



PHP and Ajax

[Set-Up](#)

[Basics](#)

[Variables and Data types](#)

[Constants](#)

[Operators](#)

[Arithmetic](#)

[Inc/dec](#)

[Output Constructors & Functions](#)

[Arrays](#)

[Associative array](#)

[Conditionals](#)

[If](#)

[Switch](#)

[Loops](#)

[While](#)

[For](#)

[ForEach](#)

[Do While](#)

[Functions](#)

[Array Functions](#)

[String Functions](#)

[Math Functions](#)

[isset\(\) and empty\(\)](#)

[isset\(\)](#)

[empty\(\)](#)

[include\(\)](#)

[header\("Location: home.php"\) → this is how you redirect to another page](#)

[RadioButtons and Checkboxes](#)

[RadioButton](#)

[Checkboxes](#)

[Hashing](#)

[SuperGlobals](#)

[\\$_GET](#)

[\\$_POST](#)

[Sanitizing inputs](#)

Validate Inputs

\$_COOKIE

Set a cookie

Get a cookie

Delete a cookie

\$_SERVER

\$_SESSION

How to start a session

End a session

\$_FILES

\$_REQUEST

File Handling

File Uploading

Errors & Exceptions

throw

try / catch

OOP & Classes

Inheritance

PHP Connection to MySQL

How to Connect to MySQL database

- <https://www.youtube.com/watch?v=zZ6vybT1HQs>

Set-Up

- VS Code
- XAMPP - so you can run your php code, get access to phpMyAdmin for better database connection and so on:
<https://www.apachefriends.org/index.html>
- Download MySQL and phpMyAdmin
- start the control panel
- start apache and MySQL
- create your folder in xampp/htdocs
- in VS Code:
 - set the path to your C:/xampp/php/php.exe
 - download the following extensions:

- PHP Intelliphense
 - Live Server
 - PHP Server
 - create index.php → your index for the app
 - go to localhost/your-folder-name → in your browser and you will see your website
-

Basics

- `<?php`
echo "This is how you write php code. You output shit with 'echo'."; `?>`
 - download live server web extension and add localhost/your-webapp to the front row and what you get from opening vs code live server down there, click live reload and apply and boom, now you can see the app lil bro
 - `//` this is how you write comments `/* multiline com */`
 - you can add `
` in your php echo
 - you can add html to your php file
-

Variables and Data types

- `$name = "Gulin Tudor";`
echo name; ⇒ shows name onto the screen
- echo "Hello, {name} !";
- you can have
 - integer
 - string
 - float
 - bool
 - null
 - etc.

Constants

- declare it like this:
 - `define('CONSTNAME', 'ACTUALVALUE');`

Operators

Arithmetic

- `+`
- `-`
- `*`
- `/`
- `**` - exponentiate
- `%`

Inc/dec

- `--` / `++`
-

Output Constructors & Functions

- `echo`
 - outputs string, numbers, html etc.
 - `echo 123, 'Hello', 44; ⇒ 123Hello44`
- `print`
 - work like echo, but can take only one parameter
 - `print 123;`
- `print_r()`
 - prints single values and arrays
 - `print_r([1,2,3]);`
- `var_dump()`

- returns more data like: data_type and length
 - `var_dump('Gulin');` → `string(5) Gulin`
 - `var_export()`
 - similar to `var_dump`
 - `var_export('Gulin');` → `'Gulin'`
-

Arrays

- `$myArray = [1, 2, 3, 4, 5];`
- `$myArray = array('ana', 'maria', 'petra');`
- `print_r($myArray);` → to show it (or `var_dump`)
- `echo myArray[1];` → `maria`

Associative array

- `$color = [
1 ⇒ 'red',
2 ⇒ 'blue'
];`
 - `$hex = [
'red' ⇒ '#f00',
'blue' ⇒ '#0f0'
];`
 - you access values this way: `$hex['red']`
-

Conditionals

If

- ```
if($age >= 18){
 echo 'you can drive';
} elseif ($age >=16) {
 echo 'you can drive a small car';
} else {
```

```
 echo 'you cant drive lil bro';
}
```

- short if:  
!empty(\$myArray) ? ifTrue : ifNotTrue;

## Switch

- switch( \$favoriteColor){  
 case 'red':  
 echo 'My fav color is red';  
 break;  
 case 'blue':  
 echo ...;  
 break;  
 default:  
 echo ...'  
}

