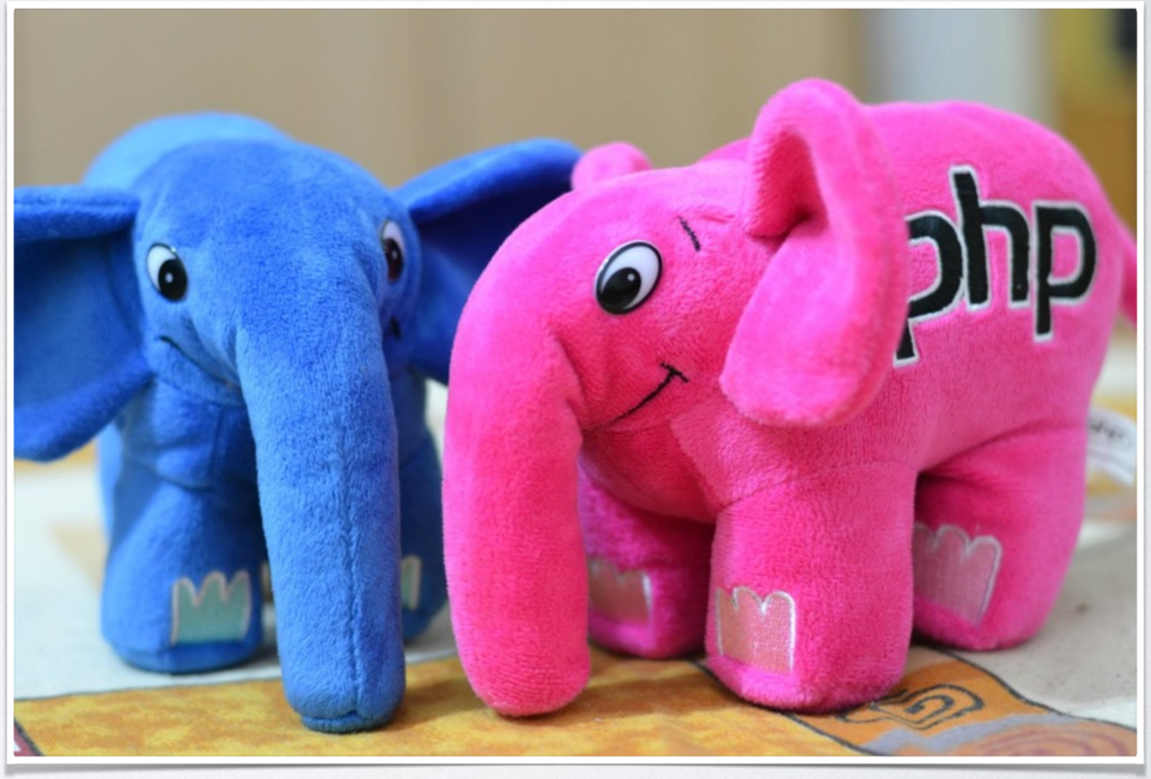
PHP5 BASIS

Hypertext Preprocessor

INTRODUCTION

- PHP: Hypertext Preprocessor
- Server-side NOT Client-side
- CLI (Command-Line Interface)



PHP ON MY COMPUTER

WAMP	**MAMP**	**XAMPP**
Windows	Mac	Linux
		+ Windows
		+ Mac
wampserver.com	mamp.info	apachefriends.org

PHP ON MY COMPUTER

Install an IDE (Integrated Development Environment)

PhpStorm from JetBrains (Why ? Because it's the best ever)

PHP ON MY COMPUTER

RUN MY FIRST PHP CODE

- File => index.php
- <?php echo “Hello World” ?>
- Go to http://localhost/index.php (You have to see “Hello World”)

BASIC SYNTAX

PHP TAGS

ESCAPING FROM HTML

INSTRUCTION SEPARATION

COMMENTS

BASIC SYNTAX

PHP TAGS

BASIC SYNTAX

PHP TAGS

- PHP parsing looks for opening "<?php" and closing "?>" tags
- It tells PHP to start and stop interpreting the code between them
- Every code outside of a pair of opening and closing tags is ignored by the PHP parser

BASIC SYNTAX

PHP TAGS

- Short open tags “<?” and “?>” can be used
- Usage is discouraged because they are only available if enabled with “short_open_tag” in php.ini configuration file

BASIC SYNTAX

PHP TAGS

```
<?php ?> // standard tags

<? ?> // short tags, need short_open_tag enabled in php.ini
```

BASIC SYNTAX

PHP TAGS

- If a file is pure PHP code, it is preferable to omit the PHP closing tag at the end of the file.

