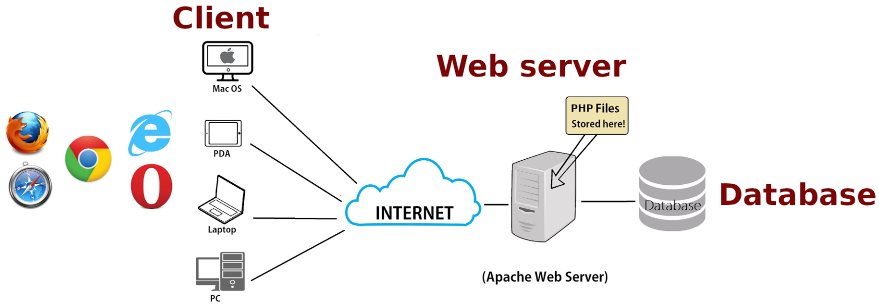
Web application development with PHP

Anis JEMMALI

EPITA Paris

April 2024

WEB network


Client
Mac OS
PDA
Laptop
PC
INTERNET
Web server
PHP Files
Stored here!
(Apache Web Server)
Database
Database

Browsers

- HTML
- JavaScript
- CSS

Installing the development environment

XAMPP is an easy to install Apache distribution containing MariaDB, PHP and more

VS Code is a code editor that supports many languages, extensions, debugging, Git integration and more

Checking the execution environment

- php –version
- php -r 'echo "Hello world";'
- http://localhost
- phpinfo()
- DOCUMENT_ROOT
- php.ini
- httpd.conf

PHP is a server scripting language, and a tool for making dynamic and interactive Web pages.

Insertion of PHP into HTML

The PHP code can be "directly" integrated into the HTML code. It can appear in different places, while being interspersed with HTML code.

```
<html>
  <head>
    <title>This is the title</title>
  </head>
  <body>
    <h1>Heading text</h1>
    <p>
      <?php echo "This␣is␣a␣paragraph."; ?>
    </P>
    <?php echo "<p>This␣is␣a␣second␣paragraph.</p>";
  </body>
</html>
```

Insertion of PHP into HTML

```
<?php $img = 'image1.jpg'; ?>
<html>
 <head>
  <title>This is the title</title>
 </head>
 <body>
  <img src="<?php␣echo␣$img;␣?>"/>
 </body>
</html>
```

Insertion of PHP into HTML

Attention:

Insertion of PHP into HTML

Attention:

- The PHP code must always be surrounded by **<?php** and **?>**

Insertion of PHP into HTML

Attention:

- The PHP code must always be surrounded by **<?php** and **?>**
- The PHP code must always be written in a file whose extension is **.php**

Insertion of PHP into HTML

Attention:

- The PHP code must always be surrounded by **<?php** and **?>**
- The PHP code must always be written in a file whose extension is **.php**
- We should never write PHP in a file whose extension is **.html**

Comments

```php
<?php
/*
comments
on several
lines
*/

//comments one line
?>
```

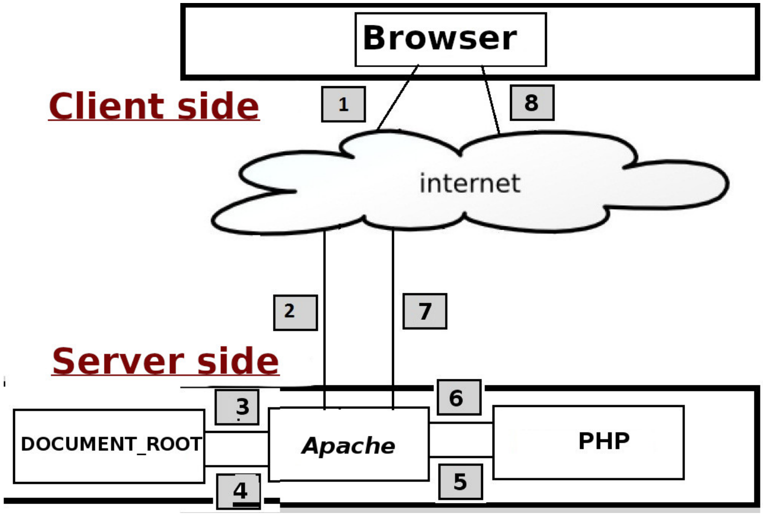
Enchaînement des instructions

PHP instruction must always end with ;.
Line breaks have no influence.

