

Forum:[Hack Forums](#)

Thread: [\[The Empire\] An Introduction to PHP](#)

1n9i9c7om - May 28, 2014, 6:39 pm

The Basics

What is PHP? An Introduction

Setting up your development environment

An easy start - your first PHP page

An introduction into variables

What if..?

Let's do the math

Array's

I GET your POST

Loops

Yay, cookies! And Sessions

Using a database

A bad example - how to NOT do it, and WHY!

A good example - how to do it, and why Prevent XSS vulnerabilities on your website Example

A simple login system

Tips

Continue learning

"PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML." - [php.net](#)

Originally PHP stood for Personal Home Page Tools, but this changed when PHP 3 got released. Now PHP stands for PHP: Hypertext Preprocessor.

In this tutorial, you'll learn the basics of PHP. You'll also learn how to work with databases and how to make your script secure.

I'll include some example at the end of this tutorial, and at last I'll give you tips for further learning.

If you've already Apache, PHP and MySQL installed OR if you got hosting, you can skip this step.

For our development environment we'll use [EasyPHP](#). It comes with Apache, PHP and MySQL - that's everything we need.

If you're on Windows XP or Vista, you'll have to download [this](#), on Windows 7, 8 and 8.1 you can use [this](#).

When you downloaded it, you obviously have to start the installer, so let's do it. I won't explain how to install it in-depth, as it is like every other installer.

When the installation is done, untick "Open Help", make sure "Launch EasyPHP" is ticked and click on "Finish".

You should now see the EasyPHP icon in your tray. Double-click it and you'll see this: [\(Click to View\)](#)

Note: If you're getting [this](#) error, log out from Skype, start Apache and log in in Skype again. It'll work then.

The root directory if your website is here: C:\Program Files
(x86)\EasyPHP-DevServer-13.1VC11\data\localweb

That means that you have to save your .php files here or in any subfolder.

(Note: This depends on where you installed EasyPHP. If you changed the path during the installation, look there).

We'll need phpMyAdmin to make it easier to work with databases. Go to the EasyPHP folder, open the modules folder and copy the phpMyAdmin folder. Paste it in the localweb folder and rename it to phpMyAdmin. It should [look like this](#) now.

Congratulations! You installed the testing environment, we are now able to start!

Now, after you installed your test environment, we can start.

Coding via Notepad is bad - you should get [Sublime Text](#) or [Notepad++](#).

This is a basic HTML page, nothing special, just the most important stuff:

PHP Code

```
<?php
<!DOCTYPE html>
<html>
<head>
  <title>
    A basic HTML site
  </title>
</head>
<body>
  <h1>Basic HTML Site</h1>
  <p>This is a basic HTML site.</p>
</body>
</html>
```

If you want, you can check how it looks like by saving it as basic.html, then move it into the root folder of your webserver and open it by entering <http://127.0.0.1/basic.html> (or, if you're using hosting, <http://yourdomain.com/basic.html>).

Now we add some PHP into this code:

PHP Code

```
<?php
<!DOCTYPE html>
<html>
<head>
  <title>
    Your first PHP script
  </title>
</head>
<body>
  <h1>First PHP script</h1>
  <p><?php echo "This is your first PHP script."; ?></p>
</body>
</html>
```

Save it as first.php and run it by opening it in your browser. It should look [like this](#).

[echo](#) is a language construct that is used to output strings.

Another language construct to output strings is [print](#). However, I'll most likely use echo in this guide.

Variables are like 'containers' - they contain values.

There are different types of variables:

Strings

Integers

Floats

Booleans

Arrays (we'll talk about them later)

...

[Here is full a list of variables types in PHP](#)

Strings basically contains text. In our first PHP page we have this line:

PHP Code

```
<?php
  <p><?php echo "This is your first PHP script."; ?></p>
```

The text in the quotes is a string.

There are four ways to declare strings in PHP. They're very well explained on PHP.net, so I think it'd be the best to just link you to it.

