Functions and Variable Scope

Copyright Notice

Last Update: Sat, 10 Aug 2002 00:00:00 –0400

Table of Contents

Functions and Variable Scope.....1

Functions and Variable Scope

Navigate: PHP Tutorials > PHP > Basics & Beginner Tutorials

Author: Lakario
Date: 08/10/2002
Version 1.0
Experience Level: Unknown

Okay this is a little tutorial on functions and variable scope, hope you enjoy.

In php as you write your code it is very likely that you will find yourself doing one action over and over possibly one that is very lengthy and when it's lengthy it's not always fun to make the code each time, this is the very reason functions were designed.

A function is a declared piece of code that has code as members of it. All the things you use when you are coding are functions like print(), echo() and exit(). You can create your own functions in PHP to do similar things as these. Let's look at the layout for writing a function.

PHP Example:

```
function function_name(arguments if any) {
procedures
}
```

Okay well that's pretty straight forward right? If not, I'll give a small example:

PHP Example:

```
function Hello() {
echo "Hello World!";
}
```

The function above simply prints out the words "Hello World!" when called. Making more sense now? Cool.

Now you are a probably wondering about my little blurb up above about arguments, so let's explain those. An argument is basically data in the form of variables you desire to pass to a function to have it use in its execution. Another reason you would pass an argument is if you wanted to use a variable declared outside of the function inside, that is variable scope, I will cover that more later on in the tutorial. Basically, when you want to pass a function some arguments you put them inside the parenthesis after the name of the function, which is required even if you don't have arguments. When declaring an argument to be used in the function it

is always in the form of a variable, regardless of whether or not it is going to be called as one, you'll see what I mean in a minute. Another thing to remember is that when you call a function that has variables declared to it, you must pass the variables in the order they were declared. Also note that when you are declaring or passing variables to an argument, each are separated by a comma. Okay that was a lot of confusing text so let me clear it up with some code for you.

PHP Example:

```
function HelloJoe($string, $name) {
echo "$string $name";
}

$string = "Hello, ";
$name = "Joe";

HelloJoe($string, $name);
```

The output there would be the words "Hello, Joe" being printed to the screen. I hope this is making more sense, because we still have a bit to go.

Variable scope refers to where variables are equal to something, when you are using a function $var1 outside the function is not the same as $var1 inside the function, and vise versa. That is why you pass the variable you want used inside the function as an argument, but what if you want to retrieve a variable declared inside the function, outside the function? That is what return() the return function is used to pass a variable (and it's value) out of the code once it is created or modified within bare in mind that return() is somewhat like exit() in that it halts the execution of the function, so don't put it in the middle of your function. Let's make an example:

PHP Example:

```
function Adding($number) {
$number = $number+5;
return $number;
}
<br>
$number = 5;
$newnumber = Adding($number);
<br>
echo "$newnumber";
```

Well that's pretty straight forward, basically what we did there is passed the argument $number as equal to 6 then inside the function we added 5 to the number then used return() to pass along the processed number. Also another thing to note about passing arguments is that you do not have to pass everything you desire to use, that is what the command global is for. Global refers to variables that are in the global variable scope as opposed to the local variable scope of the function. An example:

PHP Example:

```
function EchoFileName() {
global $PHP_SELF;
echo "$PHP_SELF";
```

```
}

EchoFileName();
```

That piece of code would echo the name of our file, on the other hand, this would not work:

PHP Example:

```
function EchoFileName() {
echo "$PHP_SELF";
}
```

One thing I haven't bothered to explain about passing and declaring arguments is the fact that the name of the argument declared does not have to be the name that is used when you call it, that is why it is so important to pass variables in the order they were declared.

Wow, we did it, that's pretty much everything you really need to know about functions, the only other thing is recursion which refers to calling a function within itself, but because recursion is almost never used except for very complex system I don't think it is necessary to include it in this tutorial. In closing, let me provide you with a note about naming functions.

When you name a function there are a couple simple things to keep in mind.

Your function name cannot be the same as another function's name.
Your function name can only contain letters, digits, and underscores.
Your function name cannot start with a digit.

The following names are legal:

teddybear()
teddy_bear()
teddy_bear_4()
_teddybear()

The following names are not legal:

4teddy_bear()
teddy−bear()
echo()

(The last one is not legal only because the function already exists.)

So there you go, that's pretty much everything there is to know about functions, happy programming.
