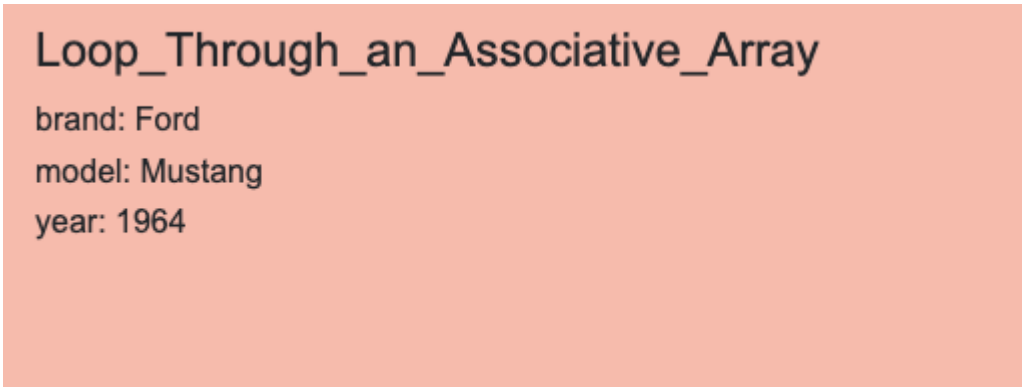
Example 3

Result View Example

PHP 4 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:

```
<h4>Loop_Through_an_Associative_Array</h4>
<?php
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);

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



Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

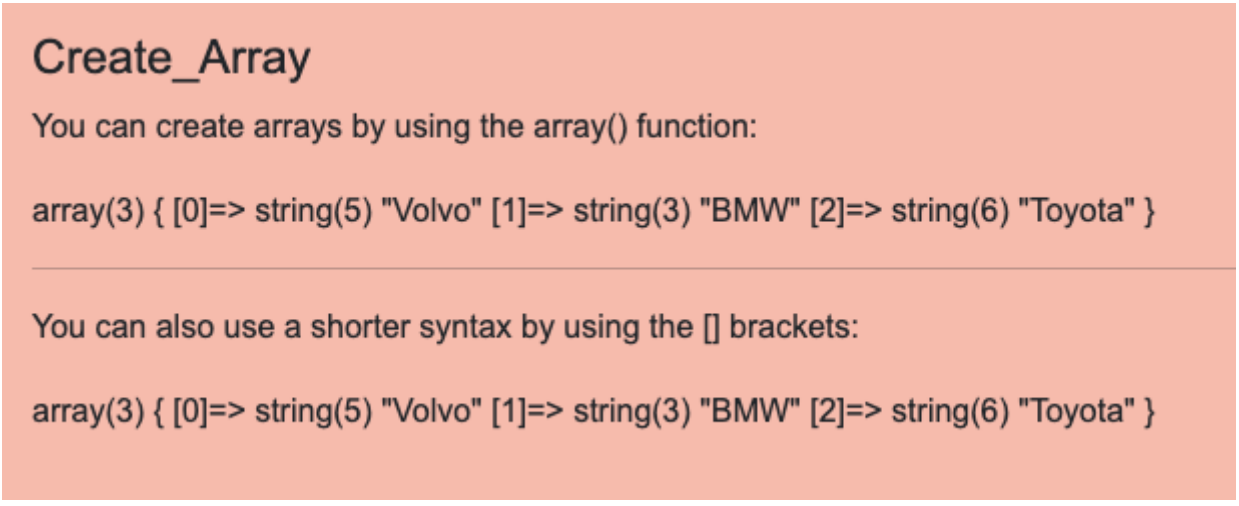
PHP Create Arrays

Back to ACP page

Table of Contents

PHP 1 Create Array

```
<h4>Create_Array</h4>
<p>You can create arrays by using the array() function:</p>
<?php
$cars1 = array("Volvo", "BMW", "Toyota");
var_dump($cars1);
?>
<hr>
<p>You can also use a shorter syntax by using the [] brackets:</p>
<?php
$cars2 = ["Volvo", "BMW", "Toyota"];
var_dump($cars2);
?>
```



Example 1

Result View Example

PHP 2 Multiple Lines

