Course:
PHP from scratch

by Sergey Podgornyy

Functions

WEB ACADEMY
PROGRAMMING COURSES

About me

Sergey Podgornyy

Full-Stack Web Developer

WebbyLab

and

ignite software outsourcing

ZEND CERTIFIED PHP ENGINEER

Linux Professional Institute LPIC-1

Overview

- What is functions? Function types
- How to create user defined function?
- Constant `__FUNCTION__`
- Function calls
- Function arguments, defining their types
- Scope; local scope / global scope
- Returning values from functions
- func_get_args() and REST params
- Anonymous functions
- String / Array / Regexp functions
- Recursion
- Static variable

Functions

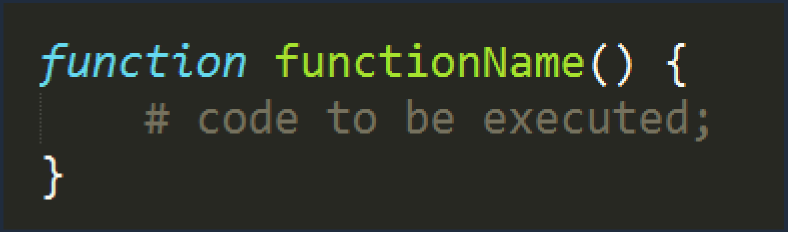
The real power of PHP comes
from its functions; it has
more than 1000 built-in functions

Internal
(built-in)
functions

User
defined
functions

User Defined Functions

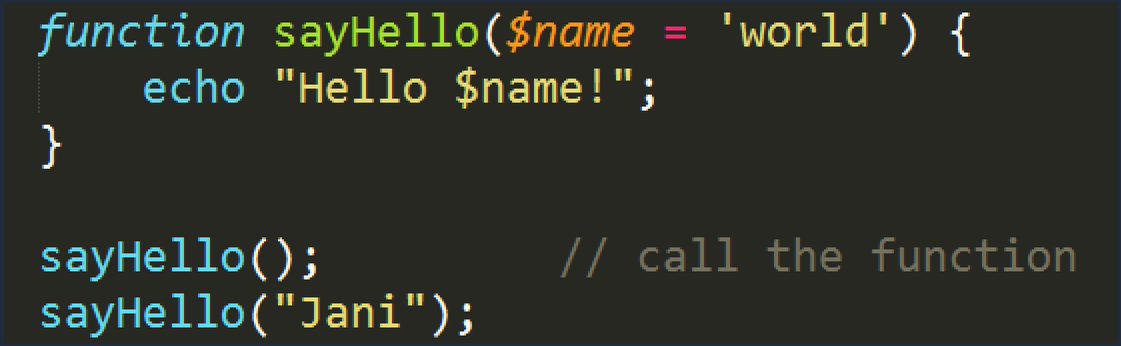
- Besides the built-in PHP functions, we can create our own functions
- A function is a block of statements that can be used repeatedly in a program
- A function will not execute immediately when a page loads
- A function will be executed by a call to the function