[Here is the explanation.](#)

If you're using a 32-bit build of PHP, integers can be from 2147483648 to 2147483647, on a 64-bit build it's from -9223372036854775808 to 9223372036854775807

An integer can only store round numbers, every number containing a decimal mark will be a float. The maximum of a float is platform-dependent.

Booleans can either contain true or false.

To assign a value to a variable, you just do this:

PHP Code

```
<?php
$string = "Hello World!";
$int = 1;
$float = 1.0214;
$bool = true;
```

For further information about variables and types, check out [this](#) and [this](#).

If you're working with PHP, you'll need to use conditional statements.

With these conditional statements you can, for example, check whether a user is logged in or not.

Basic if conditions works like this:

PHP Code

```
<?php
if($loggedIn)
{
  echo 'This will only be executed when the $loggedIn boolean is true';
}
```

You can check multiple conditions by using && (and) and || (or).

PHP Code

```
<?php
if($loggedIn && $canPost)
{
    echo "This will only be executed if both booleans are true";
}
```

PHP Code

```
<?php
if($i < 0 || $i > 10) {
    echo "This will only be executed if $i is between 0 and 10";
}
```

You can use == to check whether a variable is like a variable/string/integer/float:

PHP Code

```
<?php
if($userId == 1)
{
    echo "This will only be executed if the userId variable equals 1!";
}
if($username == "1n9i9c7om")
{
    echo "This will only be executed if the username is 1n9i9c7om.";
}
```

However, == will just check the value, not the type. For example, this will be executed:

PHP Code

```
<?php
if("0" == 0)
{
    echo "'0' is like 0";
}
```

If you also want to check the type, you'll have to use ===. This won't work as 0 is an integer while "0" is a string:

PHP Code

```
<?php
if("0" === 0)
{
    echo "This will never be executed.";
}
```

If you have to compare one variable with multiple results, you can use a switch statement instead of many loops:

PHP Code

```
<?php
switch ($i) {
    case 0:
        echo "i = 0";
}
```

```

        break;
    case 1:
        echo "i = 1";
        break;
    case 2:
        echo "i = 2";
        break;
}

```

This also works with strings and any other types:

PHP Code

```

<?php
switch ($language) {
    case "php":
        echo "The used language is php";
        break;
    case "c":
        echo "The used language is c";
        break;
    case "vb":
        echo "The used language is vb";
        break;
}

```

There are many things you can do with numbers in PHP - these are the simplest:

PHP Code

```

<?php
$result = 1 + 1; //simple addition, $result will be 2
$result = 1 - 1; //simple subtraction, $result will be 0
$result = 6 / 2; //simple division, $result will be 3
$result = 4 * 2; //simple multiplication, $result will be 8
$result = 8 % 3; //modulus, will return the remainder of 8 divided by 3, so 2.

```

//This can be used to check whether we have an even or an odd number:

```

if($i % 2 == 0)
{
    echo "Number is even.";
}
else
{
    echo "Number is odd.";
}

```

You can find out more about this [here](#).

Beside this arithmetic operators, there are other useful functions to work with numbers as well. I'm not going to give you example of all of them, but here's a quick list of the most useful ones:

[round](#) - Used to round a number.

[ceil](#) - Used to round up a number.

[floor](#) - Used to round down a number.

[rand](#) and [mt_rand](#) - Used to generate random numbers.

If you're familiar with any other coding language, you should know what an array is, if not: Arrays associate values with keys. This might sound difficult at first, but here's an example:

PHP Code

```
<?php
$pluginInfo= array(
    //key => value
    "name"      => "Easy Refer",
    "description" =>
"Creates an invitation form which adds the members referral code. Also shows the users referrals.",
    "website"   => "http://www.mybbcentral.com",
    "author"    => "Jesse Omni",
    "authorsite" => "http://www.mybbcentral.com",
    "version"   => "2.0",
    "guid"      => "",
    "compatibility" => "16*"
);
```

The array values are actually from Omni's "Easy Refer"-plugin.