```
<pre>

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

</pre>
```

Multiple_Lines

Line breaks are not important, so an array declaration can span multiple lines:

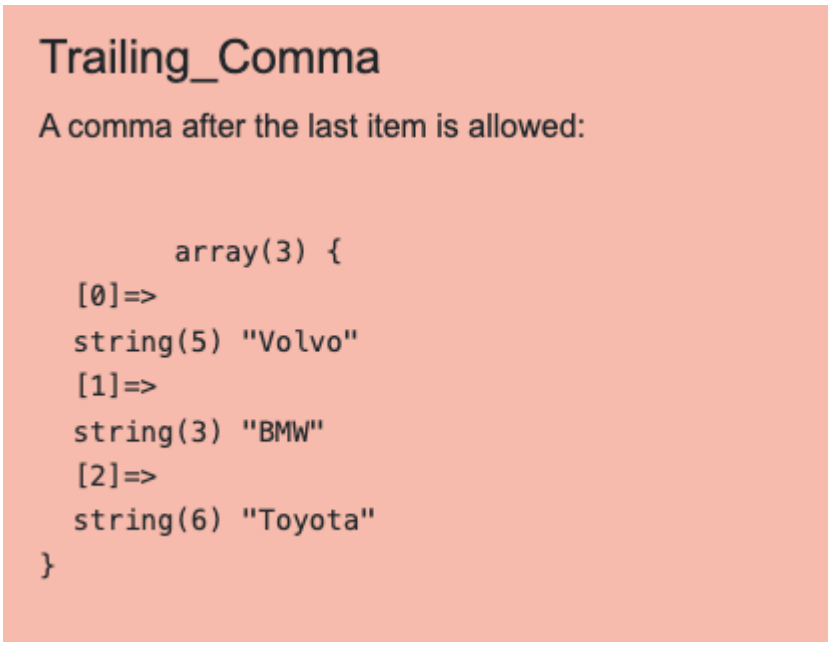
```
        array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
}
```

Example 2

Result View Example

PHP 3 Trailing_Comma

```
<h4>Trailing_Comma</h4>
<p>A comma after the last item is allowed:</p>
<pre>

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

</pre>
```



Example 3

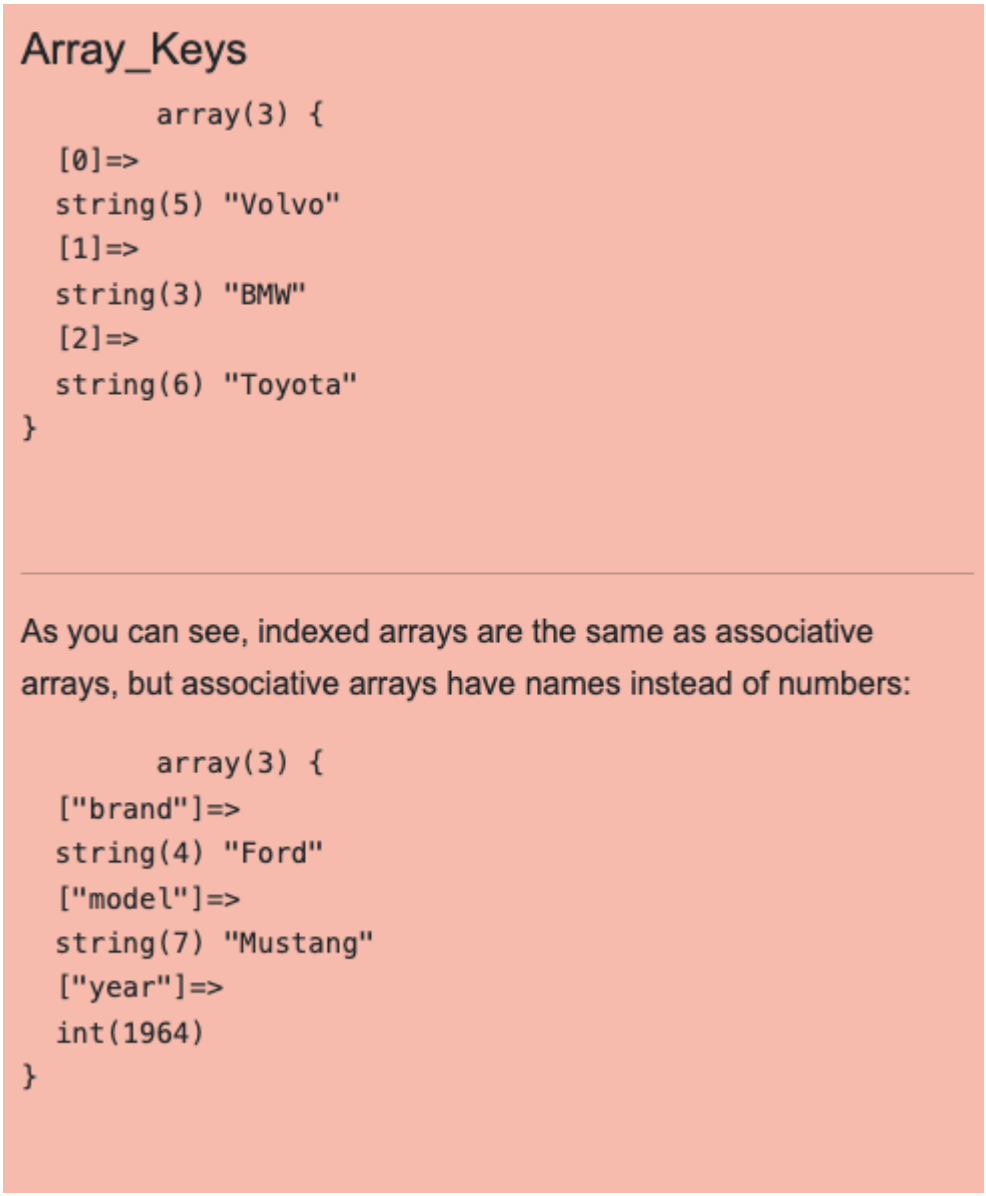
Result View Example

PHP 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:

```
<h4>Array_Keys</h4>
<pre>
<?php
$cars1 = [
0 => "Volvo",
1 => "BMW",
2 =>"Toyota"
];

var_dump($cars1);
?>
</pre>
<hr>
<p>As you can see, indexed arrays are the same as associative arrays, but
associative arrays have names instead of numbers:</p>
<pre>
<?php
$myCar2 = [
"brand" => "Ford",
"model" => "Mustang",
"year" => 1964
];
```

```
var_dump($myCar2);
?>
</pre>
```



Example 4

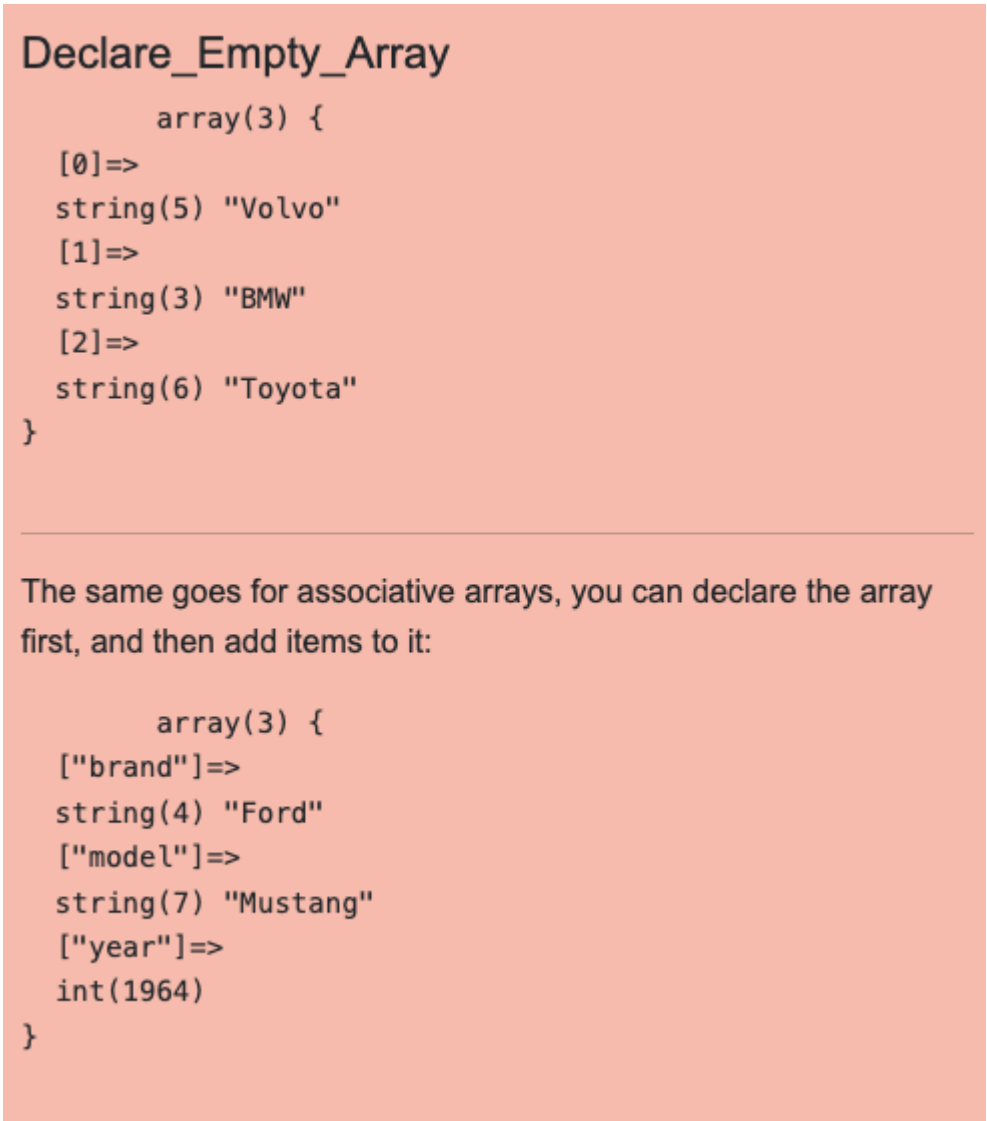
Result View Example

PHP 5 Declare Empty Array

You can declare an empty array first, and add items to it later:

```
<h4>Declare_Empty_Array</h4>
<pre>
<?php
$cars = [];
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

```
var_dump($cars);
?>
</pre>
<hr>
<p>The same goes for associative arrays, you can declare the array first,
and then add items to it:</p>
<pre>
<?php
$myCar = [];
$myCar["brand"] = "Ford";
$myCar["model"] = "Mustang";
$myCar["year"] = 1964;
var_dump($myCar);
?>
</pre>
```

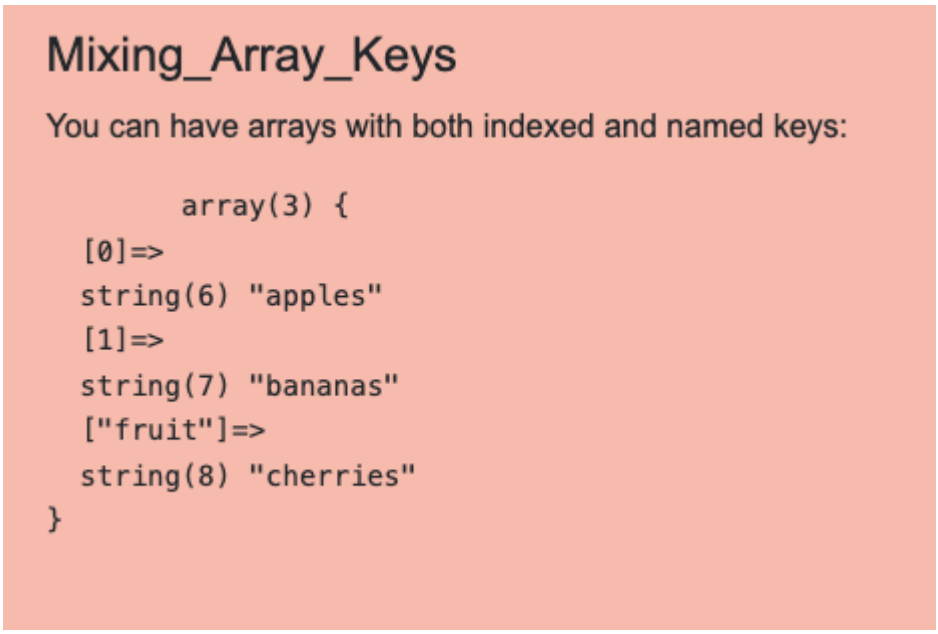


Example 5

Result View Example

PHP 6 Mixing Array Keys

```
<h4>Mixing_Array_Keys</h4>
<p>You can have arrays with both indexed and named keys:</p>
<pre>
<?php
$myArr = [];
$myArr[0] = "apples";
$myArr[1] = "bananas";
$myArr["fruit"] = "cherries";

var_dump($myArr);
?>
</pre>
```



Example 6

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Update Array Items

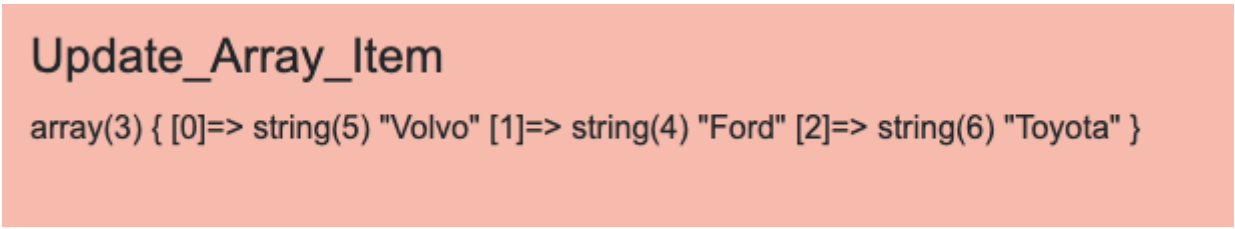
Back to ACP page

Table of Contents

PHP 1 Update Array Item

To update an existing array item, you can refer to the index number for indexed arrays, and the key name for associative arrays.

```
<h4>Update_Array_Item</h4>
<?php
$cars = array("Volvo", "BMW", "Toyota");
$cars[1] = "Ford";
var_dump($cars);
?>
```



