Server side

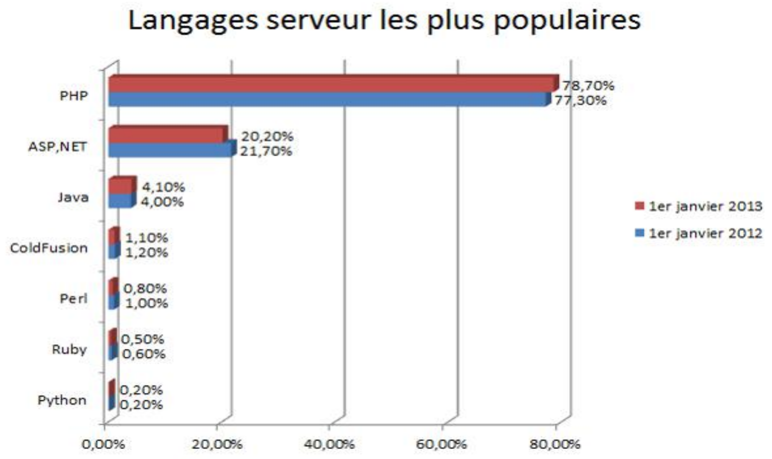
PHP

PHP

- PHP is a programming language used on the server side.
- With PHP, you can:
  - generate the dynamic content of the page
  - create, open, read, write, delete and close files on the server
  - collect data (form data)
  - send and receive cookies
  - add, delete, edit data in a database
  - …

PHP

- Why PHP?
  - open source
  - very popular
  - easy to learn and works efficiently on the server side
  - works on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
  - compatible with almost all web servers used today (Apache, IIS, etc.)
  - supports a wide range of databases (mySql, MSSQL, etc.)



PHP - Installation

- We need a web server and a database
    - Apache
    - MySQL

- Download easyPHP from http://www.easyphp.org/

PHP - Syntax

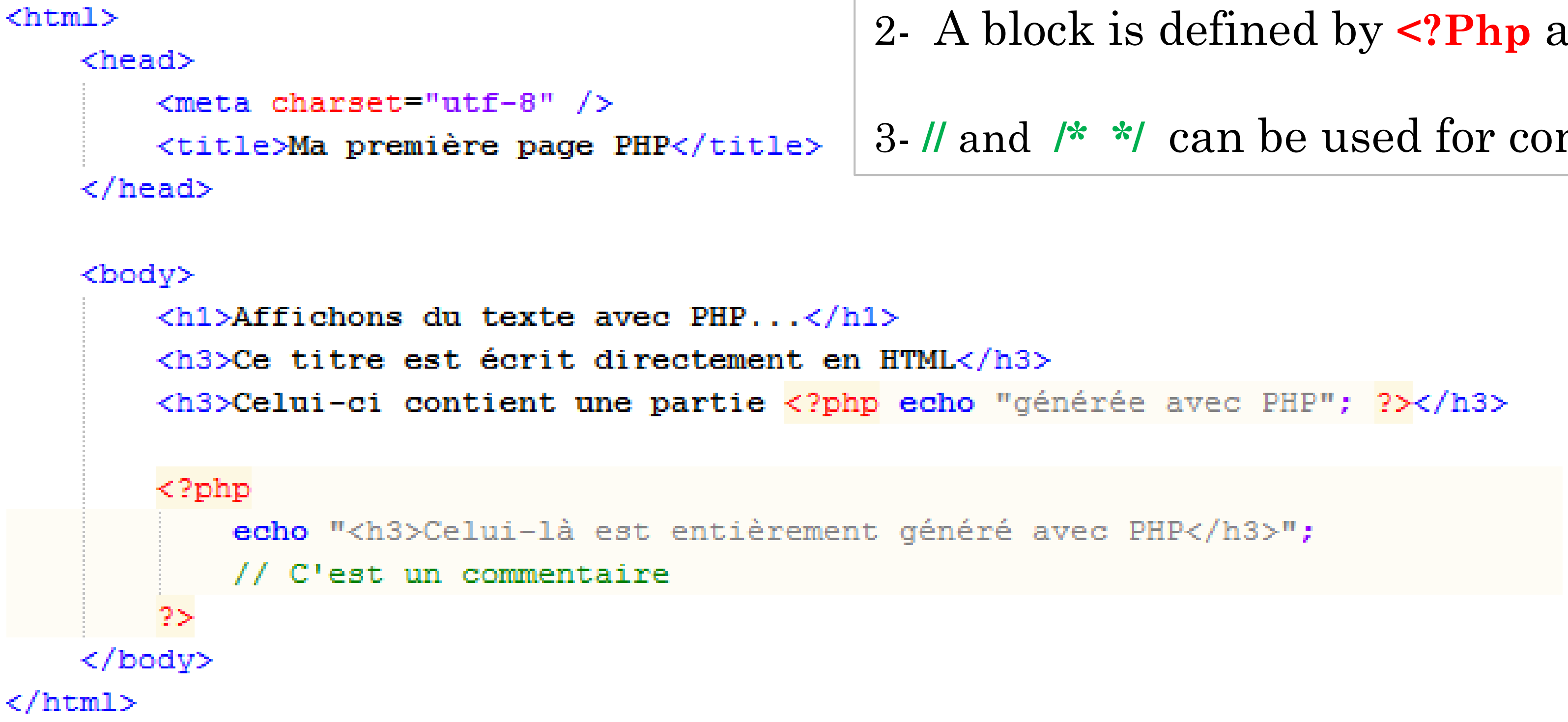
- A PHP source file
  - contains a mixture of HTML tags and PHP code
  - have the .php extension
- All keywords (if, else, while, for, echo, etc.), classes, functions, … are NOT case-sensitive.
- We define a portion of code (or block) PHP with tags ***<? Php*** and ***?>***
- Each statement in this block must end with the symbol ;

PHP – Syntax

1- A PHP script can be placed anywhere in the document.