So, now you have an array, you can easily extract information from it by doing this:

PHP Code

```
<?php
echo $pluginInfo["name"];
echo $pluginInfo["description"];
```

It's quiet easy, isn't it? If you don't specify a key, it'll numerate the values.

For example, an array specified like this:

PHP Code

```
<?php
$ourArray = array("Easy Refer", "Jesse Omni");
will actually work like that:
```

PHP Code

```
<?php
echo $ourArray[0]; //"Easy Refer"
echo $ourArray[1]; //"Jesse Omni"
```

Keep in mind that it'll start counting with 0, not with 1!

You can set a key any time, but keep in mind that if it's an number, PHP will continue counting from this number, so if you have an array with 3 values and set the key to 5, the value added after it will have the key 6, if not specified different.

The value of an array can be anything, even another array. That'd look similar to this:

PHP Code

```
<?php
$memberInfo = array("1n9i9c7om" => array("uid" => 1296484), "Tibit" => array("uid" => 651337));
$memberInfo["1n9i9c7om"]["uid"]; //returns 1296484
$memberInfo["Tibit"]["uid"]; //return 651337
```

Additional info's about arrays can be found on the official PHP.net page [here](#)

GET and POST are two methods to transmit form-data from one page to another. This might sound complex now, but it'll be something you'll be using on a regularly basis if you're going to use PHP for

your projects. It's much easier than it sounds!

First, you'll have to create a HTML form. It looks like this:

PHP Code

```
<?php
```

```
<form method="" action=""></form>
```

If you leave method empty, it'll use GET, as that's the default defined option. You can also set it to POST. action is used to tell the browser to which site the data is going to be sent. If it's left empty (or not declared at all), it'll simply send it to the page you're on right now (what can be quite helpful sometimes).

A GET request will append the data to the URL (member.php?action=profile&uid=969003) while POST data will be sent in the headers.

The max. length for GET data is [limited](#). Of course the max. length for POST data is limited as well, but it's much larger (depending on the php.ini).

Now, we have to add some 'input's to your form. This is also easy HTML:

PHP Code

```
<?php
```

```
<form method="POST">
```

```
  <input type="text" name="username"><br/>
```

```
  <input type="submit" name="submit" value="Submit"><br/>
```

```
</form>
```

We've added a textbox and a submit-button to our form. You can also add checkboxes, radioboxes, dropdown-lists and other things.

It's important that we add a name to these HTML objects, as we will get their values by using their names:

PHP Code

```
<?php
```

```
if(isset($_POST['submit']))
```

```
//make sure this code is only being executed when the submit-button has been pressed
```

```
{
```

```
  $username = $_POST['username'];
```

```
//get the value from $_POST['username'] and save it in a variable called $username
```

```
}
```

If you're using GET, it'd be the same, except that you use \$_GET instead of \$_POST.

1n9i9c7om - May 28, 2014, 6:48 pm

Loops are made to repeat a specific code block a given number of times, or until a condition is met.

There are four different kinds of loops in PHP:

The for-loop

The foreach-loop

The while-loop

The do...while-loop

The for-loop is used when you know how many times you want your code to be repeated, its syntax looks like this:

PHP Code

```
<?php
```

```
/* Syntax
```

```
for (initialization; condition; increment)
```

```
{
```

```
  code to be executed;
```

```
}  
*/
```

//Actual example:

```
for ($i=1; $i <= 10; $i++)  
{  
    echo "Current number is $i, still " . 10 - $i . " left. <br/>";  
}
```

This will set the value of the variable "\$i" to 1 and execute the loop until \$i is higher than or equals 10. It'll increment by 1 (\$i++) every time the code in the row gets executed.

A foreach-loop will iterate over elements in an array. There are two versions of the foreach-loop - one that only gives you all values and one that also gives you the key.