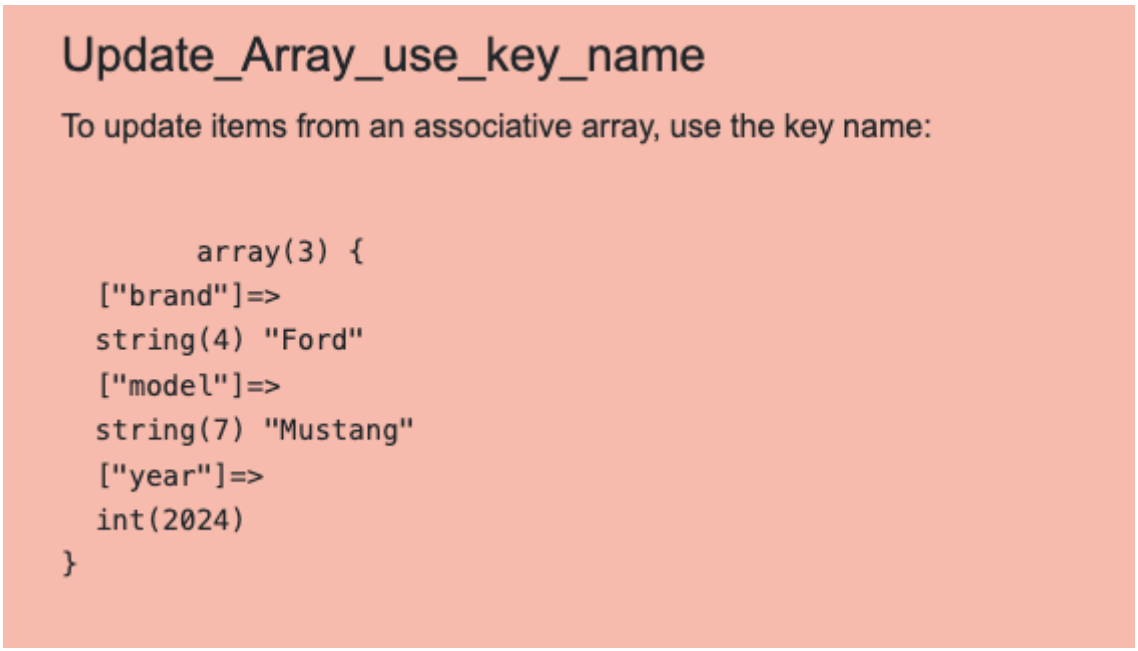
Example 1

Result View Example

PHP 2 Update Array use key name

```
<h4>Update_Array_use_key_name</h4>
<p>To update items from an associative array, use the key name:</p>
<pre>

<?php
```

```
$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);
$cars["year"] = 2024;
var_dump($cars);
?>

</pre>
```

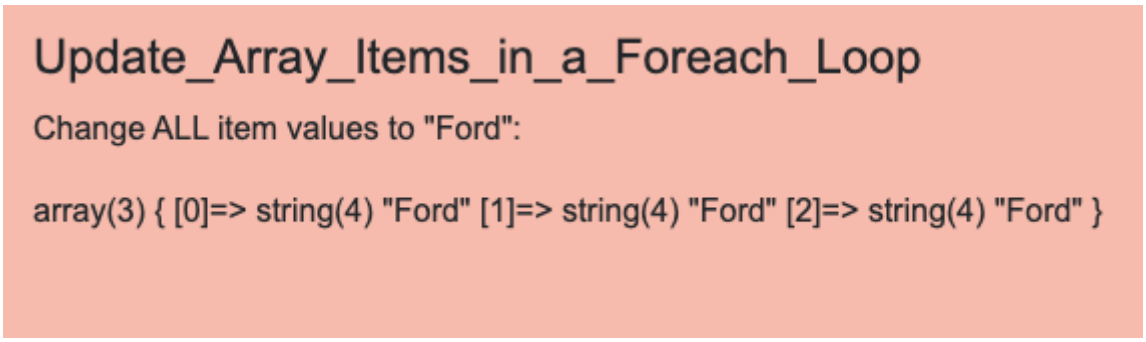


Example 2

Result View Example

PHP 3 Update Array Items in a Foreach Loop

```
<h4>Update_Array_Items_in_a_Foreach_Loop</h4>
<p>Change ALL item values to "Ford":</p>
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    foreach ($cars as &$x) {
    $x = "Ford";
    }
    unset($x);
    var_dump($cars);
?>
```



Example 3

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Multidimensional Arrays

Back to ACP page

Table of Contents

In the previous pages, we have described arrays that are a single list of key/value pairs.

However, sometimes you want to store values with more than one key. For this, we have multidimensional arrays.

