Introduction to PHP

What is PHP?

PHP is a scripting language. It is usually used to develop dynamic websites or web applications.
With PHP, you can create login pages with username and password, check the details of a form, create forums, image galleries and more.

PHP is known as a server-side language. This is because PHP is not running on your computer but on the server from which you requested the page. The results are then delivered to you and displayed in your browser. Other scripting languages you may have heard of are ASP, Python, and Perl.

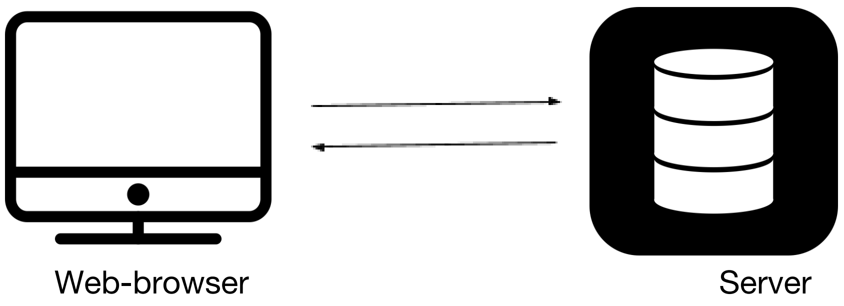
A PHP file contains PHP tags and ends with the ".php" extension.
PHP can be integrated into HTML pages.

How does it works ?

As it says in the previous chapter : ‘PHP is known as a server-side language. This is because PHP is not running on your computer but on the server from which you requested the page.
The results are then delivered to you and displayed in your browser.’

Actually, this is what’s happening behind the scenes :

1. You go to your web browser and type in the address of the website you want to visit (for example ‘facebook.com’).
2. A request to the server is made
3. The server interprete the request and the PHP code of the page.
   So according to the request, it generates the associated view.
4. It respond back to the web browser with the view it generated.
5. Then, the web browser display the view.



How to use PHP

You'll need to know two things :

1. The PHP file must have the '.php' extension
2. Any PHP code must be contained betwen the php tags
   <?php …code inside... **?>**

Installing a local server (wamp)

Before you can write and test your PHP scripts, you'll need a server!
We will use a software called WampServer for Windows.
This allows you to test your PHP scripts on your own computer.
This software installs everything you need.

Variables

What is a variable?

A variable is a storage area. You place objects in your storage areas (variables) so you can use them and manipulate them in your program.
For example, you can store numbers and text.
Think of a variable as a glass containing water. You can add water to the glass, drink everything, refill, etc ...

Another example: Maxime just opened a Skateshop. In his shop, he sells clothes (t-shirt, hats, shoes ....).
To help him in the inventory, Maxime hires 2 people, a woman and a man. They will hold and remember things for him. Man and woman are variables.

Maxime counts how many hats he has, then he gives them to the man.
Maxime counts how many shoes he has and he gives them to the woman.
Unfortunately, Maxime has a bad memory. The question is which of the (variable) people owns the hats and who owns the shoes?
So that Maxime can remember, he can name these people!

He could call them like this:

- mister_hat
- miss_shoe

In truth he can call them as he wants.

- hat_man
- shoe_girl
- hatBoy
- shoeGirl

Rule using a variable

However, There are some rules to follow when creating variables in PHP:

- All variable names must begin with a dollar sign:
  $mister_hat

- PHP is sensitive to the Capital letters. This means that $hat is different from $HAT

- All variable names must start with a letter, eg. $onehat.
  $1hat is not a legal variable name.

- Variable names should not contain any spaces,
  "$sir hat" is not a legal variable name.

- You can instead use an underscore instead of space, eg. $mister_hat.

Good practice

It is best to give your variables explicit names that will help you remember what they contain.

You may be, one day, coming back to old code to change something and you'll be happy to have written coherent and explicit things.