Here's an example for both of them:

PHP Code

```
<?php  
$memberInfo = array("username" => "1n9i9c7om", "uid" = 1296484);  
  
foreach ($memberInfo as $value) {  
    echo $value . "<br/>"; //this will give out 1n9i9c7om and 1296484  
}  
foreach ($memberInfo as $key => $value) {  
    echo "His $key is $value. <br/>";  
}
```

/* Output of this one will be:

His username is 1n9i9c7om.

His uid is 1296484. */

You can name \$value whatever you want, you just specify the variables name to access it inside the loop.

Let's move on to the while and do...while loops. These loops will run until a condition is met.

They're pretty similar, except that in the while loop the condition will be checked before the code block runs, in the do...while loop it runs after that. That means that a "do...while" loop will run at least once.

PHP Code

```
<?php  
//This will give out all numbers from 1 to 10:  
$i=1;  
while ($i <= 10) { // Output values from 0 to 10  
    echo "The number is ".$i."<br />";  
    $i++;  
}  
//This will give out nothing:  
$i=1;  
while ($i <= 10) { // Output values from 0 to 10  
    echo "The number is ".$i."<br />";  
    $i++;  
}
```

//This will also give out the numbers from 1 to 10:

```
$i = 1;
do {
    echo "The number is ".$i."<br/>";
    $i++;
}
while ($i <= 10);
```

//This will echo out the number 11, as it will check whether it should exist the loop or not after the code being executed:

```
$i = 11;
do {
    echo "The number is ".$i."<br/>";
    $i++;
}
while ($i <= 10);
```

You can always end a loop by using the break; statement:

PHP Code

```
<?php
foreach ($memberInfo as $value) {
    if($value == $neededValue)
    {
        break;
    }
}
```

Everybody likes cookies, right? Well, yeah, they're tasty. :3

You can use cookies in PHP to store information on the users browser instead of your server, but keep in mind that this information can be easily manipulated.

Cookies have to be set before the headers (therefore also before any HTML) are sent out to the user.

This is how you use cookies:

PHP Code

```
<?php
setcookie("TutorialCookie", "I hope you learn something here!", time() + 3600);
$cookieValue = $_COOKIE["TutorialCookie"];
//hey, $_COOKIE is an array! $cookieValue equals "I hope you learn something here!" now.
```

This is how you have to use cookies most times: You set a name for the cookie, set the value of it and define when it'll expire.

time() + 3600 will make the cookie available for one hour. If you don't set any time at all (it's optional, setcookie("TutorialCookie", "I hope you learn..."); will work as well) the cookie will be available until the user restarts his browser.

You can also set the path, the domain, whether it should be sent via HTTPS only and whether it's usable in Javascript.

More information about the [setcookie\(\)](#) function are available [here](#).

That sounds nice, doesn't it? But remember how I said that the data saved in cookies can be manipulated easily? Well, there's an alternative to use cookies, called sessions. Sessions will create

a cookie as well, but it'll only save to session id in it. The data is still saved on your server.

To use sessions, you have to use [session_start\(\)](#) before the headers get send (therefore also before you put out any HTML).

After that, you can easily add/edit/remove session values by modifying an array called `$_SESSION`.

PHP Code

```
<?php
session_start();
$_SESSION['userid'] = 1160594;
$_SESSION['username'] = "vitafortis";
$userid = $_SESSION['userid'];
```

Sessions are fairly easy to use and are often used to check whether a user is logged in or something similar to this.

How long a session lasts is defined in the `php.ini`, the default is 24 minutes, but this can be changed by yourself or your host.

To remove a session, in example to log out an user, just call the function [session_destroy\(\)](#).

Note: I'm not going to explain all the SQL queries here, as this tutorial isn't about learning SQL, but using it.

Using a database will be quiet important when working with PHP. You are going to store many information in them, for example user details or whatever.

It's also important to make it secure, as you don't want the 'bad guys' to get access to your database, do you? ;)

That's why I'll start with a bad example.