=====

- >
- <
- < =
- > =
- ==
- === equal and same type
- != not equal
- !== not same type

## Loops

### While

```
while($x > 5){
 //do smth
}
```

## For

```
for($x = 0; $x<10; $x++){
 // do smth
}
```

## ForEach

```
foreach($array as $value){ // as $index ⇒ $value you also get the index
 echo $value; // you can do as $key ⇒ value for associative arrays
}
```

## Do While

```
do {
 // ...
} while(...)
```

---

## Functions

```
<?php
```

```
$globalVariable = 10;
```

```
function myFunction($parameter = defaultValue) {
 GLOBAL $globalVariable;
 $localVariable = 5; // this is only in the function scope
 echo $localVariable;
 echo 'Because i said GLOBAL I can use that variable';
 echo 'You can pass parameters: {$parameter}';

 return $parameter + $localVariable + $globalVariable;
}
```

```
echo myFunction(10); ⇒ it echoes 25
```

- you can give a function to a variable and use it that way:  

```
$subtraction = function ($n1, $n2) {
 return $n1 - $n2;
 }
 echo $subtraction(5,2);
```

- `$multiply = fn($n1, $n2) ⇒ $n1 * $n2;`

## Array Functions

- built in functions for arrays
- let's define an array:  

```
<?php
$colors = ['red','blue','yellow'];
```
- add an element: ( at the end)  

```
array_push($colors, 'green');
```
- add an element(at the front):  

```
array_unshift($colors, 'purple');
```
- `print_r($colors) ⇒ purple, red, b, y, green`
- Remove an element: ( removes from the end)  

```
array_pop($colors); ⇒ removes green
```
- remove an element from the front:  

```
array_shift($colors) ⇒ removes purple
```
- remove a specific element:  

```
unset($colors[1]); ⇒ removes blue + the index of 1
```

⇒ \$colors:

```
0 : 'red',
2: 'yellow'
```
- Split into chunks:  

```
$chunked_colors = array_chunk($colors, 2);
```

⇒ 0: {2 elems} 1:{2elems} etc
- `$arr1 = [1,2,3]; $arr2=[4,5,6]`  

```
$arr3 = array_merge($arr1, $arr2);
```

⇒ `$arr3 = [1,2,3,4,5,6];`
- or do:  

```
$arr4 = [...arr1, ...$arr2];
```
- TO COMBINE and get KEY + VALUE:  

```
$keys = [...]
```

```
$values = [...]
```

```
$dictionary = array_combine($keys, $values);
```



- to flip the keys and the values( values becomes the keys for their keys)  
`$flip = array_flip($dictionary);`
- `$numbers = range(1,20);`  
`$numberSquares = array_map( function ($number) {`  
`return $number * $number;`  
`});`
- `array_filter ( analogy from above }, $numbers);`
- `array_reduce ( $numbers, fn($carry, $number) => $carry+$number);`

## String Functions

- `strlen($string) => length`
- `strpos( $string, 'a') => first position of a`
- `strrpos( $string, 'b') => last position of b`
- `strrev( $string) => reverses`
- `strtolower()`
- `strtoupper()`
- `str_replace('word', 'replace with this', $string)`
- `substr($string, start, end) => gets the substring from start to end ( ex from 0 to 5)`
- `str_starts_with($string, 'Hello') => bool`
- `str_ends_with($string, '!') => bool`
- You can give html tags to a string:  
`$string = '<h1>Hello</h1>';`  
`echo $string => HELLO ( as a header 1)`  
do echo `htmlspecialchars $string` to get `<h1>Hello</h1>`

## Math Functions

- `abs($x);` → absolute value
- `round()` → 2,4 ⇒ 2 ; 2,5 ⇒ 3
- `floor()` → 2,8 ⇒ 2
- `ceil($x)` → 2,2 ⇒ 3

- `pow($base, $power) → base^power`
- `sqrt($x) → square root`
- `max($par1, $par2, ...)` ⇒ returns the max
- `min($par1, $par2, ...);`
- `pi();` ⇒ 3.14...
- `rand(1, 100);` ⇒ random value between 1 and 100

## **isset() and empty()**

### **isset()**

- returns true if a variable is declared and not null

### **empty()**

- return true if a variable is not declared, false, null or an empty string: ""

## **include()**

- copies the content of a file ( php/html/text ) and includes it in your php file  
for example, you can reuse footers, navbars, headers and so on
- basically react components after 1 month in Galati
- `include("header.html")`  
`<!DOCTYPE html>`  
`<head>....</head>`  
`<body>...</body>`  
`include("footer.html")`