2- A block is defined by **<?Php** and **?>**

3- // and /* */ can be used for comments



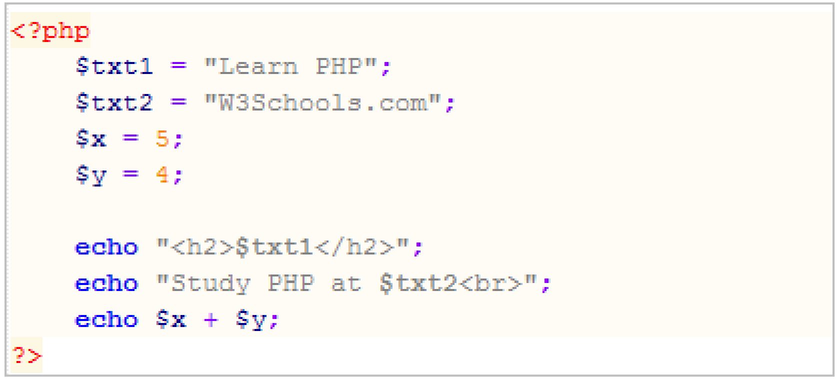
```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Ma première page PHP</title>
    </head>

    <body>
        <h1>Affichons du texte avec PHP...</h1>
        <h3>Ce titre est écrit directement en HTML</h3>
        <h3>Celui-ci contient une partie <?php echo "générée avec PHP"; ?></h3>

        <?php
            echo "<h3>Celui-là est entièrement généré avec PHP</h3>";
            // C'est un commentaire
        ?>
    </body>
</html>
```

PHP - VARIABLE

- In PHP, a variable
  - is used to store information
  - starts with the sign **$** , followed by the **name** of the variable
  - is untyped, no difference between int, float, string, … when declaring
    - PHP automatically converts the variable to the correct data type, depending on its value.
      - $ txt = "Hello world!";
      - $ x = 5;
      - $ y = 10.5;

PHP - VARIABLE

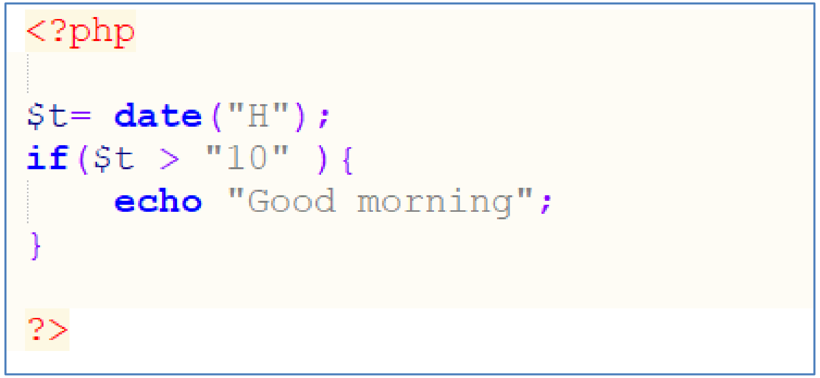
- A variable name in PHP
  - must begin with a letter or underscore character _
  - can not start with a number
  - can only contain alphanumeric characters and the character _ (AZ, az, 0-9 and _)
  - is case sensitive : $age and $AGE are two different variables



```
<?php
    $txt1 = "Learn PHP";
    $txt2 = "W3Schools.com";
    $x = 5;
    $y = 4;

    echo "<h2>$txt1</h2>";
    echo "Study PHP at $txt2<br>";
    echo $x + $y;
?>
```

PHP - CONDITION

if (*condition*) {
    code executed if the condition is true
}

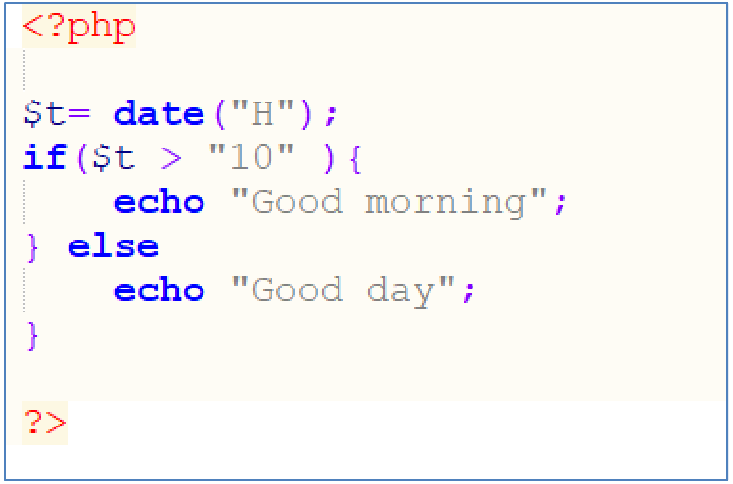


```
<?php

$t= date("H");
if($t > "10" ){
     echo "Good morning";
}

?>
```

The date ("H") function returns at the time of execution

PHP - Condition

```
if (condition) {
    code executed if the condition is true;
} else {
    code executed if the condition is false;
}
```