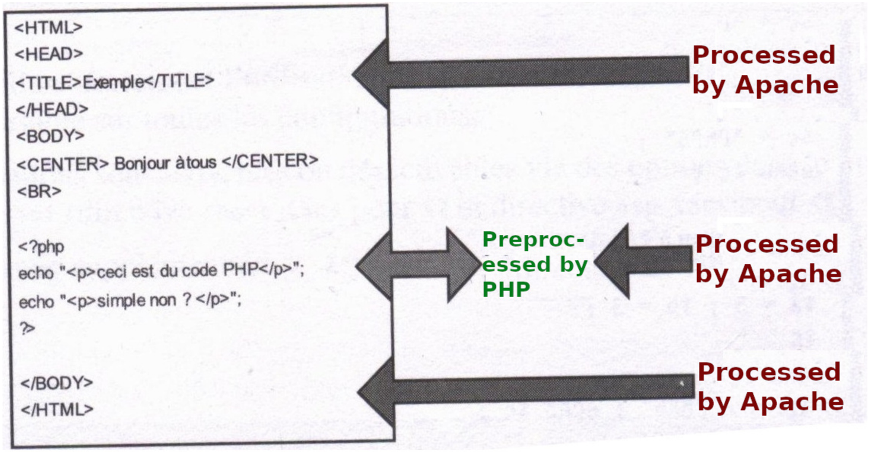
```
<?php
$a = 5;
$b = 3;
$c = "PHP5";
echo $c;

/* equivalent to: */
$a = 5; $b = 3;
$c
=
"PHP5"; echo $c;
?>
```

Client-server interaction



Client-server interaction

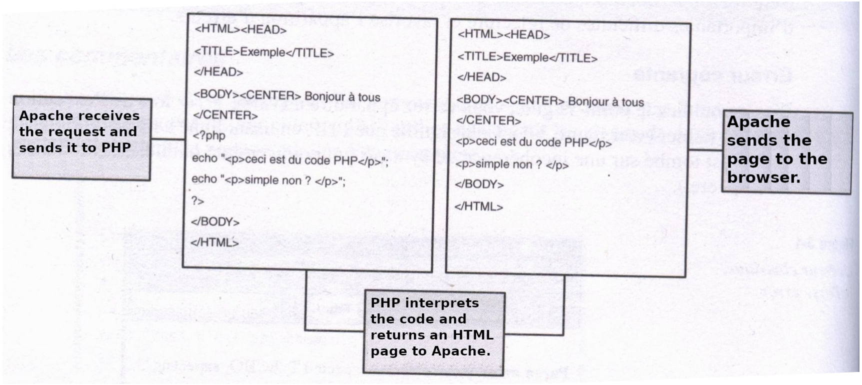


```
<HTML>
<HEAD>
<TITLE>Exemple</TITLE>
</HEAD>
<BODY>
<CENTER> Bonjour àtous </CENTER>
<BR>

<?php
echo "<p>ceci est du code PHP</p>";
echo "<p>simple non ? </p>";
?>

</BODY>
</HTML>
```

Client-server interaction


Apache receives the request and sends it to PHP
PHP interprets the code and returns an HTML page to Apache.
Apache sends the page to the browser.

```
<HTML><HEAD>
<TITLE>Exemple</TITLE>
</HEAD>
<BODY><CENTER> Bonjour à tous
</CENTER>
<?php
echo "<p>ceci est du code PHP</p>";
echo "<p>simple non ? </p>";
?>
</BODY>
</HTML>
```

```
<HTML><HEAD>
<TITLE>Exemple</TITLE>
</HEAD>
<BODY><CENTER> Bonjour à tous
</CENTER>
<p>ceci est du code PHP</p>
<p>simple non ? </p>
</BODY>
</HTML>
```

PHP Variables

In PHP:

[1] For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

PHP Variables

In PHP:

- A variable is created the moment you first assign a value to it. PHP has no command for declaring a variable.