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

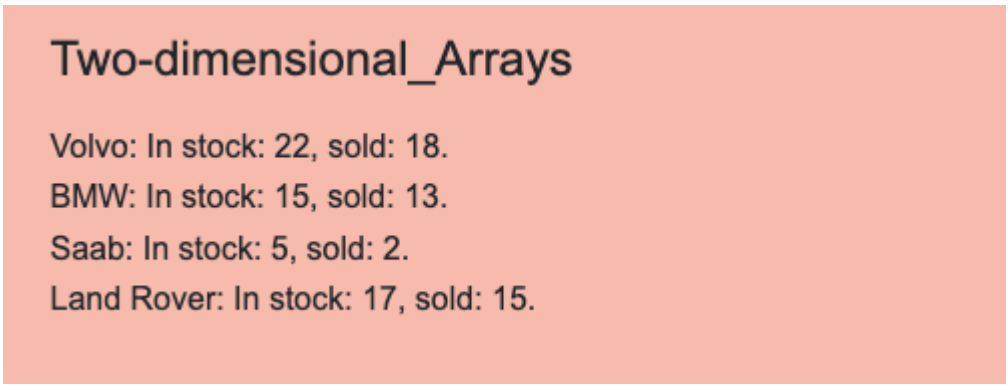
For a two-dimensional array you need two indices to select an element For a three-dimensional array you need three indices to select an element

PHP 1 Two-dimensional_Arrays

To get access to the elements of the $cars array we must point to the two indices (row and column):

```
<h4>Two-dimensional_Arrays</h4>
<p></p>
<?php
$cars = array(
    array("Volvo", 22, 18),
    array("BMW", 15, 13),
    array("Saab", 5, 2),
    array("Land Rover", 17, 15)
);

echo $cars[0][0] . ": In stock: " . $cars[0][1] . ", sold: " . $cars[0][2]
. ".<br>";
echo $cars[1][0] . ": In stock: " . $cars[1][1] . ", sold: " . $cars[1][2]
```

```
. ".<br>";
echo $cars[2][0] . ": In stock: " . $cars[2][1] . ", sold: " . $cars[2][2]
. ".<br>";
echo $cars[3][0] . ": In stock: " . $cars[3][1] . ", sold: " . $cars[3][2]
. ".<br>";
?>
```



Example 1

Result View Example

PHP 2 Two-dimensional_Arrays_for_Loops

```
<h4>Two-dimensional_Arrays_for_Loops</h4>
<p>We can also put a for loop inside another for loop to get the elements
of the $cars array (we still have to point to the two indices):</p>
<?php
$cars = array (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
  );

  for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
      echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
  }
?>
```

Two-dimensional_Arrays_for_Loops

We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):

Row number 0

- Volvo
- 22
- 18

Row number 1

- BMW
- 15
- 13

Row number 2

- Saab
- 5
- 2

Row number 3

- Land Rover
- 17
- 15

Example 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Array Functions

Back to ACP page

Table of Contents

PHP has a set of built-in functions that you can use on arrays.

Function	Description
`array()`	Create an array
`array_change_key_case()`	Changes all keys in an array to lowercase or uppercase
`array_chunk()`	Splits an array into chunks of arrays
`array_column()`	Returns the values from a single column in the input array
`array_combine()`	Combines two arrays into one array
`array_count_values()`	Counts all the values of an array
`array_diff()`	Compare arrays, and returns the differences (values)
`array_diff_assoc()`	Compare arrays, and returns the differences (values and keys)
`array_diff_key()`	Compare arrays, and returns the differences (keys)
`array_diff_uassoc()`	Compare arrays, and returns the differences (values and keys), using a user-defined key comparison function
`array_diff_ukey()`	Compare arrays, and returns the differences (keys), using a user-defined key comparison function
`array_fill()`	Fill an array with values
`array_fill_keys()`	Fill an array with values, specifying keys
`array_filter()`	Filters the values of an array using a callback function
`array_flip()`	Flips/Exchanges all keys with their associated values in an array
`array_intersect()`	Compare arrays, and returns the matches (values)
`array_intersect_assoc()`	Compare arrays, and returns the matches (values and keys)
`array_intersect_key()`	Compare arrays, and returns the matches (keys)
`array_intersect_uassoc()`	Compare arrays, and returns the matches (values and keys), using a user-defined key comparison function