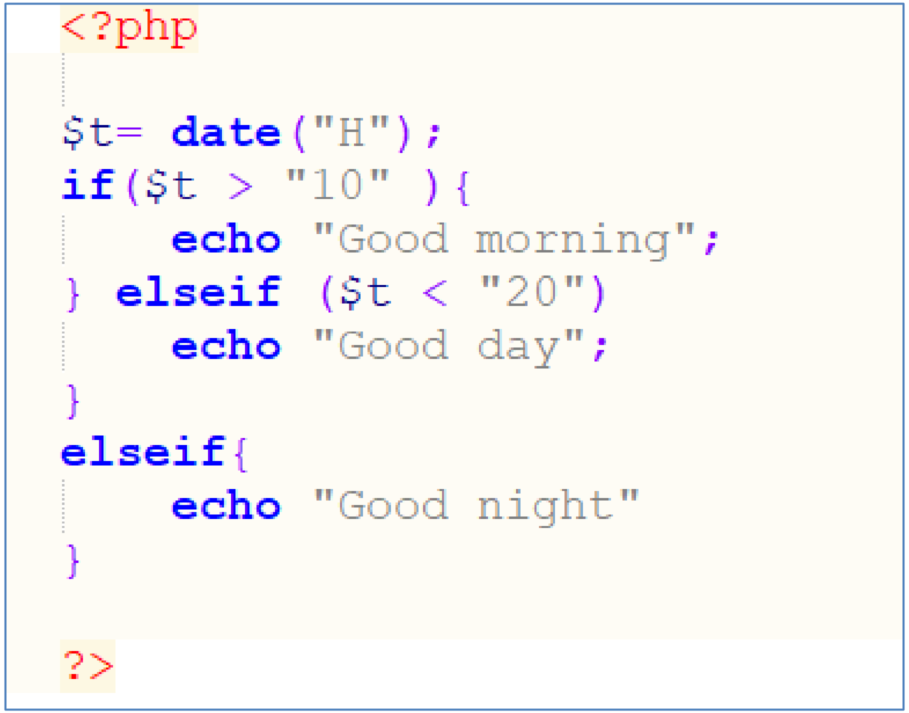


```
<?php

$t= date("H");
if($t > "10" ){
     echo "Good morning";
} else
     echo "Good day";
}

?>
```

PHP - Condition

```
if (condition1) {
    code executed if condition1 is true;
} elseif (condition2) {
    code executed if condition2 is true;
} else {
    code executed all conditions are false;
}
```

PHP - Condition

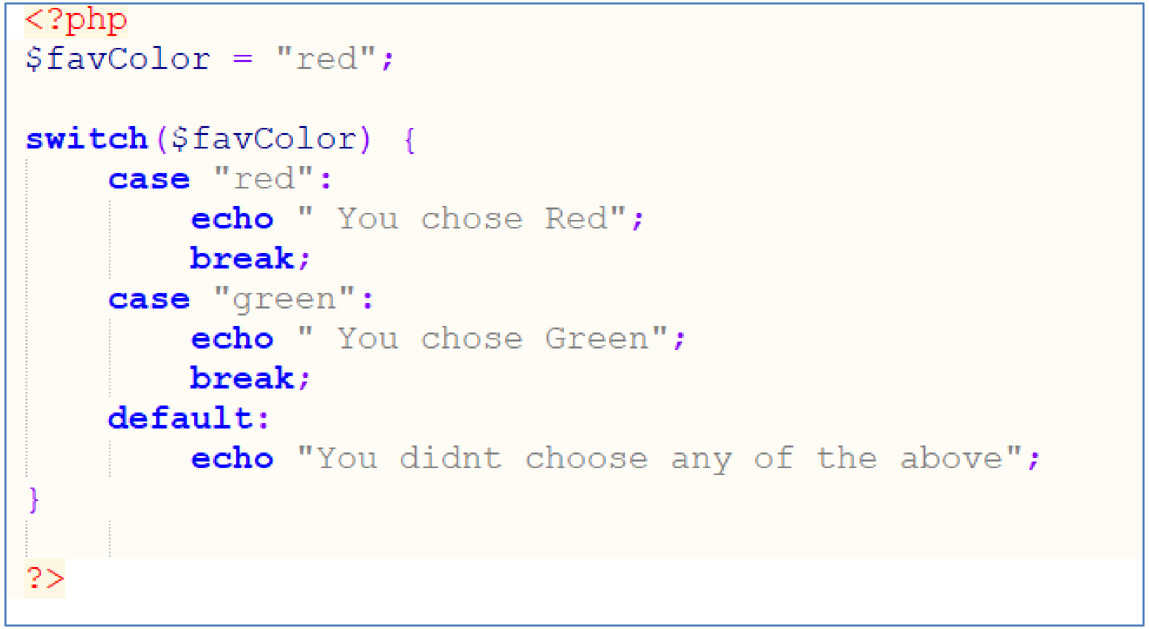


```
<?php

$t= date("H");
if($t > "10" ){
      echo "Good morning";
} elseif ($t < "20")
      echo "Good day";
}
elseif{
      echo "Good night"
}

?>
```

PHP - Condition

```
switch (n) {
    case label1:
                code executed if n = label1;
                break;
    case label2:
                code executed if n = label2;
                break;
    case label3:
                code executed if n = label3;
                break;
    ...
    default:
                code executed if n is different from all labels
}
```

PHP - Condition



```
<?php
$favColor = "red";

switch($favColor) {
    case "red":
        echo " You chose Red";
        break;
    case "green":
        echo " You chose Green";
        break;
    default:
        echo "You didnt choose any of the above";
}

?>
```

break : is used to prevent the code from passing to the following case when a condition is successful
default : is used if no match is found.

PHP – OPERATORS

- Operators are used to perform operations on variables and their values.
- PHP divides operators into the following groups:
  - arithmetic operators
  - assignment operators
  - comparison operators
  - increment / decrement operators
  - logical operators
  - string operators

PHP – Operators

Arithmetic operators

Operator	Name	Example	Result
+	Addition	$ x + $ y	Sum of $ x and $ y
-	Substraction	$ x - $ y	Subtraction of $ x and $ y
*	Multiplication	$ x * $ y	Multiplication of $ x and $ y
/	Division	$ x / $ y	Division of $ x and $ y
%	modulo	$ x% $ y	The rest of the division of $ x by $ y
**	Exponential	$ x ** $ y	$ x has the power $ y

PHP – Operators

Assignment operators

Operator
x = y
x + = y
x - = y
x * = y
x / = y
x% = y

PHP – OPERATORS

Comparison operators

Operator	Name	Example	Result
==	Equal	$ x == $ y	True if $ x equals $ y
===	Identical	$ x === $ y	True if $ x equals $ y **and** both are of the same type
! =	Not equal	$ x! = $ y	True if $ x is not equal $ y
<>	Not equal	$ x <> $ y	True if $ x is not equal $ y
! ==	Not identical	$ x! == $ y	True if $ x is not equal to $ y, **or** they are not of the same type