[1] For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

PHP Variables

In PHP:

- A variable is created the moment you first assign a value to it. PHP has no command for declaring a variable.
- A variable starts with the **$** sign, followed by the name of the variable

[1] For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

PHP Variables

In PHP:

- A variable is created the moment you first assign a value to it. PHP has no command for declaring a variable.
- A variable starts with the **$** sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character _

[1]For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

PHP Variables

In PHP:

- A variable is created the moment you first assign a value to it. PHP has no command for declaring a variable.
- A variable starts with the **$** sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character _
- A variable name cannot start with a number

[1] For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

PHP Variables

In PHP:

- A variable is created the moment you first assign a value to it. PHP has no command for declaring a variable.
- A variable starts with the **$** sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character _
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters [1] and underscores (A-z, 0-9, and _)

[1] For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

PHP Variables

In PHP:

- A variable is created the moment you first assign a value to it. PHP has no command for declaring a variable.
- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character _
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters [1] and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

[1] For our purposes here, a letter is a-z, A-Z, and the bytes from 128 through 255 (0x80-0xff)

Exercice:
Among the following variables, which ones have a valid name?

Exercice:

Among the following variables, which ones have a valid name?

- mavar

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1
- $mavar-1

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1
- $mavar-1
- $var5

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1
- $mavar-1
- $var5
- $_mavar

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1
- $mavar-1
- $var5
- $_mavar
- $_5var

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1
- $mavar-1
- $var5
- $_mavar
- $_5var
- $__élément1

Exercice:

Among the following variables, which ones have a valid name?

- mavar
- $mavar
- $mavar 1
- $mavar-1
- $var5
- $_mavar
- $_5var
- $__élément1
- $hotel4*

Output Variables

The PHP echo statement is often used to output data to the screen.

```php
<?php
$txt = "PHP";
echo 'I love '.$txt.'!';
?>
```

To concatenate two strings you can use the . operator

Output Variables

```
<?php
$txt = "PHP";
echo "I␣love␣$txt!<br>";
echo 'I␣love␣$txt!<br>';
?>
```

⚠ Double quotes process special characters, single quotes does not.

Output Variables

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP is a Loosely Typed Language

```php
<?php
$x = 5;
echo $x;

$x = 'Hello!';
echo $x;

echo $x + 7;
?>
```

PHP is a Loosely Typed Language

PHP is a Loosely Typed Language

PHP automatically associates a data type to the variable, depending on its value.

- In PHP 5 you can add a string to an integer without causing an error
- In PHP 7 adding a string to an integer cause a warning
- In PHP 8 adding a string to an integer cause an error

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
We'll learn more about `strict` and `non-strict` requirements, and data type declarations in the PHP Functions chapter.

Get the Type

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

```php
<?php
$x = 5;
var_dump($x);
var_dump("Pierre");
var_dump(3.14);
var_dump(true);
var_dump([2, 3, 56]);
var_dump(NULL);
?>
```

Get the Type

The gettype() function returns the data type [2]

```php
<?php
$data = array(1, 1., NULL, new stdClass, 'foo');

foreach ($data as $value) {
        echo gettype($value), "\n";
}
?>
```

[2] For type checking, use is_* functions like: is_int(), is_float(), is_string(), is_array(), is_object()

Change Data Type

If you assign an integer value to a variable, the type will automatically be an integer.
If you assign a string to the same variable, the type will change to a string:

```php
<?php
$x = 5;
var_dump($x);

$x = "Hello";
var_dump($x);
?>
```