Function	Description
`array_intersect_ukey()`	Compare arrays, and returns the matches (keys), using a user-defined key comparison function
`array_key_exists()`	Checks an array for a specified key, and returns true if the key exists and false if the key does not exist
`array_keys()`	Return all the keys of an array
`array_map()`	Applies a user-defined function to each element of an array
`array_merge()`	Merge one or more arrays into one array
`array_merge_recursive()`	Merge one or more arrays into one array recursively
`array_multisort()`	Sort multiple or multi-dimensional arrays
`array_pad()`	Insert a specified number of items, with a specified value, to an array
`array_pop()`	Deletes the last element of an array
`array_product()`	Calculate the product of values in an array
`array_push()`	Insert one or more elements to the end of an array
`array_rand()`	Pick one or more random keys out of an array
`array_reduce()`	Reduces an array to a single value by using a callback function
`array_replace()`	Replaces the values of the first array with the values from following arrays
`array_replace_recursive()`	Replaces the values of the first array with the values from following arrays recursively
`array_reverse()`	Return an array in the reverse order
`array_search()`	Search an array for a value and return the key
`array_shift()`	Removes the first element from an array, and returns the value of the removed element
`array_slice()`	Extract a slice of the array
`array_splice()`	Remove a portion of the array and replace it with something else
`array_sum()`	Calculate the sum of values in an array
`array_udiff()`	Compare arrays, compare the values of the arrays, and returns the differences
`array_udiff_assoc()`	Compare arrays, compare the values of the arrays, and returns the differences (values and keys)

Function	Description
`array_udiff_uassoc()`	Compare arrays, compare the values of the arrays, and returns the differences (values and keys), using a user-defined key comparison function
`array_uintersect()`	Compare arrays, compare the values of the arrays, and returns the matches
`array_uintersect_assoc()`	Compare arrays, compare the values of the arrays, and returns the matches (values and keys)
`array_uintersect_uassoc()`	Compare arrays, compare the values of the arrays, and returns the matches (values and keys), using a user-defined key comparison function
`array_unique()`	Removes duplicate values from an array
`array_unshift()`	Add one or more elements to the beginning of an array
`array_values()`	Return all the values of an array
`array_walk()`	Apply a user-defined function to every element of an array
`array_walk_recursive()`	Apply a user-defined function recursively to every element of an array
`arsort()`	Sort an associative array in descending order, according to the value
`asort()`	Sort an associative array in ascending order, according to the value
`compact()`	Create an array from variables and their values
`count()`	Count all elements in an array, or something in an object
`current()`	Return the current element in an array
`each()`	Return the current key and value pair from an array and advance the array cursor
`end()`	Set the internal pointer of an array to its last element
`extract()`	Import variables into the current symbol table from an array
`in_array()`	Checks if a value exists in an array

Document

Document in project

You can Download PDF file.

Reference

PHP Global Variables - Superglobals

Back to ACP page

Table of Contents

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

$GLOBALS $_SERVER $_REQUEST $_POST $_GET $_FILES $_ENV $_COOKIE $_SESSION The next chapters will explain some of the superglobals, and the rest will be explained in later chapters.

Document

Document in project

You can Download PDF file.

Reference

PHP Superglobals GET

Back to ACP page

Table of Contents

PHP Query string in the URL

1-tag-a.php

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

<a href="2-demo_phpfile.php?subject=PHP&web=W3schools.com">Click</a>

</body>
</html>
```

Click

Example 1 1

Result View Example

2-demo_phpfile.php

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

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
```

```
?>

</body>
</html>
```

Click

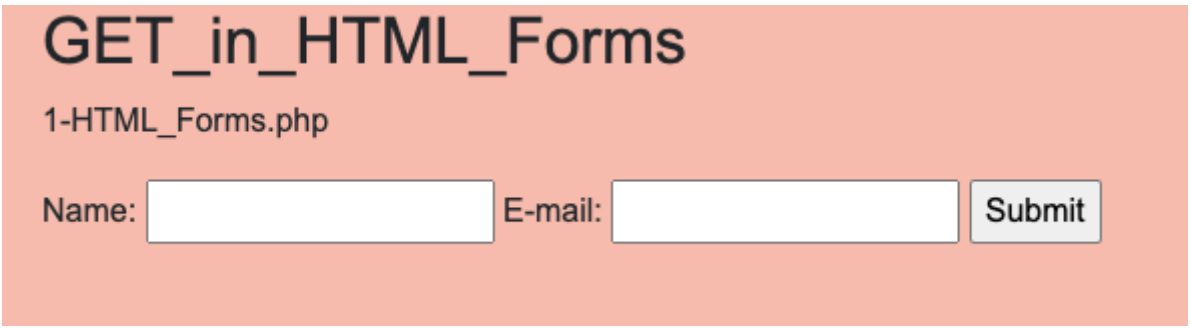
Example 1 2

Result View Example

PHP GET_in_HTML_Forms

1-HTML_Forms.php

```
<h2>GET_in_HTML_Forms</h2>
<p>1-HTML_Forms.php </p>

<form action="2-welcome_get.php" method="GET">
    Name: <input type="text" name="name">
    E-mail: <input type="text" name="email">
    <input type="submit">
</form>
```



Example 2 1

Result View Example

```
<h2>GET in HTML Forms</h2>
<p>2-welcome_get.php </p>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
```

Example 2 2

Result View Example

Document

Document in project

You can Download PDF file.