PHP – Operators

Logical operators

Operator	Name	Example	Result
and	and	$x and $ y	True if both are true
or	or	$x or $ y	True if one of them is true
xor	xor	$x xor $ y	True if one of them is true only but not both
&&	and	$x && $y	True if both are true
||	or	$x || $y	True if one of them is true
!	not	!$x	True if $x is false

PHP – Operators

Increment / decrement operators

Operator	**Name**	**Result**
++ $x	Pre-increment	Increment $x by one and return $x
$ x++	Postincrement	Returns $x then increments it by one
- $x	Pre-decrement	Decrements $x from one and returns $x
$x--	Post-decrement	Returns $x then decrements it by one

PHP – OPERATORS

String operators

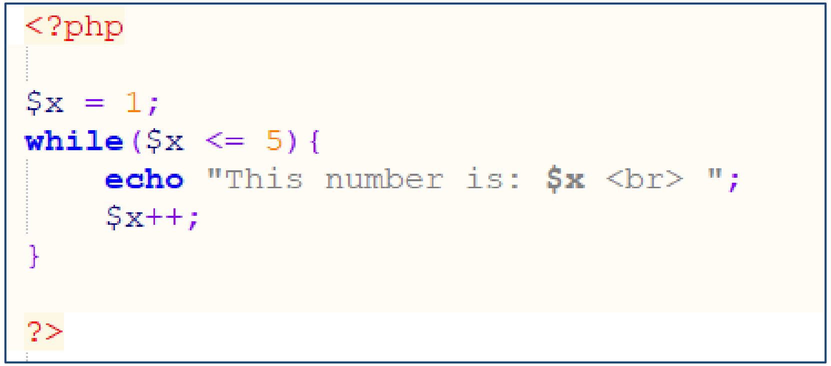
Operator	Name	Example	Result
.	Concatenation	$ txt1. $ txt2	Concatenation of txt1 and txt2
. =	Concatenation and assignment	$ txt1. = $ txt2	Assignment at txt1 the concatenation of txt1 and txt2

PHP – Loops

- A repetitive or iterative structure allows us to repeat several times the execution of one or more instructions.
- The number of repetitions can:
  - to be known in advance.
  - depend on the evaluation of a condition.
- At each repetition, the instructions contained in the loop are executed.
- This is called a loop turn or an iteration.

PHP – Loops

- The ***while*** loop allows you to repeat statements as long as a condition is true.
    - while (*condition is true*) {
      code to execute
      }



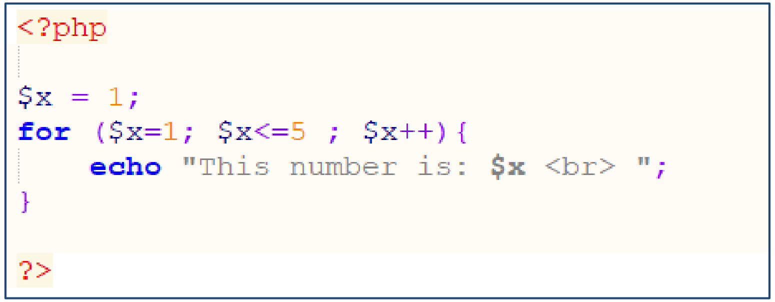
```
<?php

$x = 1;
while($x <= 5){
    echo "This number is: $x <br> ";
    $x++;
}

?>
```

```
This number is: 1
This number is: 2
This number is: 3
This number is: 4
This number is: 5
```

This loop is executed 5 times.
When $x becomes equal to 5 (or greater than 5), the loop is exited

PHP – Loops

- The ***for*** loop allows to repeat a block of instructions a defined number of times.
  - for (initialization; condition; incrementation) {
    code to execute;
    }



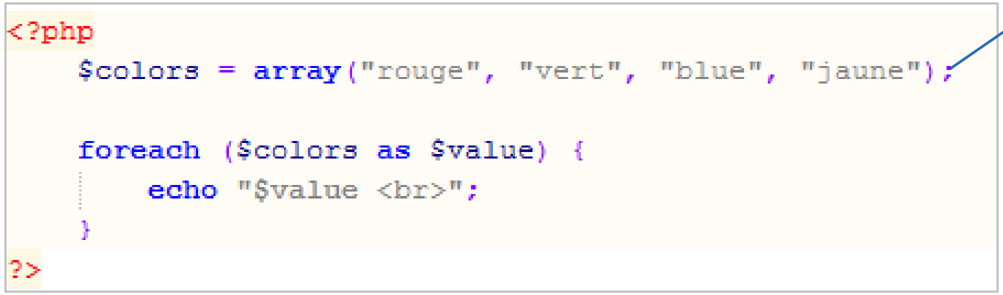
```php
<?php

$x = 1;
for ($x=1; $x<=5 ; $x++){
    echo "This number is: $x <br> ";
}

?>
```

```
This number is: 1
This number is: 2
This number is: 3
This number is: 4
This number is: 5
```

- Initialization occurs <u>only</u> once, at the beginning of execution.
- The condition is evaluated before each loop turn.
  - If true, a new loop turn is made.
  - Otherwise, the loop is complete.

