Chapter 3

Introduction to PHP

Chnoor M. Rahman

Outline

- Why PHP?
- What is PHP?
- PHP data types and variables.
- General structure of PHP.

Why PHP?

- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net.
- Simplicity, speed, and flexibility.
- PHP has remained a dominant language in web development, serving as the backbone for about 80% of websites.
- It currently ranks as the sixth most popular programming language according to the **Popularity of Programming Language Index** (PYPL), and ranked ninth by the TIOBE Index for April 2023.

4

Technology Requirements

- **WAMP| MAMP | LAMP**
  Windows | Mac | Linux, Apache, MySQL, and PHP

  For windows and mac install AMPP

  For Linux install LAMP

  Accessing the Document Root

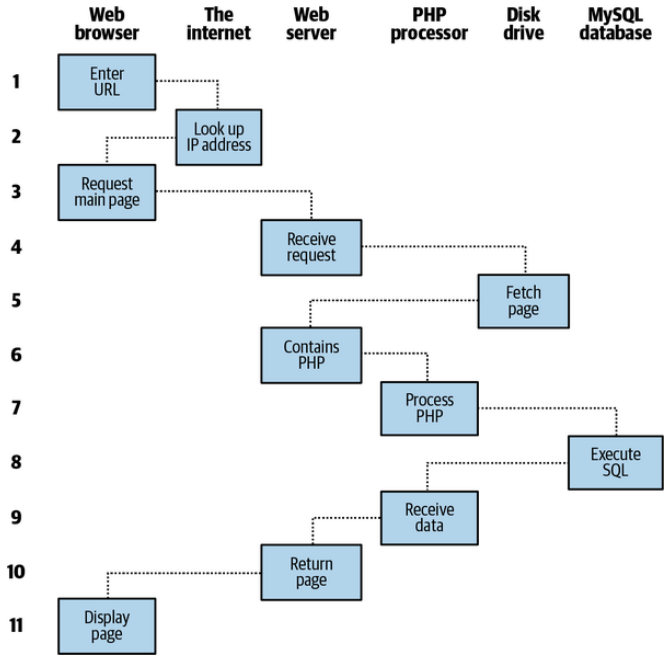
  Windows: C:\Program Files\Ampps\www

  MAC: /Applications/Ampps/www

- **Code editor:** Eclipse | NetBeans | Visual Studio

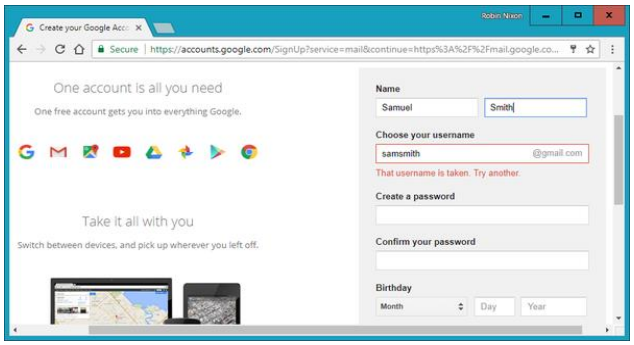
5

A dynamic client/server request/response sequence



6

Gmail uses asynchronous communication to check the availability of usernames



1. The server outputs the HTML to create the web form
2. At the same time, the server attaches some JavaScript to the HTML to monitor the username input
3. After entering the data, in the background the JavaScript code passes the username that was entered back to a PHP script on the web server and awaits a response.
4. The web server looks up the username and replies back to the JavaScript regarding whether that name has already been taken.
5. The JavaScript then places an indication next to the username input box to show whether the name is available or not.
6. If the username is not available and the user still submits the form, the JavaScript interrupts the submission and reemphasizes that the user needs to choose another username.

7

What is PHP?

- PHP stands for Hypertext Preprocessor.
- PHP is a server-side scripting language.
- Its scripts are executed on the server.
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS)

8

Basic PHP Syntax

- A PHP scripting block always starts with <?php and ends with ?>
- The php files are saved with a .php extension.

Example:

```
<?php
……..
……..
?>
```

A PHP file normally contains:
HTML tags, just like an HTML file
Some PHP scripting code.

9

Combining HTML and PHP

```
<html>
  <head> <title>A PHP script including HTML</title> </head>
  <body>
    <b> "hello world- HTML";
      <?php
        echo "hello world-PHP";
      ?>
    </b>
  </body>
</html>
```

10

Case sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are **not case-sensitive**.

However; all **variable names** are **case-sensitive**!