**`header("Location: home.php")` → this is how you redirect to another page**

## **RadioButtons and Checkboxes**

- create a form
- `<form action="action.php" method="post/get/...">`
  - you get it with `$_POST/GET/...["nameOfInputField"]`

## RadioButton

- `<input type="radio" name="yourName" value="...">`  
`<input type="radio" name="yourName" value="...">`  
`<input type="radio" name="yourName" value="...">`  
→ you need to have the same name so only one is selectable ( so they in the same group )

## Checkboxes

- `<input type="checkbox" name="pizza" value="Pizza">`
- you use `isset` to see if a checkbox is set:  
`isset($_POST["pizza"])`

## Hashing

- hide sensitive data from 3rd parties
- `$password = "jocuri2003"`
- `$hash = password_hash(password, hashingAlgorithm);`
- example:  
`$hash = password_hash($password, PASSWORD_DEFAULT)` → you store this hash in a database, this is protected and harder to decipher than a normal password
- `if( password_verify($password, $hash){ // de-hash or whatever it is called`  
    `// if the password and that hash are mathematically consistent, then it`  
    `returns true`  
    `}`

## SuperGlobals

- variables available in all scopes like `$_GET`, `$_POST` etc. ( they are all arrays )
- when you have a form for example, you give it an action:  
`action = "action.php"`  
and you also give it a method: `post, get, delete, put` etc.  
and then you have smth like:

`$_POST["confirm"]` where confirm is your submit button with `name="confirm"`  
you can get values from your form inputs like this:  
`$value = $_POST["inputFieldName"]`  
and then do smth with it :)  
aint that deep lil ni

## **\$\_GET**

- you give the `"method = "get" "` attribute to an html form tag and then:
- `$_GET` → you can use it to get a certain field this way:  
`$_GET["username"]` ( where username is your field )
- However, this is not secure ( it will change the url and show the username, password and all the fields of that form ⇒ use method post and `$_POST`
- Summary:
  - Data appended to url ⇒ NOT SECURE
  - character limit
  - Bookmark is possible
  - get requests can be cached
  - Better for search pages

## **\$\_POST**

- Summary
  - Data is packaged inside the body of the http request ( more secure )
  - No Data Limit
  - Cannot bookmark
  - Can't be cached
  - Better for submitting credentials

## **Sanitizing inputs**

- this means

- You do not want to get inputs exactly as they are, as someone might insert javascript in there or smth that will ruin your page or database  
example of such thing  
`$username = $_POST["myInputField"]`
- In order to 'sanitize' that input you do smth like this:  
`$username = filter_input( INPUT_POST/GET/..., "inputFieldName", typeOfFilter )`
- example:  
`$username = filter_input(INPUT_POST, "username ( nameOfAnInputField )", FILTER_SANITIZE_SPECIAL_CHARS)` → this filters a post request for all special characters ( so javascript doesnt run )
- `FILTER_SANITIZE_NUMBER_INT` → it checks to be an integer( from fafjaj43dgs it will only leave 43)
- `$email = filter(...,..., FILTER_SANITIZE_EMAIL)` → it will only get an email with legal characters, if there is one

## Validate Inputs

`$age = filter_input(INPUT_POST, "age", FILTER_VALIDATE_INT)`

→ if the input field with name="age" is not an int, it will assign an empty string  
unlucky sanitizing, which filters only the characters that fit, this checks more rigidly

( I assume this is superior )

## \$\_COOKIE

- a mechanism for storing data in the remote browsers and tracking or identifying return users
- you can set specific data to be stored in the browser and retrieve it when the user visits again
- it is good to remember some basic info about the user ( dont use it for sensitive data, it's not secure, use session for that instead :P)

## Set a cookie

- `setcookie(a key, a value, whenToExpire)`

- example:  
`setcookie('email', 'gulintudor@gmail.com', time() + 86400 * 30); ( 86400s = 1day → 30 days)`
- you can also save to a specific file path:  
`setcookie(key, value, whenToExpire, "/players")` or whatever
- you can see your cookies in  
inspect → Application

## Get a cookie

- `isset($_COOKIE['email'])` → do smth with the cookie ( usefull for forms and so on so they dont need to re-input the credentials )

## Delete a cookie

- basically u set the cookie, but to the past:  
`setcookie('..',.., time()-12);`

