

Forum:[Hack Forums](#)

Thread: [PHP beginner guide](#)

Remix - November 28, 2015, 8:04 am

Introduction

PHP (Hypertext Preprocessor) is one of the most used web development languages. It's a full back end language which was introduced in 1995. In this guide I want to give you a short insight into the basics of PHP. Notice that you have to put some effort on it - I won't spoon-feed you.

Table of contents:

This subjects will be covered in this guide:

- Getting started
- Basic usage
- Variables
- Arrays
- Conditionals
- Constants

Getting started

Before starting with PHP you should know how it works.

PHP is a back end language which means it's just running on server side and produces front end (client side) code.

For executing PHP code you'll need an interpreter.

The easiest choice for setting up web environments at the beginning would be Apache. But you also could use nginx or other webserver software which supports PHP.

On Windows you should use program "packages" like XAMPP just because they are very easy to use. XAMPP includes Apache and MySQL so you don't have to install it on a more difficult way.

But there are several IDE, like PHPStorm, which only need a PHP executable.

Once you've set up your environment you can start with the basic usage.

Basic usage

PHP has a really simple structure.

Every PHP code has to be inside of the PHP tags `<?php ?>` and the extension of the file which contains PHP code has to be `.php` otherwise it won't be executed and just printed out as clientside

code.

As I said before PHP produces clientside code.
That basically happens with the echo command.
You also can print simple HTML code with PHP.

Here's a short example of how to print HTML code (clientside code) with PHP.

PHP Code

```
<?php
```

```
echo "<p>Hello, world</p>";
```

```
?>
```

With that code a paragraph tag with the text Hello, world will be showed in your browser.
But you don't need to use HTML elements for showing text.

It works on that way too:

PHP Code

```
<?php
```

```
echo "Hello, world";
```

```
?>
```

The result will be the same except the fact that it won't be a paragraph element.

You should notice that the echo command doesn't make a new line by default.
If you want to break the lines you have to use the
 HTML tag on this way:

PHP Code

```
<?php
```

```
echo "line 1<br>";
```

```
echo "line 2";
```

```
?>
```

Variables

In PHP are marked by a dollar sign (\$).

You don't have to type any datatypes when you declare a variable or assign a value to it.

You always have to mark it with the dollar sign doesn't matter if you declare it, use it or what ever.

Here's a simple usage of a variable which will be printed out:

PHP Code

```
<?php
```

```
$var = 5;
```

```
echo $var;
```

?>

If you want to concatenate variables with other variables or string you have to do it with a simple .

An example for concatenating:

PHP Code

```
<?php
```

```
$var = 5;
```

```
echo "Value of the variable: ".$var;
```

?>

Basically there are three ways to concatenate vars with strings:

```
"$var"
```

```
"string ".$var
```

```
"{$var}"
```

Which one you use doesn't matter at all.

If you use the second way it would be a little bit faster but you won't see any differences.

Only if you're using arrays there you can just use two ways. But that will be explained later.

Arrays

Arrays are like tables which store multiple values.

Unlike in languages like C++ or Java you don't have to set a specific datatype or length for the array.

Every value inside of the array has its own index.

It starts, like in other languages, at 0 following by 1, following by 2 and so on.

Since PHP 5.4 you have two ways to declare an array:

```
array()
```

```
[]
```

Both ways does exactly the same so it doesn't matter which way you'll use.

But they are used on the same way:

PHP Code

```
<?php
```

```
$arr1 = array(1,2,3,4);
```

```
$arr2 = [1,2,3,4];
```

In both cases the first index (0) will have the value 1, the second index (1) will have the value 2, the third index (2) will have the value 3 and so on.

You can call the values on that way:

PHP Code

```
<?php
```

```
echo $arr1[0];
```

You can edit them too:

PHP Code

```
<?php
$arr1[0] = 10;
echo $arr[0]
```

If you want to concatenate the value by calling it with the index you have to it on that ways:

```
"string {$arr[0]}"
"string ".$arr1[0]
```

The other way of concatenating won't work.

You can easily add new values to the array just by using `$arr[] = value`

It would look like that:

PHP Code

```
<?php
$arr = [1,2,3,4];
$arr[] = 5;
```

Now the index 4 of the array would contain the value 5.

You can set a "custom" index too.

Then the value will be assigned to the given index.

That would look like this:

PHP Code

```
<?php
$arr = [1,2,3,4];
$arr[5] = 5;
```

Now the index 5 would contain the value 5.

Keep in mind that the index 4 still would be null because you didn't set a value.

Conditionals

Like in every other languages conditionals are used to check whether the statement is true or false.

Conditionals are fundamental for nearly every existing language.

The keywords for conditionals are `if` and `else`.

The basic useage of conditionals looks like that:

PHP Code

```
<?php
if(...) {
    // do something
} else {
```

```
// do something
}
```

Whatever is in the if condition it will check if it's true or false.
If it's true the code block of the if part will be executed.
Otherwise the code block of the else part will be executed.

You're also able to check for several conditions using the keyword elseif

```
PHP Code
<?php
if(...) {
    // do something
} elseif(...) {
    // do something
} else {
    // do something
}
```

First it would check if the first condition is true. If it's false it will check if the condition of the elseif is true and if it is it will execute the code.
If everything is false it will execute the else code block.

Constants

Constants are basically like variables.
It's just the case that you can't edit them.

There are two ways of creating constants:

```
define()
const
```

The main difference between both is that const defines the constant at the compile time and define defines it in the run time.