PHP – Loops

- The ***foreach*** loop is used to browse an array.
- *foreach* (*$array* as *$value*) {
    code to execute
  }



```
<?php
    $colors = array("rouge", "vert", "blue", "jaune");

    foreach ($colors as $value) {
        echo "$value <br>";
    }
?>
```

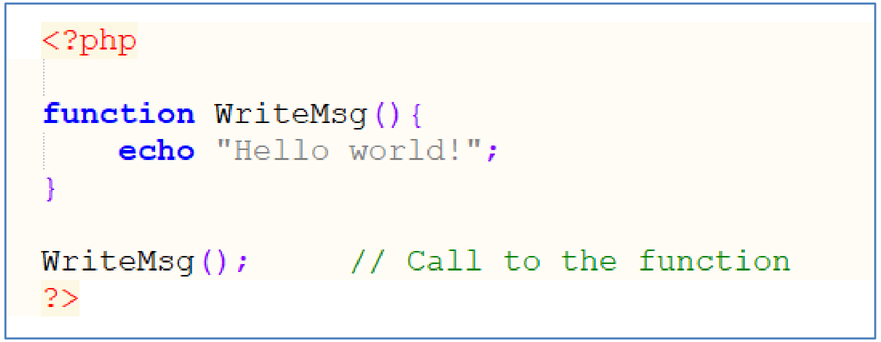
table: $colors

- For each iteration of the loop,
  - the value of the current element (of the array) is assigned to $ value
  - and the table pointer is moved by 1
- Until it reaches the last element of the array.

PHP – Functions

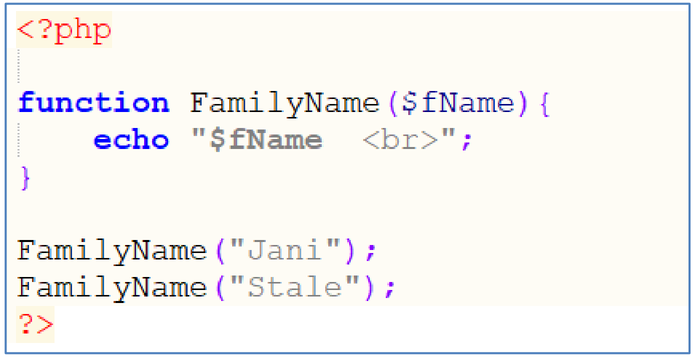
- *function FunctionName($parameter) {*
  // code to execute
  return $valeur
  }

No parameter, no return value



```
<?php

function WriteMsg(){
    echo "Hello world!";
}

WriteMsg();     // Call to the function
?>
```

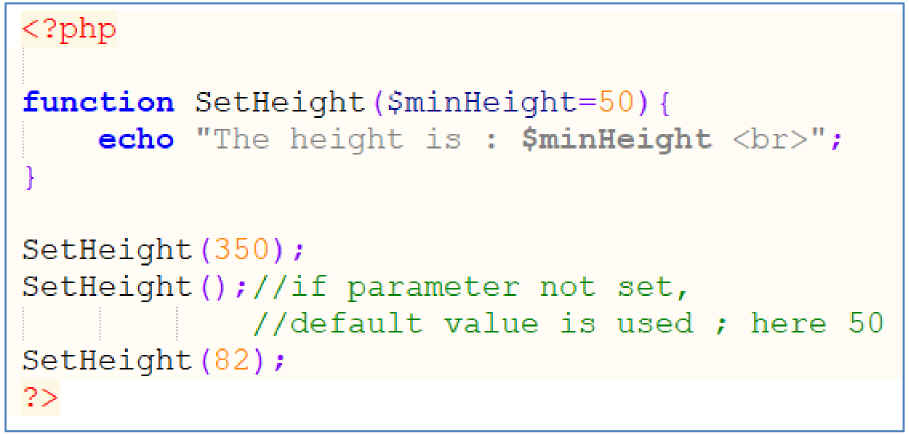
With parameter, no return value



```
<?php

function FamilyName($fName){
    echo "$fName  <br>";
}

FamilyName("Jani");
FamilyName("Stale");
?>
```

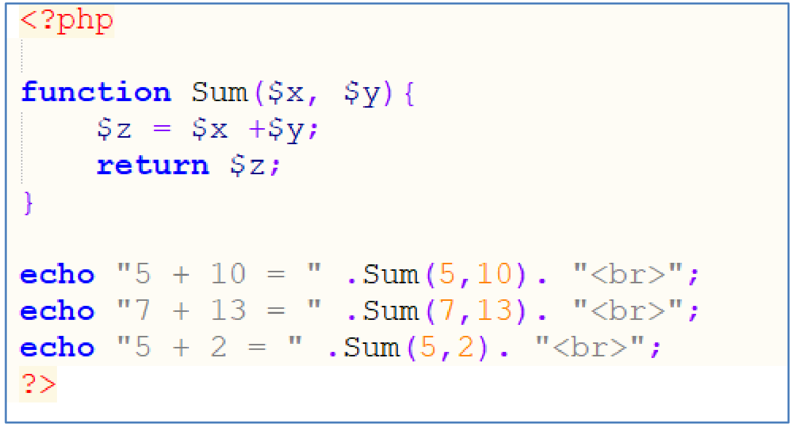
PHP – FUNCTIONS

With parameter, no return value



```php
<?php

function SetHeight($minHeight=50){
    echo "The height is : $minHeight <br>";
}

SetHeight(350);
SetHeight();//if parameter not set,
            //default value is used ; here 50
SetHeight(82);
?>
```

With parameter and return value



```php
<?php

function Sum($x, $y){
    $z = $x +$y;
    return $z;
}

echo "5 + 10 = " .Sum(5,10). "<br>";
echo "7 + 13 = " .Sum(7,13). "<br>";
echo "5 + 2 = " .Sum(5,2). "<br>";
?>
```

Note: Function names are not case-sensitive.

