

# SERVER-SIDE WEB PROGRAMMING UNIT2: PROGRAMMING BASED ON EMBEDDED LANGUAGE

# Index

- Functions
  - Calling a Function
  - Defining a Function
  - Function Parameters
  - Variable Parameters
  - Missing Parameters
  - Variable Functions
  - Internal built-in functions

## 2. Functions

3

- A function is a named block of code that performs a specific task, possibly acting upon a set of values given to it, or parameters, and possibly returning a single value.
- *Example: `print_header` or `validate_user`*
- Code reuse
- Also, it makes easier to use code that other people have written.

## 2.1. Calling a Function

4

- Functions in a PHP program can be *built-in* or *user-defined*.
- Regardless of their source, all functions are evaluated in the same way:  
`$someValue = function_name( [ parameter, ... ] );`

## 2.1. Calling a Function

5

- Here are some examples of functions:

```
// strlen() is a built-in function that returns the length of a string
$length = strlen("PHP"); // $length is now 3

// sin() and asin() are the sine and arcsine math functions
$result = sin(asin(1)); // $result is the sine of arcsin(1), or 1.0

// unlink() deletes a file
$result = unlink("functions.txt"); // false if unsuccessful
```

```
function page_header() {
    print '<html><head><title>Welcome to my site</title></head>';
    print '<body bgcolor="#fff1ff">';
}

page_header();
print "Welcome, Ines";
print "</body></html>";
```

## 2.2. Defining a Function

6

- To define a function, use the following syntax:

```
function [&] function_name([parameter[, ...]])  
{  
    statement list  
}
```

- The statement list can include HTML, with or without PHP code. For instance:

```
<?php function column()  
{ ?>  
    </td><td>  
<?php}
```

## 2.2. Defining a Function

7

- Function names are case-insensitive.
- If we put this function on a PHP page, we can call it from anywhere within the page (*In the near future we will learn how to refer to “functions” in another php file*)
- Typically, functions return some value. To return a value from a function, use the return statement:

```
function strcat($left, $right)
{
    $combinedString = $left . $right;
    return $combinedString;
}
```

## 2.2. Defining a Function

8



A.2.6. What does this function does?

```
function doubler($value)
{
    return $value << 1;
}
```





## 2.2. Defining a Function

10



A.2.7. Try this:

1. Change previous *page\_header* method (slide 5), in order to take the page color as an argument.
2. After that, when you test the code, check out what happens at runtime if you pass an argument to the method (and if not).
3. How can we avoid that warning??
4. If we had more than one argument, where should optional arguments be?



## 2.2. Defining a Function

11

- PHP functions can return only a single value with the return keyword:

```
function returnOne()  
{  
    return 42;  
}
```

- To return multiple values, return an array:

```
function returnTwo()  
{  
    return array("Fred", 35);  
}
```

## 2.3. Function Parameters

14

- There are two different ways to pass parameters to a function. The first, and more common, is by value. The other is by reference.
  1. Passing Parameters by Value: PHP engine copies the value of the parameter.

```
function countdown($stop) {  
    while ($stop > 0) {  
        print "$stop..";  
        $stop--;  
    }  
    print "boom!\n";  
}  
$counter = 5;  
countdown($counter);  
print "Now, counter is $counter";
```

## 2.3. Function Parameters

15

2. Passing Parameters by Reference: give a function direct access to a variable.

```
<?php
function doubler(&$value)
{
    $value = $value << 1;
}
$a = 3;
doubler($a);
echo $a;
```

## 2.3. Function Parameters

16

- Unless you tell the PHP engine, function arguments and return values don't have any constraints on their types or values.

```
function countdown($stop) {  
    while ($stop > 0) {  
        print "$stop..";  
        $stop--;  
    }  
    print "boom!\n";  
}  
$counter = 5;  
countdown($counter);  
print "Now, counter is $counter";
```

- But... **Type declarations** are a way to express constraints on argument values (new in PHP7).

## 2.3. Function Parameters

17

- **Type declarations** for the passed parameters:


```
function countdown(int $stop) {  
    while ($stop > 0) {  
        print "$stop..";  
        $stop--;  
    }  
    print "boom!\n";  
}  
$counter = 5;  
countdown($counter);  
print "Now, counter is $counter";
```



## 2.3. Function Parameters

18

- PHP 7 also supports *type declarations* for the kind of value a function returns:



```
function restaurant_check($meal, $tax, $tip): float {  
    $tax_amount = $meal * ($tax / 100);  
    $tip_amount = $meal * ($tip / 100);  
    $total_amount = $meal + $tax_amount + $tip_amount;  
    return $total_amount;  
}  
echo "\n".restaurant_check(5,5,2);
```

## 2.4. Variable Parameters

19

- ***A function may require a variable number of arguments!!!***
- To do so, leave out the parameter block entirely:

```
function getPreferences()  
{  
    // some code  
}
```

- PHP provides three functions you can use in the function to retrieve the parameters passed to it:

`$array = func_get_args();`

`$count = func_num_args();`

`$value = func_get_arg(argument_number);`



## 2.4. Variable Parameters

20

```
<?php
function countList()
{
    if (func_num_args() == 0) {
        return false;
    }
    else {
        $count = 0;
        for ($i = 0; $i < func_num_args(); $i++) {
            $count += func_get_arg($i);
        }
        return $count;
    }
}
echo countList(1, 5, 9); // outputs "15"
```



## 2.5. Missing Parameters

21

- PHP lets you be as lazy as you want: when you call a function, you can pass any number of arguments to the function.
- Any parameters the function expects that are not passed to it remain unset, and a warning is issued for each of them.
- Optional attributes must be at the end of the passed parameters.

## 2.5. Missing Parameters

22

```
function takesTwo($a, $b)
{
    if (isset($a)) {
        echo " a is set\n";
    }
    if (isset($b)) {
        echo " b is set\n";
    }
}
echo "With two arguments:\n";
takesTwo(1, 2);
echo "With one argument:\n";
takesTwo(1);
```



## 2.6. Variable Functions

23

- You can add parentheses after a variable to call the function whose name is the value held by the apparent variable. Consider this:

```
switch ($which) {  
    case 'first':  
        first();  
        break;  
    case 'second':  
        second();  
        break;  
    case 'third':  
        third();  
        break;  
}  
  
if (function_exists($which)) {  
    $which(); // if $which is "first", the function first() is called, etc...  
}
```

## 2.7. Internal built-in functions

24

- There is no installation needed to use these functions, they belong to PHP core:
  - PHP String functions
  - PHP arrays functions
  - (and so on...)