But there are some more differences:

For example const can't be used to define constants conditionally. If you want to define global constants they have to be declared in the outermost scope:

```
PHP Code
<?php
if(true) {
    const FOO = "BAR";
}
```

You can't use it on that way.
But you can use the define function conditionally:

```
PHP Code
<?php
if(true) {
```

```
define("FOO", "BAR");  
}
```

A common usage is to check whether the constant was already defined:

PHP Code

```
<?php  
if(!defined("FOO")) {  
    define("FOO", "BAR");  
}
```

Furthermore const is just able to take plain names.
define accepts every expression as name.
That's an advantage if you want to do something like that:

PHP Code

```
<?php  
$i = 3; // or any other value  
define("d_". $i, 1);
```

Another difference is that const is always case sensitive (lower and upper case letters matter) whereas define allows to define it by passing a boolean as third argument:

PHP Code

```
<?php  
define("FOO", "BAR", true);  
echo FOO; // valid  
echo foo; // valid
```

Closing words

Thanks for reading this guide.
I hope it was useful for you.
If you have any suggestions feel free to hit me up.

Greetings

Remix ?

stackoverflowin - November 28, 2015, 12:52 pm

Good job, Remix!

For this though:

PHP Code

```
<?php  
echo "line 1<br>";
```

I'd of have put

PHP Code

```
<?php
```

```
echo "line 1\n";
```

That's just my personal choice, though.

Nosymbols - November 28, 2015, 1:12 pm

(Nov 28, 2015 01:52 PM)StackOverlowin Wrote: [\[CITE\]](#) Good job, Remix!

For this though:

PHP Code

```
<?php  
echo "line 1<br>";
```

I'd of have put

PHP Code

```
<?php  
echo "line 1\n";
```

That's just my personal choice, though.

That would output

Code Line 1 Line 2

whereas using `<br/>` would output

Code Line 1

Line 2

Unless of course you're running scripts on the command line, then `\n` would be required in place of `<br/>`

(Reply is based on a test in Chrome only.

FF and IE may differ)

stackoverflowin - November 28, 2015, 1:27 pm

(Nov 28, 2015 02:12 PM)Nosymbols Wrote: [\[CITE\]](#) -snip-

I know, but for CLI based stuff. A lot of PHP stuff is CLI, so yeah - it's not all web y'know.

thesonandheir - November 28, 2015, 8:53 pm

It is generally frowned upon to use closing PHP tags. Since they're essentially useless, it's better ([and safer](#)) to eliminate these from your code.

<https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md> - some recommend reading.

(Nov 28, 2015 02:27 PM)StackOverlowin Wrote: [\[CITE\]](#) I know, but for CLI based stuff. A lot of PHP stuff is CLI, so yeah - it's not all web y'know.

<http://php.net/usage.php>

ravesh - November 28, 2015, 10:58 pm

Thanks for the guide. Its very easy and handy.

stackoverflowin - November 29, 2015, 4:38 am

(Nov 28, 2015 09:53 PM)TeamViewer Wrote: [\[CITE\]](#) It is generally frowned upon to use closing PHP tags. Since they're essentially useless, it's better ([and safer](#)) to eliminate these from your code.

<https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md> - some recommend reading.

<http://php.net/usage.php>

Sorry but what's that meant to mean? Obviously, PHP is used more for the web than console based stuff.

We recommend you take the time to read the [HF Help Documents](#) section. The answer to your questions should be contained in there.

Vellichor - November 29, 2015, 7:06 am

Zend standard disallows closing your PHP tags:

framework.zend.com/manual/1.12/en/coding-standard.php-file-formatting.html

and "{\$var}" isn't concentrating strings, it's escaping strings and shouldn't honestly be used like that.

MineSQL - November 29, 2015, 3:41 pm

(Nov 29, 2015 08:06 AM)Vellichor Wrote: [\[CITE\]](#) Zend standard disallows closing your PHP tags:

framework.zend.com/manual/1.12/en/coding-standard.php-file-formatting.html

and "{\$var}" isn't concentrating strings, it's escaping strings and shouldn't honestly be used like that.

That is not escaping a string... The curly braces are used to insert variables into string with "" (those types of quotations) it is useful if you have an array value you want to echo and don't want to confuse the code interpreter with the complex variable you are trying to insert directly into a string.

Anyway, OP you should work on your grammar and try to expand on important topics like conditional statements.

BoliBerrys - January 16, 2016, 11:53 pm

Pretty detailed and easy to follow.

Thanks alot for this :D

noname1 - January 17, 2016, 3:50 am

History

PHP originally stood for "Personal Home Page" as it was created in 1994 by Rasmus Lerdorf to track visitors to his online resume. As its usefulness and capabilities grew (and it started being used in more professional situations) it came to mean "PHP: Hypertext Preprocessor".

Quote from my book by Larry Ullman

Kedox - January 17, 2016, 5:05 am

You should mention using `<?=$var?>` since using it when working with HTML is so much easier.

And as Stack said add that if using CLI use `/n` instead of `</br>`

Sakuto - January 18, 2016, 6:13 am

(Jan 17, 2016 06:05 AM)theklaus101 Wrote: [\[CITE\]](#) You should mention using `<?=$var?>` since using it when working with HTML is so much easier.

And as Stack said add that if using CLI use `/n` instead of `</br>`

Better go for `PHP_EOL` instead, with this you'll support cross-platform scripting.