PHP – Arrays

- An array stores multiple values in a single variable

- If you have a list of items (a list of car names, for example), storing cars in simple variables might look like this:

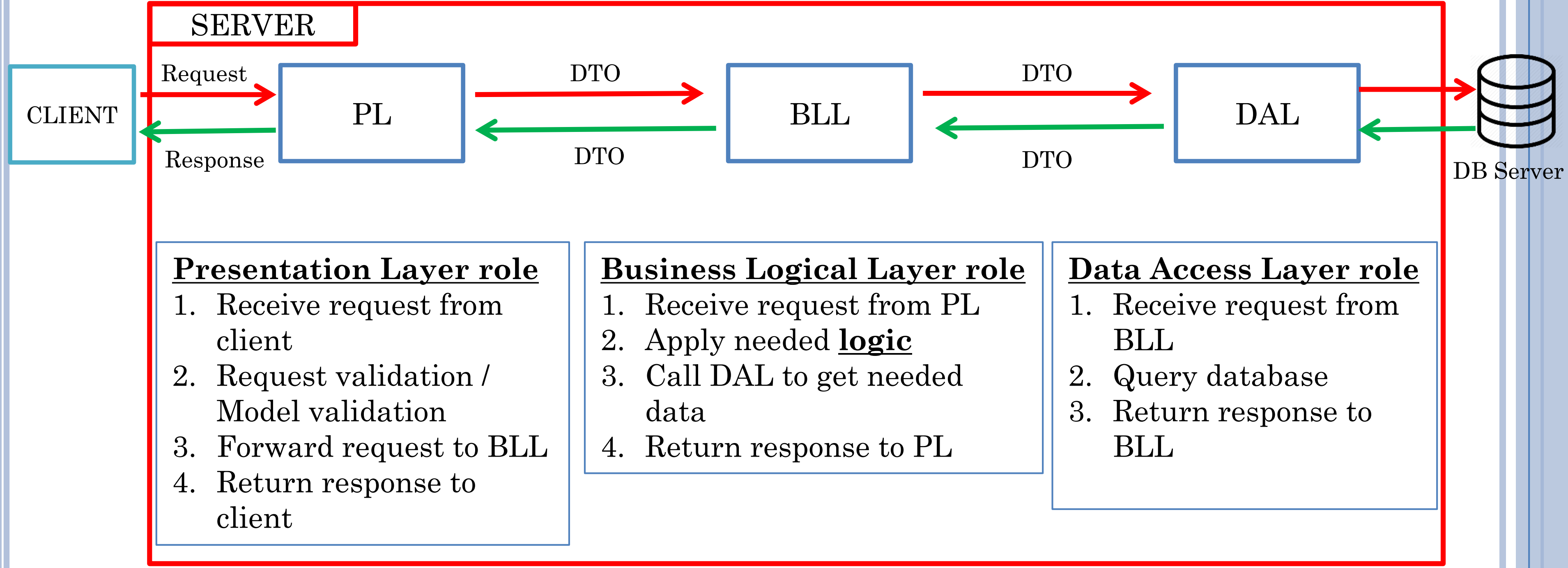
```
$cars1 = “Honda";
$cars2 = "BMW";
$cars3 = "Toyota";
…
```

PHP – Arrays sorting

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

Three tiers architecture

Three tiers architecture



<u>Presentation Layer role</u>

1. Receive request from client
2. Request validation / Model validation
3. Forward request to BLL
4. Return response to client

<u>Business Logical Layer role</u>

1. Receive request from PL
2. Apply needed **<u>logic</u>**
3. Call DAL to get needed data
4. Return response to PL

<u>Data Access Layer role</u>

1. Receive request from BLL
2. Query database
3. Return response to BLL

<u>Data Transfer Object - DTO</u>

Are only used to pass data between layers
Does not contain any business logic

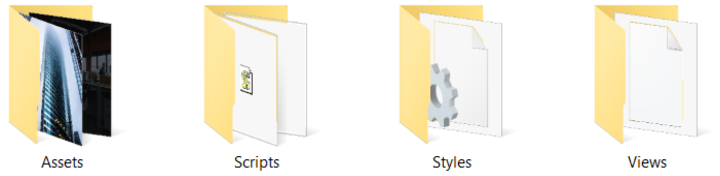
EXAMPLE : SIGNUP

- The following slides show a small example (SignUp) on how to write/organize code to follow the three tiers architecture pattern.
- Code is divided into folders
  - PL : representing the presentation layer
  - BLL : representing the business logical layer
  - DAL : representing the data access layer
- Those folders are all hosted/uploaded to the root folder (www) on the Apache web server

BLL DAL PL

Example : Signup

- Presentation Layer is divided into folders
  - Assets : containing all images, videos, fonts, …
  - Scripts : containing all js files
    - jQuery reference : jquery-3.2.1.js
    - sign-up.js (in this example)
  - Styles : containing all css files
    - site.css
  - Views : containing html and php files
    - SignupForm.php, Signup.php



EXAMPLE : SIGNUP

SignupForm.php

```
<form class="form-control" id="myform" method="post" action="Signup.php" >

    <label for="lname">Last name</label>
    <input name="lname" type="text" class="lname" required>

    <label for="fname">First name</label>
    <input name="fname"class="fname"  type="text" required>

    <label for="email">Email</label>
    <input name="email" class="email"  type="text" required>

    <label  for="gender">Gender</label>
    <input name="gender" value="Male" type="radio" class="male"><label class="gmale">Male</label>
    <input name="gender" value="Female" type="radio" class="female"><label class="gfemale">Female</label>

    <label for="country" >Country</label>
    <select name="country">
        <option value="Lebanon" name="country">Lebanon</option>
        <option value="United States" name="country">United States</option>
    </select>

    <label for="pass">Password</label>
    <input class="password" name="pass" type="password" required>


    <label for="pass">Confirm Password</label>
    <input class="conpass" name="pass"type="password" required>

    <input type="submit" class="submit btn btn-primary" name="SubmitButton" value="submit">

</form>
```