PHP Code

```
<?php
mysql_query("SELECT COUNT(id) FROM users WHERE username = '$name' AND password = '$password'");
```

This is a bad example for several reasons:It uses the old `mysql_*` functions

It's not secured against SQL injections

It doesn't escape characters

An attacker could use something like this as his name: ' OR 1=1 /*

This would make the query look like this: `SELECT COUNT(id) FROM users WHERE username = " OR 1=1 /*' AND password = '$password'`

Everything after the `/*` is ignored, because it's commented out. If you'd log-in somebody based on this query, they could easily gain access to any account as long as they know the usernames.

Now let's start with the good example. The `mysql_*` extension for PHP is deprecated, so you should use `mysqli` or `PDO` instead. I'll use `PDO` for this tutorial as it also supports other database types than `MySQL`.

One big advantage of `mysqli` and `PDO` are prepared statements, which will help you to prevent SQL injections.

To connect to a `MySQL` Database using `PDO`, use the following snippet:

PHP Code

```
<?php
$dbc = new PDO('mysql:dbname=DATABASE;host=HOST;charset=utf8', 'USERNAME',
'PASSWORD');
$dbc->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//this is using for debugging applications
//you should remove this line when making a project public
```

Replace DATABASE with the name of the database you want to use, HOST with the host (e.g. 127.0.0.1). I don't have to explain USERNAME and PASSWORD, do I?

I'll explain two types of queries here: These that return something (SELECT) and these that don't (INSERT, UPDATE, DELETE).

We'll be using prepared statements to secure our application, giving you an example is easier than explaining it, so I'll simply start here:

PHP Code

```
<?php
$exampleQuery = $dbc->prepare("UPDATE users SET email = :email WHERE uid = :uid");
$exampleQuery->bindParam(":email", $_POST['newemail']);
$exampleQuery->bindParam(":uid", $_SESSION['uid'], PDO::PARAM_INT);
if($exampleQuery->execute())
{
    //query successfully executed
}
else
{
    //something's wrong with your query
}
```

You see how we used :email and :uid as placeholders? We 'replace' them with their real values using bindParam(), which accepts 2 or 3 arguments.

The first one is the name of the placeholder (":email"), the second one is the variable containing the value you want to add in your query. The third one is optional, but you should use it if you're not adding a string. It declares the type of the variable, the default is string. PDO::PARAM_INT for example tells PDO to add it as a integer. [Here](#)'s a list of all PDO constants.

So far so good, but what if we need to values from a SELECT query? Well, that's easy as well. I'll simply start with an example, it should be self-explaining:

PHP Code

```
<?php
$selectQuery = $dbc->prepare("SELECT uid FROM users WHERE ip_address = :ip_address");
$selectQuery->bindParam(":ip_address", $user['ip']);
$selectQuery->execute();
$selectQueryResults = $selectQuery->fetchAll();
echo "These are the UIDs of all users sharing the same IP-address:"
foreach($selectQueryResults as $row)
//This will make sure that the code will run for every row selected
{
    $row['ip_address'] . "<br/>";
//$row is an array containing all rows (useful when selecting the value of multiple columns). Use $row[columnname] to get the value of the column.
}
```

It's quite easier when you only need to select one column:

PHP Code

```
<?php
$usernameExistsQuery = $dbc->prepare(
"SELECT COUNT(id) FROM users WHERE username = :username");
$usernameExistsQuery->bindParam(":username", $_POST['username']);
$usernameExistsQuery->execute();
$usernameExists = $usernameExistsQuery->fetchColumn() > 0
//Old version using a ternary operator, basically like a shorthand if:
$usernameExists = ($usernameExistsQuery->fetchColumn() > 0 ? true : false);
```

I used an shorthand syntax, so I don't have to use the whole if(){ } thingy. More information about it are available [here](#).

A more in-depth tutorial about PDO by me can be found [here](#).

As we're talking about security right now, I thought it'd be good to tell you how to prevent XSS vulnerabilities on your website in this little part of this tutorial.

A XSS vulnerability is basically when an attacker tries to inject HTML/Javascript code on your website, for example by making an account and calling it `<script>alert("XSS");</script>`.