- This prevents accidental whitespace or new lines being added after the PHP closing tag, which may cause unwanted effects because PHP will start output buffering when there is no intention from the programmer to send any output at that point in the script.

BASIC SYNTAX

PHP TAGS

- Fix the following error :

  “Warning: Cannot modify header information - headers already sent”

```php
<?php

echo "My First echo statement";

// ... more code

echo "My Second echo statement";

// the script ends here with no PHP closing tag
```

BASIC SYNTAX

ESCAPING FROM HTML

TOO EASY

...

BASIC SYNTAX

INSTRUCTION SEPARATION

TOO EASY 2

...

BASIC SYNTAX

COMMENTS

BASIC SYNTAX

COMMENTS

- PHP supports 'C', 'C++' and Unix shell-style (Perl style) comments.

```php
<?php

echo 'This is a test'; // This is a one-line c++ style comment

/* This is a multi line comment
   yet another line of comment */
echo 'This is yet another test';

echo 'One Final Test'; # This is a one-line shell-style comment
```

TYPES

INTRODUCTION

SCALAR TYPES

COMPOUNDED TYPES

SPECIAL TYPES

PSEUDO TYPES

TYPES

INTRODUCTION

TYPES

INTRODUCTION

- The type of a variable is not usually set by the programmer
- It is decided at runtime by PHP depending on the context in which that variable is used

TYPES

SCALAR TYPES

TYPES

SCALAR TYPES

Boolean

Integer

Float

String

TYPES

SCALAR TYPES - BOOLEAN

- A boolean expresses a truth value
- It can be either TRUE or FALSE

```php
<?php

$booleanTrue = true;

$booleanFalse = false;
```

TYPES

SCALAR TYPES - INTEGER

- An integer is a number of the set $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$.

```php
<?php

$integer = -999;

$integer = 0;

$integer = 2015;
```

TYPES

SCALAR TYPES - FLOAT

- Floating point numbers (also known as "floats", "doubles", or "real numbers") can be specified using any of the following syntaxes:

```php
<?php

$float = 1.234;

$float = 1.2e3;

$float = 7E-10;
```

TYPES

SCALAR TYPES - STRING

- A string is series of characters, where a character is the same as a byte.

```php
<?php

$string = 'Oh my God, they killed Kenny!';

$string = '';

$string = '|-|3LL0 \/\/0rLD';
```

TYPES

COMPOUNDED TYPES

TYPES

COMPOUNDED TYPES

Array

Object

TYPES

COMPOUNDED TYPES - ARRAY

- An array in PHP is actually an ordered map. A map is a type that associates values to keys.

- As array values can be other arrays, trees and multidimensional arrays are also possible.

```php
<?php

$array = array(
    'key' => 'value',
);

// As of PHP 5.4
$array = [
    'key' => 'value',
];
```

TYPES

COMPOUNDED TYPES - ARRAY

```php
<?php

// As of PHP 5.4
$array = [
    'key' => [
        'key2' => [
            'key3' => [/*...*/],
        ],
    ],
];
```

TYPES

COMPOUNDED TYPES - OBJECT

- Don’t care about this one for now
- We’ll see it more in detail in OOP lessons!

TYPES

SPECIAL TYPES

TYPES

SPECIAL TYPES

Resource

NULL

TYPES

SPECIAL TYPES - RESOURCE

- A resource is a special variable, holding a reference to an external resource
- Resources are created and used by special functions
- Could be a MySQL link, a stream, …

```php
<?php

// Stream
$fp = fopen('output.csv', 'w');
```

TYPES

SPECIAL TYPES - NULL

- The special NULL value represents a variable with no value. NULL is the only possible value of type null.
- A variable is considered to be null if:
  - It has been assigned the constant NULL.
  - It has not been set to any value yet.
  - It has been unset().

```php
<?php

$null;

$null = null;
```

TYPES

PSEUDO TYPES

TYPES

PSEUDO TYPES

Mixed

Number

Callback / Callable

TYPES

PSEUDO TYPES - MIXED

- Mixed indicates that a parameter may accept multiple (but not necessarily all) types

```php
<?php

/**
 * @param mixed $var
 */
function gettype ($var) {}
```

TYPES

PSEUDO TYPES - NUMBER

- Number indicates that a parameter can be either integer or float

```php
<?php

/**
 * @param number $number
 */
function sub($number) {}
```

TYPES

PSEUDO TYPES - CALLBACK / CALLABLE

- Callback pseudo-types was used in this documentation before callable type hint was introduced by PHP 5.4
- It means exactly the same.