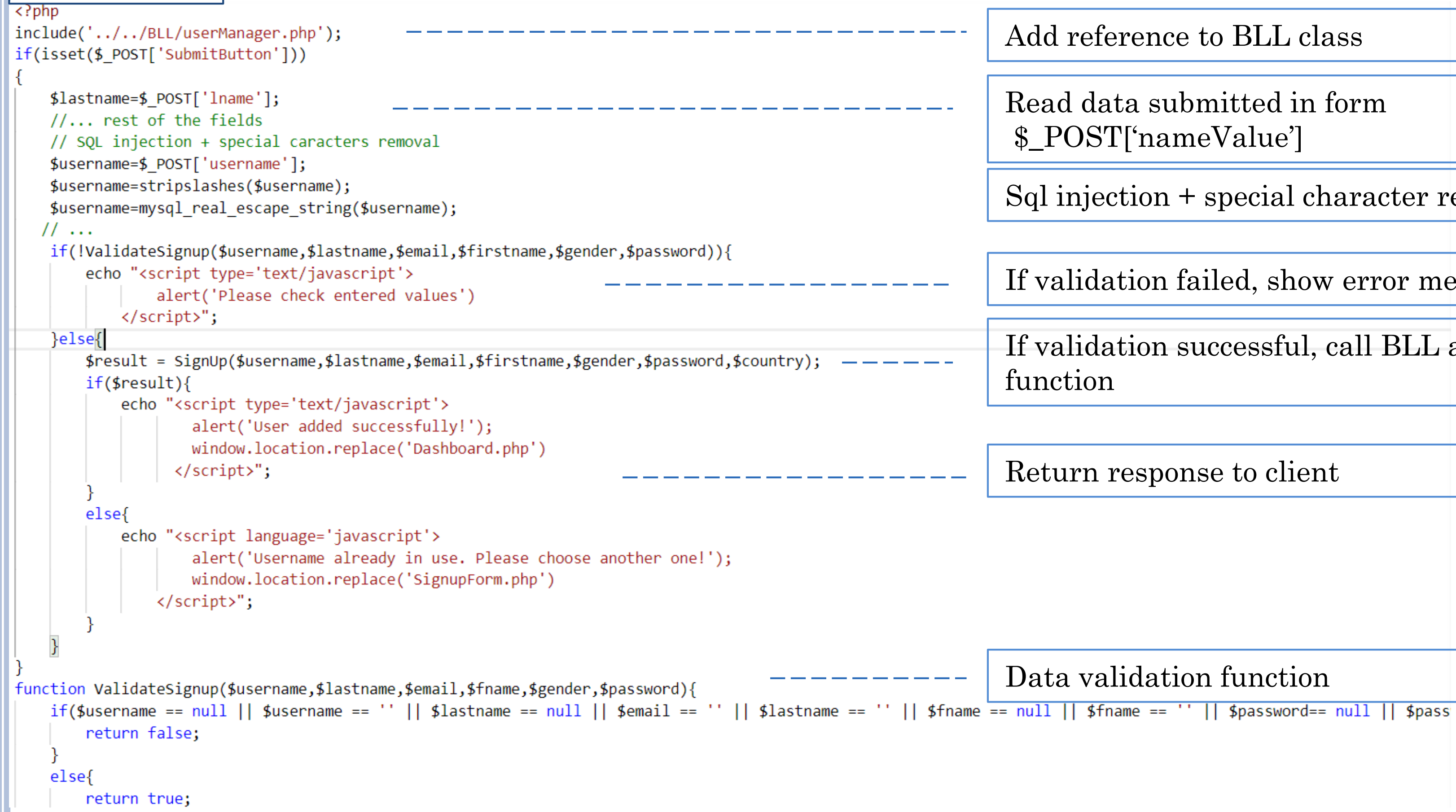
→ The form action attribute specifies where to send the form-data when a form is submitted

- here to *Signup.php*

→A form is triggered only when a button with type="submit" is clicked

Example : Signup

Signup.php



```
<?php
include('../../BLL/userManager.php');
if(isset($_POST['SubmitButton']))
{
    $lastname=$_POST['lname'];
    //... rest of the fields
    // SQL injection + special caracters removal
    $username=$_POST['username'];
    $username=stripslashes($username);
    $username=mysql_real_escape_string($username);
   // ...
    if(!ValidateSignup($username,$lastname,$email,$firstname,$gender,$password)){
        echo "<script type='text/javascript'>
                alert('Please check entered values')
            </script>";
    }else{
        $result = SignUp($username,$lastname,$email,$firstname,$gender,$password,$country);
        if($result){
            echo "<script type='text/javascript'>
                    alert('User added successfully!');
                    window.location.replace('Dashboard.php')
                  </script>";
        }
        else{
            echo "<script language='javascript'>
                    alert('Username already in use. Please choose another one!');
                    window.location.replace('SignupForm.php')
                </script>";
        }
    }
}
function ValidateSignup($username,$lastname,$email,$fname,$gender,$password){
    if($username == null || $username == '' || $lastname == null || $email == '' || $lastname == '' || $fname == null || $fname == '' || $password== null || $pass
        return false;
    }
    else{
        return true;
```

Add reference to BLL class

Read data submitted in form
$_POST['nameValue']

Sql injection + special character removal

If validation failed, show error message

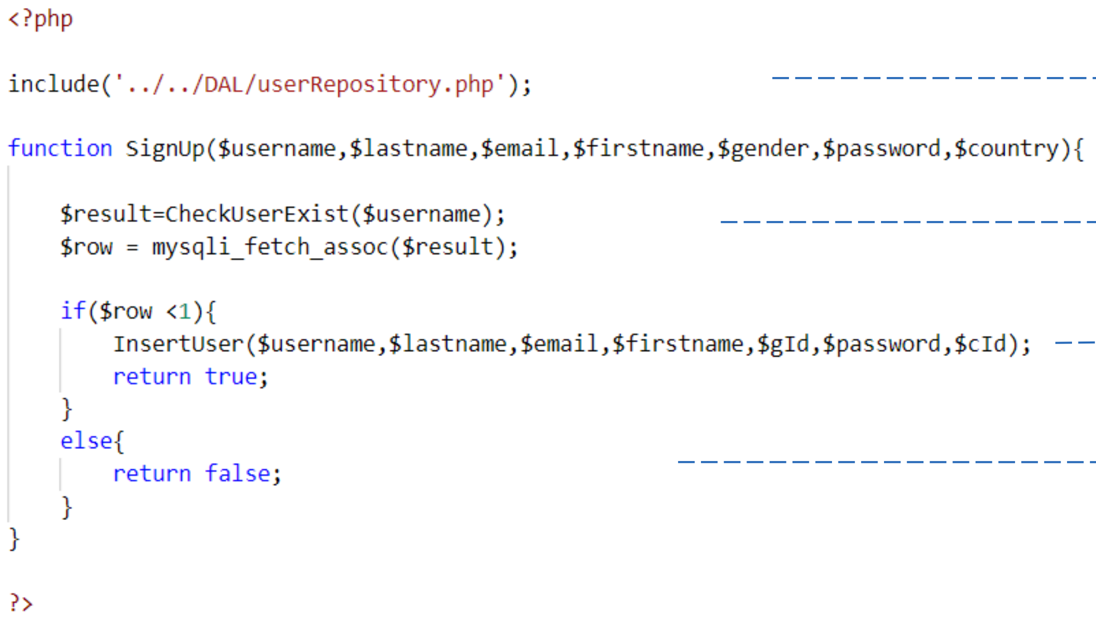
If validation successful, call BLL appropriate function