To prevent that, simple call `htmlspecialchars()` whenever you're putting out userinput.

PHP Code

```
<?php
echo htmlspecialchars($_POST['message']);
```

That's it already! Easy, huh?

I thought making a small example script would be a great way to show you how it's done, you should be able to understand the code if you understood this tutorial.

[Login Example](#)

It contains all PHP files and the database (.sql) which you can import using PHPMysqlAdmin.

Keep in mind that this is really basic and doesn't prevent mass-registrations or brute-force-attacks (captchas are easy to integrate though, check out re-captcha). You could also check out this guys login examples: <https://github.com/panique/>

As an idea for your first project, why don't you try making a login system which you could use for a real website?

The best way to continue learning PHP is by coding something you want to create. Whenever you're stuck, try to find help using Google or ask around here on HF (or even on this thread!). This way, you'll learn a lot about the language.

Good sources to find information are php.net and StackOverflow.

You could also take a look at some open source projects.

Thanks for reading my tutorial, I really hope it helped you to learn some PHP.

This is just the beginning of learning PHP - there are still many things to do, for example classes and objects!

Good luck, and if you have any questions, you know where you can get help now :)

1n9i9c7om

King Cin - May 29, 2014, 6:23 am

Huge, and well written thread. I will save this to read all of it on my 7 hour bus trip to France next week.

Thanks.

al_gharib - May 29, 2014, 9:17 am

Hi,

Thank you for your tutorial beginner. Simple and easy to understand.

Sommmen - May 29, 2014, 10:14 am

(May 29, 2014 06:23 AM)Unknown † Wrote: [\[CITE\]](#) Huge, and well written thread. I will save this to read all of it on my 7 hour bus trip to France next week.

Thanks.

this comment in combination with your profile pic. is absolutely hilarious XD

Neeeeeeeeeeel.- - May 29, 2014, 10:46 am

Good tutorial for newbies! Congratulations.

1n9i9c7om Wrote:To use sessions, you have to use session_start() before any HTML gets outputted.Also, you should use it before sending any header.

1n9i9c7om - May 29, 2014, 11:50 am

(May 29, 2014 10:46 AM)Neeeeeeel.- Wrote: [\[CITE\]](#) Also, you should use it before sending any header.

Oh right, I should include that as that's the reason why you have to do it before sending out HTML as well. Thanks!

Kodra - May 29, 2014, 12:20 pm

Thanks alot for sharing this with the community.

Should be a great help for all newbies who want to get started with PHP.

Fox™ - May 29, 2014, 1:05 pm

Nice tutorial babe. All looks great.

Nico is very sexy, and he and I have a lengthy knowledge on PHP.

jQuery - May 29, 2014, 2:36 pm

Good tutorial. One thing I have to point out though is the following line of code

PHP Code

```
<?php
```

```
$usernameExists = ($usernameExistsQuery->fetchColumn() > 0 ? true : false);
```

I understand you're trying to introduce the ternary operator but that's a very poor example. A more efficient way to write that line would be

PHP Code

```
<?php
```

```
$usernameExists = $usernameExistsQuery->fetchColumn() > 0;
```

1n9i9c7om - May 29, 2014, 2:52 pm

(May 29, 2014 02:36 PM)jQuery Wrote: [\[CITE\]](#) Good tutorial. One thing I have to point out though is the following line of code

PHP Code

```
<?php
```

```
$usernameExists = ($usernameExistsQuery->fetchColumn() > 0 ? true : false);
```

I understand you're trying to introduce the ternary operator but that's a very poor example. A more efficient way to write that line would be

PHP Code

```
<?php
```

```
$usernameExists = $usernameExistsQuery->fetchColumn() > 0;
```

Whops, yep, you're right. I wrote this when I was pretty tired, thanks for correcting that.

I commented the old version out, so the readers will at least see the ternary operator once. :P

VipVince - May 29, 2014, 4:03 pm

Very well written, love how you designed it. And covers most of the fundamentals. GJ!