## \$\_SERVER

- contains headers, paths and script locations
- the entries in this array are created by the web server; it shows almost everything about the current web page environment
- `foreach($_SERVER as $key ⇒ $value){`  
`echo "{$key} = {$value} <br>";`  
// to get all that info  
`}`
- example of key ↔ value pairs in \$\_SERVER:  
SERVER\_NAME → can be localhost  
SERVER\_PORT → the port u on  
REQUEST\_METHOD → get/post etc  
SCRIPT\_NAME → index.php or whatever  
etc.
- 

## \$\_SESSION

- it is used to store information on a user in order to be used across multiple pages
- a user is assigned a session-id
- ex: login → so you can access pages as a logged-in user

## How to start a session

- `session_start()` ⇒ start a session
- once started, you can create session key ↔ value pairs:  
`$_SESSION["password"] = jocuri2003`
- you can get the session key/values from a form and check in other pages that certain keys have a certain value and do smth specific for that session id

## End a session

- `session_destroy();`

## \$\_FILES

- 

## \$\_REQUEST

# File Handling

- you create a file with some content
- you can check if a file exists at a given path:  
`file_exists($file='src/users.txt')`
- READ from a file  
`$handle = fopen($file, 'r')`  
`$contents = fread($handle, filesize($file))`  
`fclose($handle)`  
`echo $contents;` → a line with all the content in a certain file
- WRITE in a file  
`$handle = fopen($file, 'w')`

```

$contents = 'Username1' . PHP_EOL . 'Username2' . PHP_EOL . 'Username3'
// php_eol is used to create a new line
fwrite($handle, $contents)
fclose($handle)

```

## File Uploading

- `<input type="file" name="upload">` → you can upload a file from your computer
- `if(!empty($_FILES['upload']['name'])){`  
`print_r($_FILES) // this prints the name, fullpath, size, error and so on`  
`}`
- you can get the file type ( png/jpg etc. ) and check if it is the correct file type
- you can also limit the size of the file from `$_FILES` and so on

## Errors & Exceptions

### throw

- `throw new Exception("Division by zero or smth")`

### try / catch

```

try{
//your code that might throw an Exception
} catch(Exception $e){
echo 'Caught Exception', $e->getMessage(), ' ' ; // this is how u display the
message
} finally{
// what is here happens after everything in the try/catch
}

```

## OOP & Classes

- You can have classes that hold properties and methods and create Objects from that class, like any OOP



- properties can be public → accessed from anywhere  
private → accessed from inside the class  
protected → from inside the class and by inheriting classes

```
class User{
 public $name;
 protected $email;
 protected $password;

 //Constructor
 public function __construct($name, $email, $password){
 $this->name = $name;
 $this->email = $email;
 $this->password = $password;
 }

 //Methods:
 function setName ($name){
 $this->name = $name
 }
 function getName(){
 return $this->name;
 }
}
```

- create an object:  
\$user1 = new User('Marian', 'dumibumi@gmail.com', 'jocuri2004');
- \$user1->name = 'Marian';
- \$user1.setName('Dorel')

## Inheritance

- class Employee extends User {  
protected \$title  
public function \_\_construct(\$name, \$email, \$password, \$title){  
 parent::\_\_construct(\$name, \$email, \$password); //basically the super  
function  
 \$this->title = \$title;  
}

}

# PHP Connection to MySQL

## How to Connect to MySQL database

- either MySQLi Extension or PDO (PHP Data Objects)
- MySQLi:
  - create a MySQL database in your xampp folder
  - press the admin button to your MySQL database
  - go to databases tab → create database ( you can drop it from the databases tab)
  - go to User accounts overview → get username, hostname and password
  - create a php file for the database connection: database.php
  - database.php:

```
<?php
$db_server = "server name"
$db_user = "user name"
$db_password = "";
$db_name = "database name"
$connection = "";
//Establish a connection:
//$conn = mysqli_connect(databaseServerName, username, password,
databaseName);
try{
 $conn = mysqli_connect($db_server, $db_user, $db_password,
$db_name);
}catch(Exception e){
 echo "Could not connect to your database!", $e->getMessage();
}
if($conn){
 // do smth if connected
}else{
```

```
//do smth if not connected
}
?>
```

- make sure to include("database.php") wherever u want to connect

## Create a table

- open phpMyAdmin → go to database tab → select a database → you have a button to create tables → tableName + number of columns + hit create
- Enter Name, Type, Length, default value, null