Reference

PHP Superglobals - Sessions

Back to ACP page

Table of Contents

PHP Start PHP Session

A session is a way to store information (in variables) to be used across multiple pages. By default, session variables last until the user closes the browser.

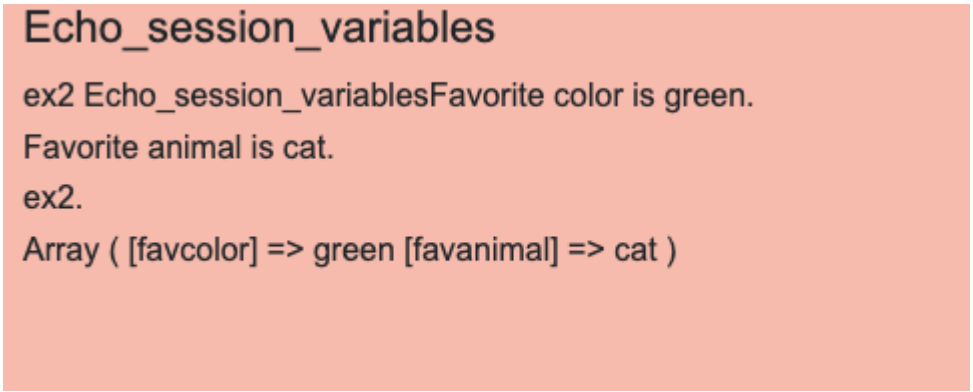
```
<?php
    /**
     * Start_PHP_Session.php
    */
    // Start the session
    session_start();
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <title>phpTeeratus</title>
</head>
<body>
<?php

// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";

echo "Session variables are set.";
?>
</body>
</html>
```

Example 1

Result View Example

Echo session variables

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>

<body>
    <h4>Echo_session_variables</h4>
    <?php
    // Echo session variables that were set on previous page
    echo "ex2 Echo_session_variables";
    echo "Favorite color is " . $_SESSION['favcolor'] . ".<br>";
    echo "Favorite animal is " . $_SESSION['favanimal'] . ".<br>";

    echo "ex2" . ".<br>";
    print_r($_SESSION);
    ?>
</body>
</html>
```

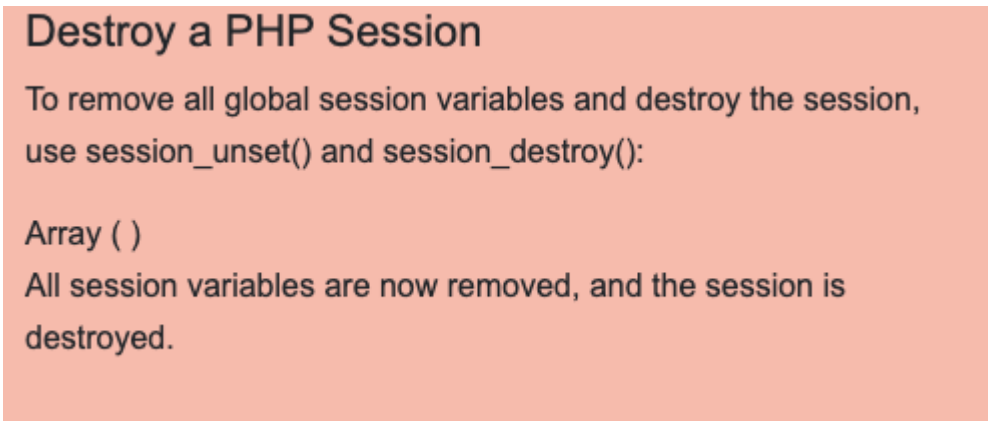


Example 2

Result View Example

Destroy a PHP Session

```php
<?php
    // Start the session
    session_start();
?>
<!DOCTYPE html>
<html lang="en">
<body>

        <h2>phpTeeratus</h2>
        <p>From w3schools.com, Experiment by Teeratus_R </p>

        <h4>Destroy a PHP Session</h4>
        <p>
        To remove all global session variables and destroy the session,
use session_unset() and session_destroy():
        </p>


        <?php
        // remove all session variables
        session_unset();

        // destroy the session
        session_destroy();

        print_r($_SESSION);
        echo "<br>";
        echo "All session variables are now removed, and the session is
destroyed.";
        ?>

</body>
</html>
```



Example 3

Result View Example

Document

Document in project

You can Download PDF file.

Reference