Sommmen - May 30, 2014, 12:48 am

just like to point out there that easy php also gives an error if you didn't install the right redistributabublal pack...

Badge - May 30, 2014, 3:58 am

really like this, going to bookmark it so i can view it again later ?

SomeWhiteGuy - May 30, 2014, 10:09 am

//This will give out all numbers from 1 to 10:

```
$i=1;
```

```
while ($i <= 10) { // Output values from 0 to 10
```

```
    echo "The number is ".$i."<br />";
```

```
    $i++;
```

```
}
```

Will give out all numbers up to 11. You're checking if \$i is less than OR equal to 10. If it equals 10 the code will be executed again until it is above 10 (because then \$i would not be less than or equal to 10), stopping at 11 in this case.

1n9i9c7om - May 30, 2014, 12:09 pm

(May 30, 2014 10:09 AM)SomeWhiteGuy Wrote: [\[CITE\]](#) //This will give out all numbers from 1 to 10:

```
$i=1;
```

```
while ($i <= 10) { // Output values from 0 to 10
```

```
    echo "The number is ".$i."<br />";
```

```
    $i++;
```

```
}
```

Will give out all numbers up to 11. You're checking if \$i is less than OR equal to 10. If it equals 10 the code will be executed again until it is above 10 (because then \$i would not be less than or equal to 10), stopping at 11 in this case.

Run the code yourself, it'll display them from 1 to 10.

It checks before actually running the code, and if \$i is higher than 10, the code won't be executed, therefore 11 isn't being displayed.

Spoiler ([Click to View](#))

AngularJS - May 30, 2014, 3:26 pm

Thanks a lot for this tutorial, have been looking for this.

SomeWhiteGuy - May 30, 2014, 6:04 pm

(May 30, 2014 12:09 PM)1n9i9c7om Wrote: [\[CITE\]](#) Run the code yourself, it'll display them from 1 to 10.

It checks before actually running the code, and if \$i is higher than 10, the code won't be executed, therefore 11 isn't being displayed.

Spoiler ([Click to View](#))

hurrdurr I explained it to myself and still got it wrong.

Nice tutorial, none-the-less. ;)

Canonsx130 - May 30, 2014, 7:03 pm

This is a great thread Nico nigga. Keep going strong :)

ASDA - May 31, 2014, 12:46 pm

Under maths operations you forgot

Disease - May 31, 2014, 6:13 pm

(May 31, 2014 12:46 PM)ASDA Wrote: [\[CITE\]](#) Under maths operations you forgot **

That operator is a feature of PHP 5.6. It will be quite a while before PHP 5.6 is commonly supported across hosts. As such, the OP leaving that operator out is not only understandable, but encouraged. He would likely run into issues of people getting errors for using that operator because their host is still on 5.4.x (or 5.5.x if they're lucky).

Hexley - June 2, 2014, 5:05 pm

Great tutorial for beginners OP, but the "or" statement was missed, and that is a great (and underused) statement..

```
Code if($Number < 0 || $Number > 10) {
```

```
    echo "The variable has a variable that is not allowed, you typed in " + htmlentities($Number) . ".";
}Rennet, in double quotes echo, you can reference a variable with brackets. {$Number}
```

Imperious1 - June 2, 2014, 5:06 pm

Looks like a nice thread. This will definitely help beginners a lot.

TRH4CK - June 2, 2014, 9:03 pm

So you're telling that if I use PDO, I would not have any XSS problems ? And would that be good for SQLInjections aswell :D ?

Btw, nice post Nico <3

Neeeeeeeeeeel.- - June 2, 2014, 11:21 pm

(Jun 2, 2014 09:03 PM)TRH4CK Wrote: [\[CITE\]](#) So you're telling that if I use PDO, I would not have any XSS problems ? And would that be good for SQLInjections aswell :D ?

Btw, nice post Nico <3Even using PDO you can have XSS problems.