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

Function arguments

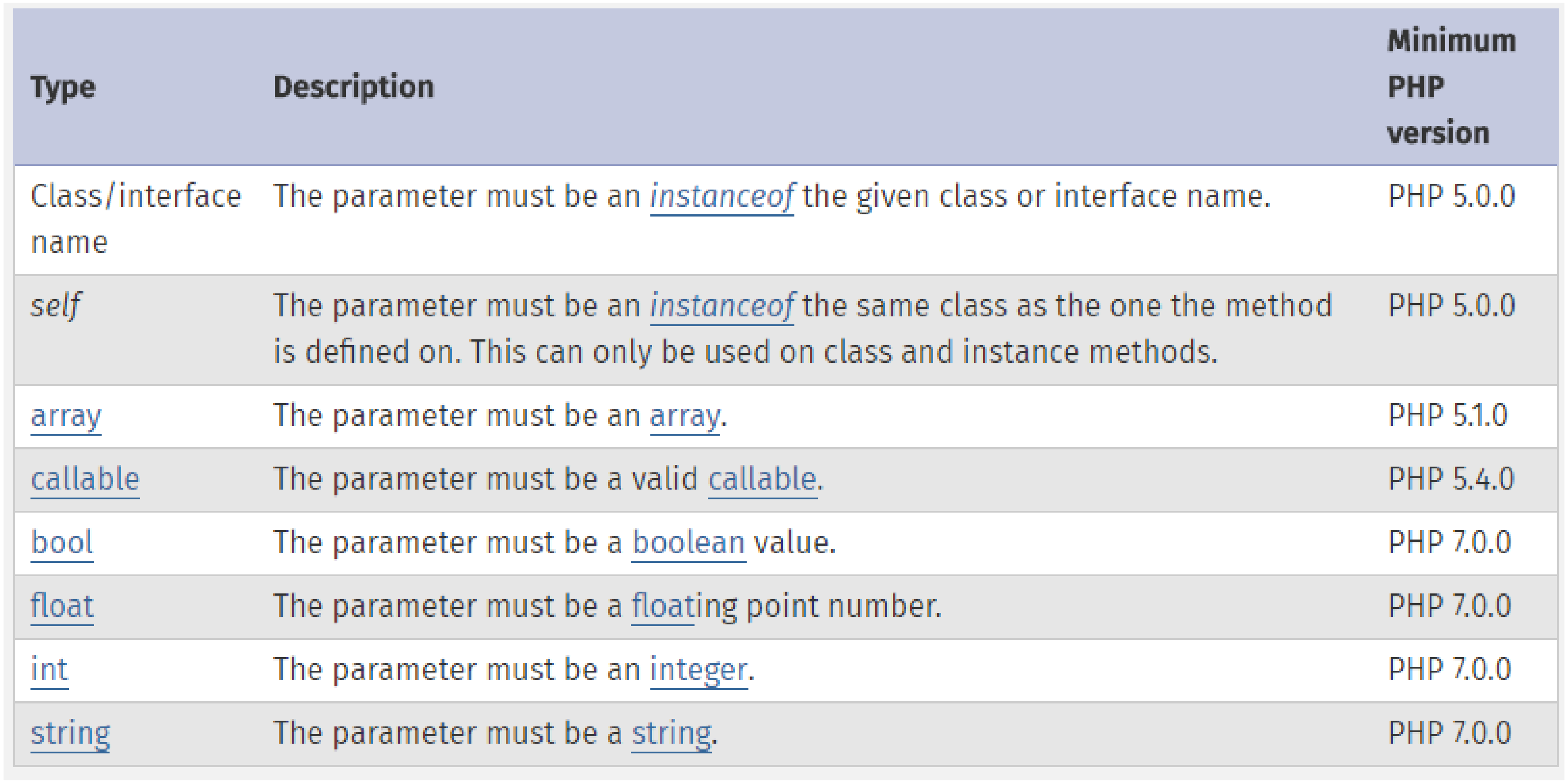
Information may be passed to functions via the argument list, which is a comma-delimited list of expressions. The arguments are evaluated from left to right



```php
function sayHello($name = 'world') {
    echo "Hello $name!";
}

sayHello();          // call the function
sayHello("Jani");
```

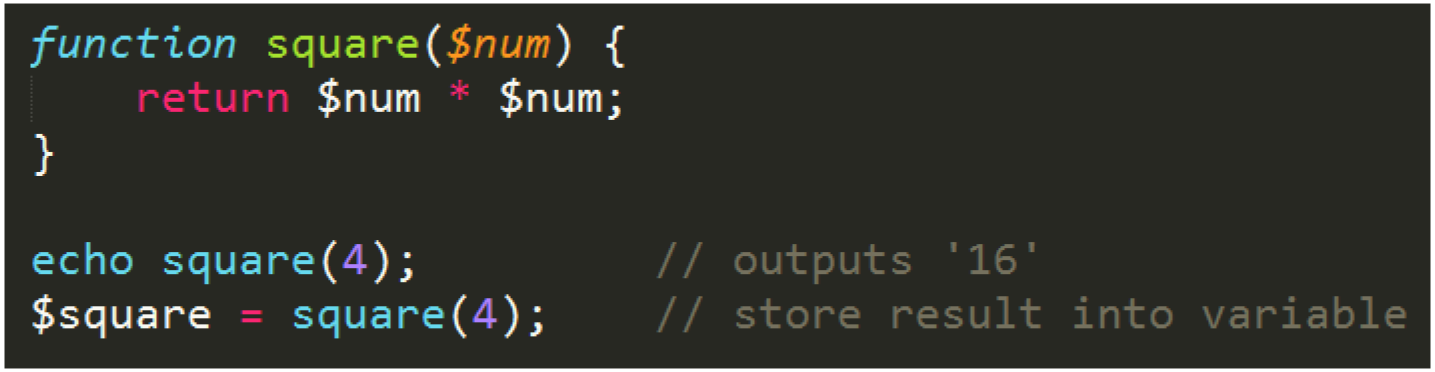
Valid types for arguments



Type	Description	Minimum PHP version
Class/interface name	The parameter must be an *instanceof* the given class or interface name.	PHP 5.0.0
self	The parameter must be an *instanceof* the same class as the one the method is defined on. This can only be used on class and instance methods.	PHP 5.0.0
array	The parameter must be an array.	PHP 5.1.0
callable	The parameter must be a valid callable.	PHP 5.4.0
bool	The parameter must be a boolean value.	PHP 7.0.0
float	The parameter must be a floating point number.	PHP 7.0.0
int	The parameter must be an integer.	PHP 7.0.0
string	The parameter must be a string.	PHP 7.0.0