11

Commenting

```
<html>
  <body>
    <?php
      // This is a single-line comment
      # This is also a single-line comment
      /* This is a
      multi-line comment */
    ?>
  </body>
</html>
```

12

Variables

Rules for PHP variables:

- A variable starts with the **$** sign, followed by the name of the variable like $x and $y.
- A variable name must start with a letter or underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

13

Example

```php
<?php
  $txt = "Hello World!";
  $number = 16;
  echo $txt;
  echo $number;
?>
```

14

Variable data types

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

15

Variables

```
$var = “Fall semester";
echo “Hello $var!";

$x = 5;
$y = 4;
echo $x + $y
```

- PHP is a Loosely Typed Language!
- In PHP 7, type declarations were added.
- The var_dump() function returns the data type and the value:

```
$x = 5;
var_dump($x);
```

16

Variables scope

PHP has three different variable scopes:

- local
- global
- static

17

Variables scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside the function.

```
$var = 20; // global scope
function printVar() {
// using var inside this function will produce an error
  echo "<p>Variable var inside function is: $var</p>";
}
printVar();
echo "<p>Variable var outside function is: $var</p>";
```

Output:
Variable var outside function is: 20

18

Variables scope

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Output:
Variable x inside function is: 20

```
function printVar() {
 $var = 20; // local scope
 echo "<p>Variable var inside function is: $var</p>";
}
printVar();
// using var outside the function will produce an error
echo "<p>Variable var outside function is: $var</p>";
```

19

The global keyword

- A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function.
- When you want all your code to be able to access the variable define it as a global variable. Also, some data may be large and complex, and you don't want to keep passing it as arguments to functions.
- To access variables from global scope, add the keyword global.
- Example: **global** $is_logged_in;

20

The global keyword

```php
$x = 12;
$y = 13;
function globalTest() {
 global $x, $y;
 $y = $x + $y;
}
globalTest();
echo $y; // outputs 25
```

In PHP you get nothing (or an error) when referring to a global variable without defining them as global with the global keyword OR using the $GLOBALS.

21

The global keyword

- PHP also stores all global variables in an array called $GLOBALS[*index*].
- The *index* holds the name of the variable.

```
$x = 12;
$y = 13;
function globalsTest() {
    $GLOBALS['y'] = $GLOBALS['x'] +$GLOBALS['y'];
}
globalsTest();
echo $y; // outputs 25
```

22

Static variables

What if you have a local variable inside a function that you don't want any other parts of your code to have access to, but you would also like to keep its value for the next time the function is called? Why?

- Ex: a counter to track how many times a function is called.
- Solution: define the variable as static.

```
<?php
  function test(){
    static $count = 0;
    echo $count;
    $count++;
  }
?>
```

23

Static variables

They can be initialized only with predetermined values

```
<?php
  static $int = 0; // Allowed
  static $int = 1+2; // Correct (as of PHP 5.6)
  static $int = sqrt(144); // Disallowed
?>
```

24

Superglobal variables

- They are predefined variables.
- They contain lots of useful information about the currently running program and its environment.

25

Superglobal name	Contents
$GLOBALS	All variables that are currently defined in the global scope of the script. The variable names are the keys of the array.
$_SERVER	Information such as headers, paths, and locations of scripts. The entries in this array are created by the web server, and there is no guarantee that every web server will provide any or all of these.
$_GET	Variables passed to the current script via the HTTP GET method.
$_POST	Variables passed to the current script via the HTTP POST method.
$_FILES	Items uploaded to the current script via the HTTP POST method.
$_COOKIE	Variables passed to the current script via HTTP cookies.
$_SESSION	Session variables available to the current script.
$_REQUEST	Contents of information passed from the browser; by default, $_GET, $_POST, and $_COOKIE.
$_ENV	Variables passed to the current script via the environment method.