TRH4CK - June 3, 2014, 7:26 am

(Jun 2, 2014 11:21 PM)Neeeeeeel.- Wrote: [\[CITE\]](#) Even using PDO you can have XSS problems. Not in outputting, I mean, possible to do xss or sqlinection during input to db?

POLY MATH - June 3, 2014, 10:01 am

Great Tutorial. Needs more HQ post like this across the forums.

TrojanPoem - June 4, 2014, 1:47 pm

Thanks for this nice tutorial. I really learnt a lot.

VipVince - June 5, 2014, 7:13 am

(Jun 3, 2014 07:26 AM)TRH4CK Wrote: [\[CITE\]](#) Not in outputting, I mean, possible to do xss or sqlinjection during input to db?

Yes. Consider a guestbook; non sanitized input being stored in the DB will result in a persistent XSS being displayed on the webpage.

Logs? - June 5, 2014, 9:55 am

This is a great guide. Very informative and extremely helpful for the people who want to learn how to code in PHP.

1n9i9c7om - June 6, 2014, 5:54 am

(Jun 2, 2014 09:03 PM)TRH4CK Wrote: [\[CITE\]](#) So you're telling that if I use PDO, I would not have any XSS problems ? And would that be good for SQLInjections aswell :D ?

Btw, nice post Nico <3

PDO does nothing to help you against XSS, just use htmlspecialchars() whenever you output something.

Don't save the HTML encoded string in the database, just use it when you show a string.

Jaldre - June 7, 2014, 9:35 am

Wow, awesome!

This will help the newbies, and I learned a lot too!

Thank you very much for this well written guide!

Cheers,

Jaldrè

1n9i9c7om - June 9, 2014, 2:53 pm

Thanks for the feedback!

(Jun 2, 2014 05:05 PM)Hexley Wrote: [\[CITE\]](#) Great tutorial for beginners OP, but the "or" statement was missed, and that is a great (and underused) statement..

```
Code if($Number < 0 || $Number > 10) {
```

```
    echo "The variable has a variable that is not allowed, you typed in " + htmlentities($Number) . " .";  
}Rennet, in double quotes echo, you can reference a variable with brackets. {$Number}
```

I'm gonna add this as soon as the tutorial contest ends. I think there shouldn't be any problem with editing a tutorial while the contest is running, but I wanna make sure. ;)

TRH4CK - June 10, 2014, 6:25 am

(Jun 5, 2014 07:13 AM)VipVince Wrote: [\[CITE\]](#) Yes. Consider a guestbook; non sanitized input being stored in the DB will result in a persistent XSS being displayed on the webpage.

But if I use htmlspecialchars() in output of that unsanitized input, would there still be a XSS?

1n9i9c7om - June 10, 2014, 6:29 am

(Jun 10, 2014 06:25 AM)TRH4CK Wrote: [\[CITE\]](#) But if I use htmlspecialchars() in output of that unsanitized input, would there still be a XSS?

Nope, there won't. htmlspecialchars() converts special chars in HTML entities. For example, "<" would become "<".

A browser would still display it as "<", but in the source of the page it appears as "<", so XSS isn't possible.

scam. - June 12, 2014, 9:31 pm

Your php game too strong Nico. Solid tutorial my man!

1n9i9c7om - June 16, 2014, 7:30 am

I'm going to edit this tutorial a bit this weekend, quiet busy because of my graduation. :)

1n9i9c7om - June 22, 2014, 5:34 am

(Jun 2, 2014 05:05 PM)Hexley Wrote: [\[CITE\]](#) Great tutorial for beginners OP, but the "or" statement was missed, and that is a great (and underused) statement..

```
Code if($Number < 0 || $Number > 10) {
```

```
    echo "The variable has a variable that is not allowed, you typed in " + htmlentities($Number) . " .";
```

```
}Rennet, in double quotes echo, you can reference a variable with brackets. {$Number}
```

I mentioned the "or" statement, but I forgot to give an example. I added that. :)

Nivarka - September 4, 2014, 11:50 pm

Why is this not a sticky? PM Omni right now, my man!