Returning values

- Values are returned by using the optional return statement
- If the return is omitted the value NULL will be returned



```php
function square($num) {
    return $num * $num;
}

echo square(4);          // outputs '16'
$square = square(4);     // store result into variable
```

Get function args

There are 2 ways to get function arguments:

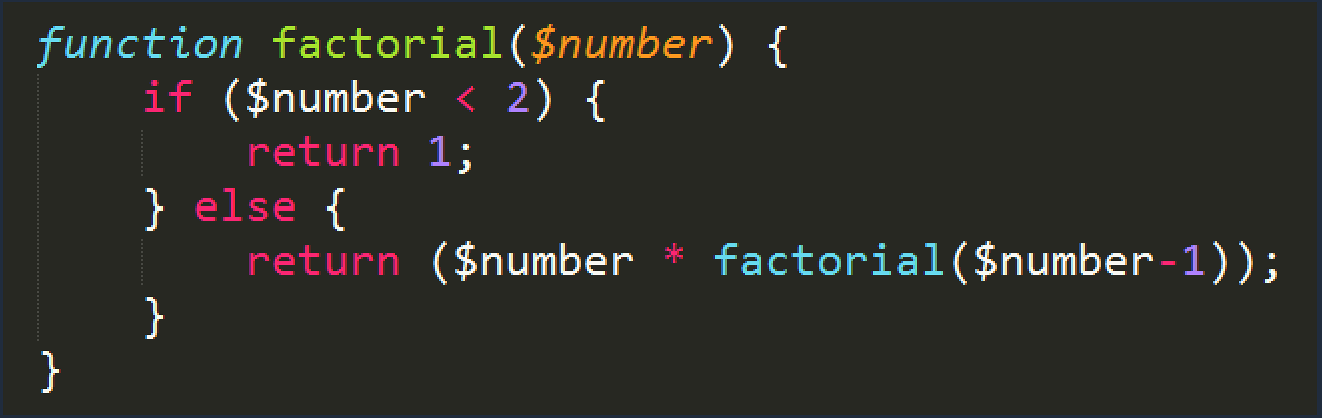
- using func_get_args — Returns an array comprising a function's argument list

OR

- using REST params

Recursive Function

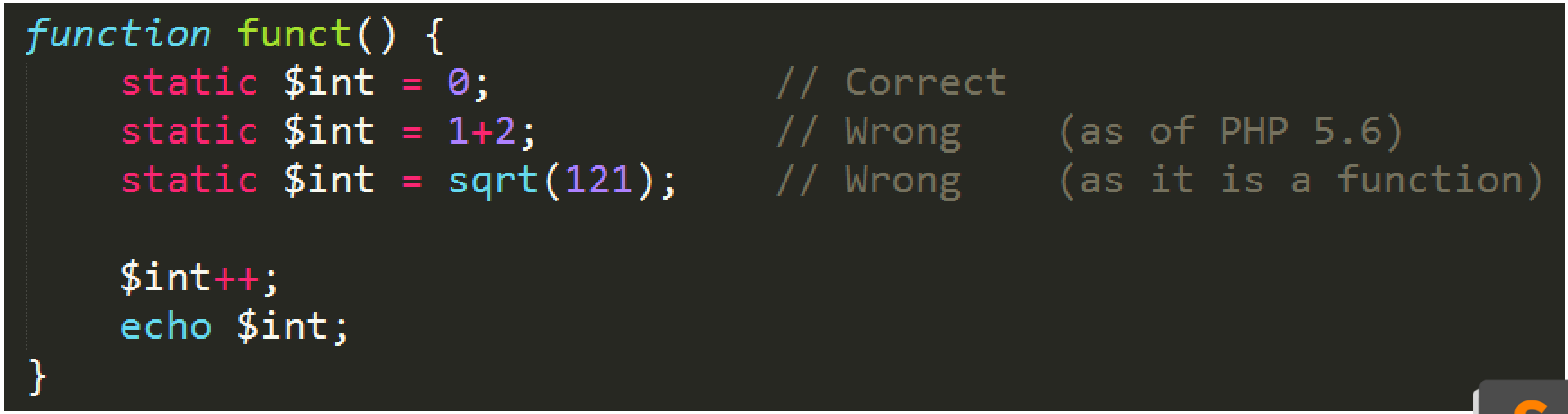
Recursion is a method where the solution to a problem depends on solutions to smaller instances of the same problem



```
function factorial($number) {
    if ($number < 2) {
        return 1;
    } else {
        return ($number * factorial($number-1));
    }
}
```

Static variables in functions

- Another important feature of variable scoping is the *static* variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope

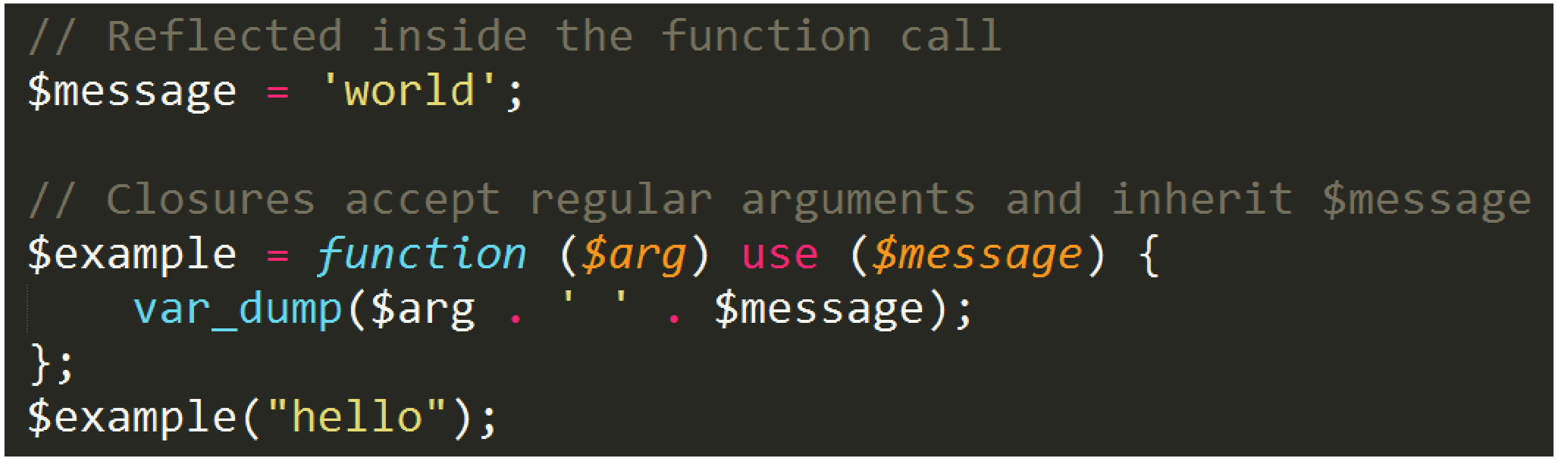


```php
function funct() {
    static $int = 0;            // Correct
    static $int = 1+2;          // Wrong    (as of PHP 5.6)
    static $int = sqrt(121);    // Wrong    (as it is a function)

    $int++;
    echo $int;
}
```

Anonymous functions

Anonymous functions, also known as ***closures***, allow the creation of functions which have no specified name. They are most useful as the value of ***callback*** parameters, but they have many other uses



```
// Reflected inside the function call
$message = 'world';

// Closures accept regular arguments and inherit $message
$example = function ($arg) use ($message) {
    var_dump($arg . ' ' . $message);
};
$example("hello");
```

Anonymous functions are implemented using the Closure class

String functions

The PHP string functions are part of the PHP core.
No installation is required to use these functions

- strlen()
- strpos()
- substr()
- str_replace()
- explode()
- implode()
- trim()

str_split()
strcmp()
strrev()
strtolower()
strtolower()
md5()
number_format()
printf()

Array functions

The array functions allow you to access and manipulate arrays

- array_push()
- array_pop()
- array_unshift()
- array_shift()
- count()
- list() / compact()
- in_array()
- array_merge()
- array_unique()

Regular Expressions

Symbols **Quantifiers**

Delimiters

Often used delimiters are forward slashes (/), hash signs (#) and tildes (~). The following are all examples of valid delimited patterns

```
/foo bar/
#^[^0-9]$#
+php+
%[a-zA-Z0-9_-]%
```

Regular Expressions

- Symbols:

`a`	- symbol
`[abc]`	- one of symbols
`[^abc]`	- not one of symbols
`(a|b|c)`	- one of symbols (using entities)

- Quantifiers:

`{0, 3}`	- occurs 0…3 times
`?`	- occurs never or once `{0,1}`
`*`	- occurs any number of times `{0, }`
`+`	- occurs one or more times

Regular Expressions

Meta-characters

`^`	assert start of subject
`$`	assert end of subject or before a terminating new line
`.`	match any character except newline
`\`	general escape character with several uses

Escape sequences

`\n`	new line
`\t`	tab
`\s`	any whitespace character
`\d`	any decimal digit
`\w`	any word character

PCRE functions

- `preg_match`
- `preg_replace`
- `preg_split`

Useful resources

- Functions
- Anonymous functions
- String functions
- Array functions
- PCRE (Perl compatible regular expression)
- PCRE functions

Thanks for your attention

Q & A