PHP Functions

The real power of PHP comes from its functions.
PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

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 Functions

PHP User Defined Functions

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

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

Create a User Defined Function in PHP

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

Syntax

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

PHP Functions

PHP User Defined Functions

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

Tip: Give the function a name that reflects what the function does!

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name followed by brackets ():

```
<?php
function writeMsg() {
  echo "Hello world!";
}
writeMsg(); // call the function
?>
```

PHP Functions

PHP Function Arguments

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

PHP Functions

PHP Function Arguments

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

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

PHP Functions

PHP Function Arguments

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

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

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

PHP Functions

PHP is a Loosely Typed Language

In the example 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.

PHP Functions

PHP is a Loosely Typed Language

In the following example we try to send both a number and a string to the function without using strict:

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

PHP Functions

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

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

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

The strict declaration forces things to be used in the intended way.

PHP Functions

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

```
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
  echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions

PHP Functions - Returning values

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

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
  $z = $x + $y;
  return $z;
}

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

PHP Functions

PHP Return Type Declarations

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

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

PHP Functions

PHP Return Type Declarations

You can specify a different return type, than the argument types, but make sure the return is the correct type:

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

PHP Functions

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: Use a pass-by-reference argument to update a variable:

```
<?php
function add_five(&$value) {
  $value += 5;
}
$num = 2;
add_five($num);
echo $num;
?>
```