Using and writing functions

Using and writing functions

A function is a named block of code that can be executed by simply calling (invoking) its name. We have already used several functions in our scripts:

The rand() function

The PHP rand() function has the following signature:

int rand ([ int $min, int $max ] )

The first word (int) is its return type, which specifies the type of data it returns. rand() returns an integer (whole number). Following its name is its parameter list (in parentheses). The fact that $min and $max are enclosed in square brackets means that these are optional parameters. When used with no parameters, rand() will return a random integer between 0 and RAND\_MAX (which varies from computer to computer). Otherwise rand() is used with two parameters, one specifying the minimum and the other the maximum possible random values, inclusively.

<?php

echo rand() . "<BR>"; // Outputs something like 1887

echo rand() . "<BR>"; // Outputs something like 31881

echo rand(5, 15); // Outputs something like 12

$die1 = rand(1,6); // The next three statements simulate rolling of dice

$die2 = rand(1,6);

echo "You just rolled the dice: $die1 $die2";

?>

Notice how the arguments 5 and 15 are substituted for the parameters $min and $max when the rand() function is called. The parameters serve as place holders for data that are passed to the function.

Function calls

When you call a function from within a PHP script, the script will jump to the function, execute the code it finds there and then return to where it left, possibly returning a value:

<?php

$mystring = "mystring";

$signature = md5($mystring); // Jump to md5() function and return its value.

echo $signature; // Prints a 128-bit long binary string representing "mystring"

?>

In this example, the script jumps to the md5() function (located somewhere in the PHP function library). This is a built-in hash function that runs an algorithm over the individual letters in "mystring" and returns a 128-bit message digest.

Defining functions

To define your own function, you need to follow the prescribed syntax:

function function\_name([parameters]) { code to execute; }

Let's write a simple function that prints hello:

<?php

function hello() {

echo ("Hello from inside the hello() function");

}

// Call hello()

hello(); // Prints 'Hello from inside the hello() function'

?>

Let's improve the function by adding a parameter for a person's name and have it say hello to that person. Notice that in PHP you just give the name of the parameter not its type (as in some other languages).

<?php

function hello($name) {

echo ("Hello $name from inside the hello() function");

}

// Call hello()

hello("Dolly"); // Prints 'Hello Dolly from inside the hello() function'

hello("Mother"); // Prints 'Hello Mother from inside the hello() function'

?>

Let's improve it some more by adding a parameter that tells it how many times to print hello.

<?php

function hello($name, $n) {

for ($k=0; $k < $n; $k++)

echo ("Hello $name from inside the hello() function");

}

// Call hello()

hello("Dolly",2); // Prints 'Hello Dolly from inside the hello() function' twice

hello("Mother",1); // Prints 'Hello Mother from inside the hello() function' once

?>

Parameters can be given default values, which makes the parameter optional when the function is called:

<?php

function hello($name, $n=1) {

for ($k=0; $k < $n; $k++)

echo ("Hello $name from inside the hello() function");

}

// Call hello()

hello("Dolly",2); // Prints 'Hello Dolly from inside the hello() function' twice

hello("Mother"); // Prints 'Hello Mother from inside the hello() function' once

?>

Defining functions that return a value

The hello() function just printed some output. It didn't return a value. Some functions return a value that represents the result of their computation:

<?php

// Computes n^k power

function power($n, $k) {

$result = 1;

for ($i = 0; $i < $k; $i++)

$result = $result \* $n;

return $result;

}

$twocubed = power(2,3); // Assigns 8 to $twocubed

echo ("Two cubed is $twocubed<br>"); // Prints 'Two cubed is 8'

echo ("Three squared is ". power(3,2) . "<br>"); // Prints 'Three squared is 9'

echo ("Two to the 10th power is ". power(2,10) . "<br>"); // Prints 'Two to the 10th power is 1024'

?>

Note that a function that returns a value is an expression and can be embedded in other expressions.

Example Functions

Click here for a couple of examples.

In class exercises

Write and test a PHP function that accepts two numbers and that calculates the sum of the numbers in between those two numbers. For example, sum(1, 10) would return 55.

Write and test a PHP function that accepts a number, $n, and returns its factorial. The factorial of n is n \* (n-1) \* (n-2) \* ... \* 1. The factorial of 0 is 1.

Write and test a PHP function called countBlanks($str) that counts the number of blanks in its string argument. To use determine the length of the string use: strlen($str). To refer to a single letter in the string use $str[0] to refer to the first letter or $str[$k] to refer to the kth letter.

Click for solutions to the exercises.