TYPES

SUMMARY 2 REMEMBER

TYPES

SUMMARY 2 REMEMBER

- Boolean => true / false
- Integer => 42
- Float => 3.14159265359
- String => “Some text here”
- Array => [‘key’ => ‘value’]
- Null => null (means no type)

VARIABLES

BASICS

PREDEFINED VARIABLES

VARIABLE SCOPE

VARIABLE VARIABLES

VARIABLES

BASICS

VARIABLES

BASICS

- A variable ALWAYS START with a dollar => $
- After $ it's the variable's name => $name
- Variable's name is CASE-SENSITIVE => $a ≠ $A
- Always write variables in English (it's better to share code and work with collaborators)

VARIABLES

BASICS

- A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

```php
<?php

$myVariableName = true;

$_doNotUseThat = true;

$3exemples = false;
```

VARIABLES

PREDEFINED VARIABLES

VARIABLES

PREDEFINED VARIABLES

- PHP provides a large number of predefined variables to any script which it runs.
- Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors.
- Some of these variables will not be available when PHP is run on the command line.

VARIABLES

PREDEFINED VARIABLES

- $_SERVER => Server and execution environment information
- $_GET => HTTP GET variables
- $_POST => HTTP POST variables
- $_FILES => HTTP File Upload variables
- $argv => Array of arguments passed to script
- …

VARIABLES

VARIABLE SCOPE

VARIABLES

VARIABLE SCOPE

- The scope of a variable is the context within which it is defined.
- For the most part all PHP variables only have a single scope.
- This single scope spans included and required files as well.

VARIABLES

VARIABLE SCOPE

```php
<?php

$hello = 'world';

// $hello variable will be available within test.php
include __DIR__.'/test.php';
```

VARIABLES

VARIABLE VARIABLES

VARIABLES

VARIABLE VARIABLES

- Sometimes it is convenient to be able to have variable variable names.

- That is, a variable name which can be set and used dynamically.

```php
<?php

$kenny = 'status';

$$kenny = 'dead';

echo $$kenny; // Display "dead"
```

CONTROL STRUCTURES

INTRODUCTION

IF / ELSE / ELSEIF

INCLUDE / REQUIRE

WHILE

FOREACH

SWITCH

RETURN

CONTROL STRUCTURES

INTRODUCTION

CONTROL STRUCTURES

INTRODUCTION

- Any PHP script is built out of a series of statements.
- A statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing (an empty statement).

CONTROL STRUCTURES

INTRODUCTION

- Statements usually end with a semicolon.
- In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces.
- A statement-group is a statement by itself as well.

CONTROL STRUCTURES

IF / ELSE / ELSEIF

CONTROL STRUCTURES

IF / ELSE / ELSEIF

- if ($condition) {}
- elseif ($condition) {}
- else {}

CONTROL STRUCTURES

INCLUDE / REQUIRE

CONTROL STRUCTURES

INCLUDE / REQUIRE

- include ‘config.php’;
- include_once ‘config.php’;
- require ‘config.php’;
- require_once ‘config.php’;

CONTROL STRUCTURES

WHILE

CONTROL STRUCTURES

WHILE

- while ($condition) {}

CONTROL STRUCTURES

FOREACH

CONTROL STRUCTURES

FOREACH

- foreach ($array as $key => $value) {}

CONTROL STRUCTURES

SWITCH

CONTROL STRUCTURES

SWITCH

```
switch ($i) {

    case 0:

        echo ‘i equals 0’;

        break;

}
```

CONTROL STRUCTURES

RETURN

CONTROL STRUCTURES

RETURN

- return;
- function myFunction() { return $result; }

THE END

Everything have to be understood!
Questions ?