You can write your code (here your variables) as you want, in French, German, English or Scandinavian.
The best is to define upstream the language you will use for your variables or other and stick to it.
The English notation is the best to remember. Indeed, the PHP community is very present in English.

Our two variables will now be called: *$mr_hat* and *$mrs_shoe*

Data type

There are 4 data types for a variable that PHP implicitly manages:

- **INTEGER:** Integers.
- **STRING:** String of characters.
- **BOOLEAN:** A boolean is a variable that has only two possible values: 1 (true) or 0 (false).
- **DOUBLE**: The decimal numbers.

Concrete Example - The Numbers

Let's go back to our Skateshop.
So now that our (variable) employees have a name, they will have to work a little bit. Let's not forget that we are doing the inventory and we need the number of hats and shoes.

There were 20 hats in all.
We are going to hold the hats to the man and tell him that there are 20.
We will therefore "tell" him like this:

- $mr_hat = 20;

Lines of code in PHP need a semicolon at the end.
The name of the variable comes first, then an equal sign. Then what the variable should remember, here the number 20.
Do not forget the dollar before the variable and the semicolon at the end.
These are mistakes that happen often.
The woman is also told to retain the number of shoes (there are 30):

- $mrs_shoe = 30;

Concrete Example - Character Strings

We have just seen how to declare / create variables and assign them numbers.
But you can put text directly into your variables.

Suppose we want to differentiate and catalog our sports shoes, skate, city.
You do it in the same way as to store numbers:

- $shoes1 = "Sport Shoes";

Notice the double quotes around our text.
If you do not surround your direct text with quotation marks, you will get errors.
However, you can use single quotation marks instead of double quotation marks. So you can do this:

- $shoes2 = 'Skate Shoes';

Exercise 1 - Creating the first PHP script

Now is the time to create our first PHP script!

To do this, create a file called 'variable.php'.
It must be created in the folder 'c: // wamp64 / www'.

The file contains this:

```
<?php echo "Hello!"; ?>
```

Let's take a closer look at the contents of this file:

- The PHP tags look like this: ***<? Php ...?>***
  <? Php is the opening tag.
  ?> is the closing tag.
  The content (the code) is between these two tags.
- You can put as much space as you want between the opening and closing tag.
- Thecommand ***echo*** displays a string of characters on the screen.
  In our case, we will display "Hello! ".

Now we rewrite the same thing but with more readability in the code.

```
<? php
   echo "Hello!";
?>
```

It's not a lot of change but spreading your code on more than one line makes it easier to see what you're doing.

Then we add a line of code:

```
<? Php
   $my_var = "Hello!";
   echo "Hello!"
?>
```

Then we modify one last time:

```
<? Php
   $my_var = "Hello!";
   echo $my_var;
?>
```

You have just switched from text to variable!
Then make PHP show the contents of this variable.

Concatenation

In PHP, you can put text end-to-end directly. Or what contains your variables. The point **.** is used for this. Suppose you want to print the following: "The number of hats is 10".

In PHP, you can do this like this:

```
<? Php
   $nb_hats = 10;
   $text = "The number of hats is";
   echo $text. $nb_hats;
?>
```

Here, we define a variable that contains the text to display and a second variable that contains the number of hats. Then we say to display the result of these two variables put end to end. It's the concatenation!

Comments

Why use comments?

Comments help to understand the code.
If you do not work on your code for a while, it's easy to forget what part of the code does.
Commenting on the source code helps to remember what the code does in a specific place.

Commenting is also very important when several developers have to work on the same project.
Changes made by a developer can be easily understood by other developers by simply reading comments.

How to use comments

Comment on a single line ***// (double slash)***
Comment on several lines: ***/ * content * /***

/ * This is an example
**** of comment***
**** /***

Conclusion

- A php script is contained in the <?php ...tag… ?>
- A variable is a storage area.
- Do not forget the semicolon**;**
- Do not confuse HTML / PHP. These are 2 distinct languages.