26

Superglobals and security

- They are often used by hackers trying to find exploits to break into your website.
- Always sanitize superglobals before using them. One way to do this is via the PHP *htmlentities* function. It converts all characters into HTML entities.
- Therefore, a much better way to access $_SERVER (and other superglobals) is:

```
$came_from = htmlentities($_SERVER['HTTP_REFERER']);
```

27

htmlentities function

Using the *htmlentities* function for sanitization is an important practice in any circumstance where user or other third-party data is being processed for output, not just with superglobals.

echo and print

- They are both used to output data to the screen.
- echo has no return value while print has a return value of 1 so it can be used in expressions.
- echo can take multiple parameters.
- print can take one argument.
- echo is marginally faster than print.
- They can be used with or without parentheses: echo or echo(), print or print();

Using echo

```
echo "<h2>PHP stands for Hypertext Preprocessor!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
```

PHP stands for Hypertext Preprocessor!
Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

30

Using print

```
print "<h2>PHP stands for Hypertext Preprocessor!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!<br>";
print "This ", "string ", "was ", "made ", "with multiple parameters.";
```

error

PHP stands for Hypertext Preprocessor!
Hello world!
I'm about to learn PHP!
error

31

Using print – single quotes

```
$txt1 "Hypertext Preprocessor!";
$txt2 "Hello world!";
echo ‘<h2>’ . $txt2 . ‘</h2>’;
echo ‘<p> PHP stands for’ . $txt1 . ‘</p>’;
```

Hello world!
PHP stands for Hypertext Preprocessor!

32

Using echo – single quotes

```
$txt1 "Hypertext Preprocessor!";
$txt2 "Hello world!";
print ‘<h2>’ . $txt2 . ‘</h2>’;
print ‘<p> PHP stands for’ . $txt1 . ‘</p>’;
```

Hello world!
PHP stands for Hypertext Preprocessor!

33

PHP Data Types

Variables can store data of different types, and different data types can do different things.

The PHP var_dump() function returns the data type and value.

```
$x = 5985;
var_dump($x);
```

Output:

int(5985)

34

php String

The PHP strlen() function returns the length of a string.

```
echo strlen("Hello world!");
```

Count the number of word in the string "Hello world!":

```
echo str_word_count("Hello world!");
```

Search for the text "world" in the string "Hello world!":

```
echo strpos("Hello world!", "world");
```

The trim() removes any whitespace from the beginning or the end:

```
$x = " Hello World! ";
echo trim($x);
```

35

php String

- Use negative indexes to start the slice from the end of the string:
- Example: Get the 3 characters, starting from the "o" in world (index -5):

```
$x = "Hello World!";
echo substr($x, -5, 3);
Output: orl

$x = "Hello World!";
echo substr($x, 6, 3);
Output: Wor
```

36

Escape Character

- To insert characters that are illegal in a string, use an escape character.

- An escape character is a *backslash* \ followed by the character you want to insert.

37

PHP Constants

Constants are like variables, except that once they are defined they cannot be changed or undefined.

Constants are automatically global and can be used across the entire script.

To create a constant, use the define() function or the const keyword.

38

PHP Constants

Structure:

```
define(name, value);
```

Parameters:

name: Specifies the name of the constant

value: Specifies the value of the constant

Note: Defining case-insensitive constants was deprecated in PHP 7.3. PHP 8.0 accepts only false, the value true will produce a warning.

39

PHP Constants

Example:

```
define(“lang", “PHP");
echo lang;
```

Output:

```
PHP
```

Also, constants can be created using the const *keyword*.

Example

```
const lang = “PHP";
echo lang;
```

40

Keyword const vs. define()

- const variables are always case-sensitive
- const cannot be created inside another block scope, like inside a function or inside an if statement.
- define can be created inside another block scope.

41

PHP Constant Arrays

- From PHP7, you can create an Array constant using the define() function.

Example

```
define("langs", [
  "PHP",
  "Java",
  "C#"
]);
echo langs[0];
```

Output:
PHP

42

Magic Constants

- PHP has nine predefined constants that change value depending on where they are used, and therefor they are called "magic constants".

- These magic constants are written with a double underscore at the start and the end, except for the ClassName::class constant.

43

Magic constants

Constant	Description
__CLASS__	If used inside a class, the class name is returned.
__DIR__	The directory of the file.
__FILE__	The file name including the full path.
__FUNCTION__	If inside a function, the function name is returned.
__LINE__	The current line number.
__METHOD__	If used inside a function that belongs to a class, both class and function name is returned.
__NAMESPACE__	If used inside a namespace, the name of the namespace is returned.
__TRAIT__	If used inside a trait, the trait name is returned.
ClassName::class	Returns the name of the specified class and the name of the namespace, if any.

44

References:

- Nixon, R. (2021) *Learning PHP, MySQL & JAVASCRIPT a step-by-step guide to creating dynamic websites*. S.l.: O'reilly Media. Chapter 2.

- Nixon, R. (2021) *Learning PHP, MySQL & JAVASCRIPT a step-by-step guide to creating dynamic websites*. S.l.: O'reilly Media. Chapter 3.

45