Change Data Type

But what if you want to change the data type of an existing variable, but not by changing the value, you can use casting.

```
<?php
$x = 5;
$x = (string) $x;
var_dump($x);
?>
```

Check existence

`isset()` function determine if a variable is set/declared and is different than null.
`isset($var);` return `TRUE` if `$var` exists and has any value other than null. `FALSE` otherwise
Exercice:

```php
<?php
$x = '';
$y = 'Test';
echo isset($x);
echo isset($y);
echo isset($z);
?>
```

Check content

empty() function check whether a variable is empty. Also check whether the variable is set/declared.
empty() return TRUE if variable does not exist or has a value that is empty or equal to zero. Otherwise returns FALSE

```php
<?php
$x = '';
$y = 'Test';
$z = 0;
echo empty($x);
echo empty($y);
echo empty($z);
echo empty($t);
?>
```

Destroying variables

unset() function destroys the specified variables.

Exemple:

```
<?php
$x = 'Test';
var_dump(isset($x));
unset($x);
var_dump(isset($x));
?>
```

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.
The scope of a variable is the part of the script where the variable can be referenced/used.
PHP has three different variable scopes:

- local
- global
- static

Global Scope

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

```
<?php
$x = 5; // global scope

function myFunction() {
        // using $x inside this function will
        // generate an error
        echo "<p>x␣inside␣the␣function:␣$x</p>";
}
myFunction();
echo "<p>x␣inside␣the␣function:␣$x</p>";
?>
```

Local Scope

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

```php
<?php
function myFunction() {
        $x = 5; // local scope
        var_dump($x);
}
myFunction();
echo '<br>';
// using x outside the function will
// generate an error
var_dump($x);
?>
```

The global Keyword

The global keyword is used to access a global variable from within a function.

```php
<?php
$x = 5;
$y = 10;

function myFunction() {
        global $x, $y;
        $y = $x + $y;
}

myFunction();
echo $y; // outputs 15
?>
```

PHP global variables $GLOBALS[index]

PHP also stores all global variables in an array called
$GLOBALS[index]
The example above can be rewritten like this:

```php
<?php
$x = 5;
$y = 10;

function myFunction() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myFunction();
echo $y; // outputs 15
?>
```

The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

```php
<?php
function myFunction() {
        static $x = 0;
        echo $x.'<br>';
        $x++;
}

myFunction();
myFunction();
myFunction();
?>
```

PHP Constants

A constant is an identifier (name) for a simple value.
The value cannot be changed during the script.

A valid constant name starts with a letter or underscore
(no $ sign before the constant name).

⚠ Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

To create a constant, you can use the `define()` function.

Syntax: `define(name, value, case-insensitive);`

- name: Specifies the name of the constant
- value: Specifies the value of the constant
- case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false.

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

Create a PHP Constant

Create a constant with a case-sensitive name:

```php
<?php
define("GREETING", "Welcome␣to␣EPITA!");
echo GREETING;
?>
```

Create a PHP Constant

Create a constant with a case-insensitive name:

```php
<?php
define("GREETING", "Welcome␣to␣EPITA!", true);
echo greeting;
?>
```

⚠ Do not do that in practice

Create a PHP Constant

You can also create a constant by using the const keyword.

```php
<?php
const MYDOG_NAME = "Speed";
echo MYDOG_NAME;
?>
```

Create a PHP Constant

`const` vs. `define()`

- `const` are always case-sensitive
- `define()` has a case-insensitive option
- `const` cannot be created inside another block scope
- `define()` can be created inside another block scope.

Create a PHP Constant

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

```
<?php
define("LESSONS", [
        "PHP",
        "REACT",
        "Database"
]);
echo LESSONS[0];
?>
```

Constants are Global

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

```php
<?php
define("GREETING", "Welcome␣to␣EPITA!");

function displayGreeting() {
        echo GREETING;
}

displayGreeting();
?>
```

PHP Predefined 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.

- __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.

PHP Predefined Constants

- __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.