Return response to client

Data validation function

Example : Signup

BLL/userManager.php



```
<?php

include('../../DAL/userRepository.php');

function SignUp($username,$lastname,$email,$firstname,$gender,$password,$country){

    $result=CheckUserExist($username);
    $row = mysqli_fetch_assoc($result);

    if($row <1){
        InsertUser($username,$lastname,$email,$firstname,$gId,$password,$cId);
        return true;
    }
    else{
        return false;
    }
}

?>
```

Add reference to DAL class

Function called from PL

Call DAL function to check if username exist

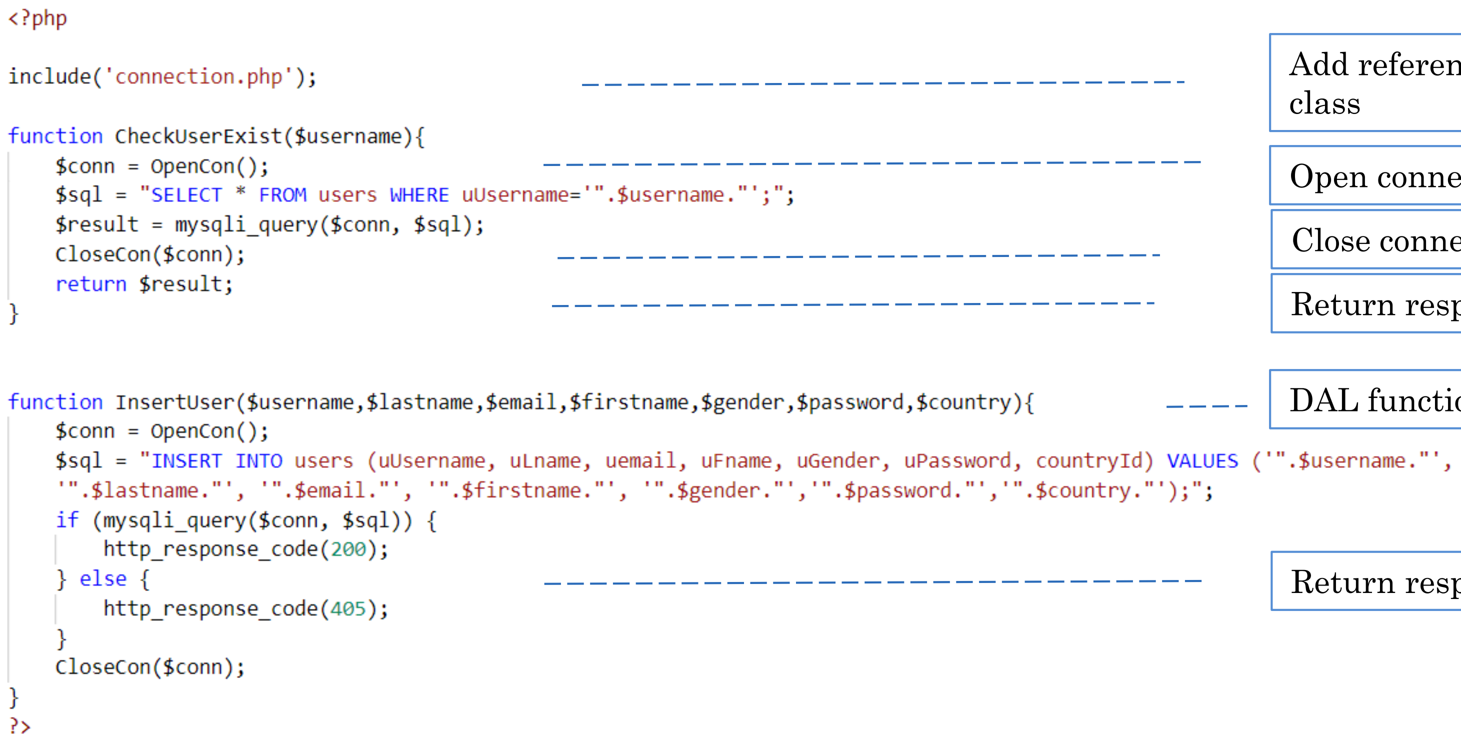
Signup logic

Call DAL function to add new user

Return response to PL

EXAMPLE : SIGNUP

DAL/userRepository.php



```php
<?php

include('connection.php');

function CheckUserExist($username){
    $conn = OpenCon();
    $sql = "SELECT * FROM users WHERE uUsername='".$username."';";
    $result = mysqli_query($conn, $sql);
    CloseCon($conn);
    return $result;
}


function InsertUser($username,$lastname,$email,$firstname,$gender,$password,$country){
    $conn = OpenCon();
    $sql = "INSERT INTO users (uUsername, uLname, uemail, uFname, uGender, uPassword, countryId) VALUES ('".$username."',
    '".$lastname."', '".$email."', '".$firstname."', '".$gender."','".$password."','".$country."');";
    if (mysqli_query($conn, $sql)) {
        http_response_code(200);
    } else {
        http_response_code(405);
    }
    CloseCon($conn);
}
?>
```

Add reference to connection class

Open connection to DB

Close connection to DB

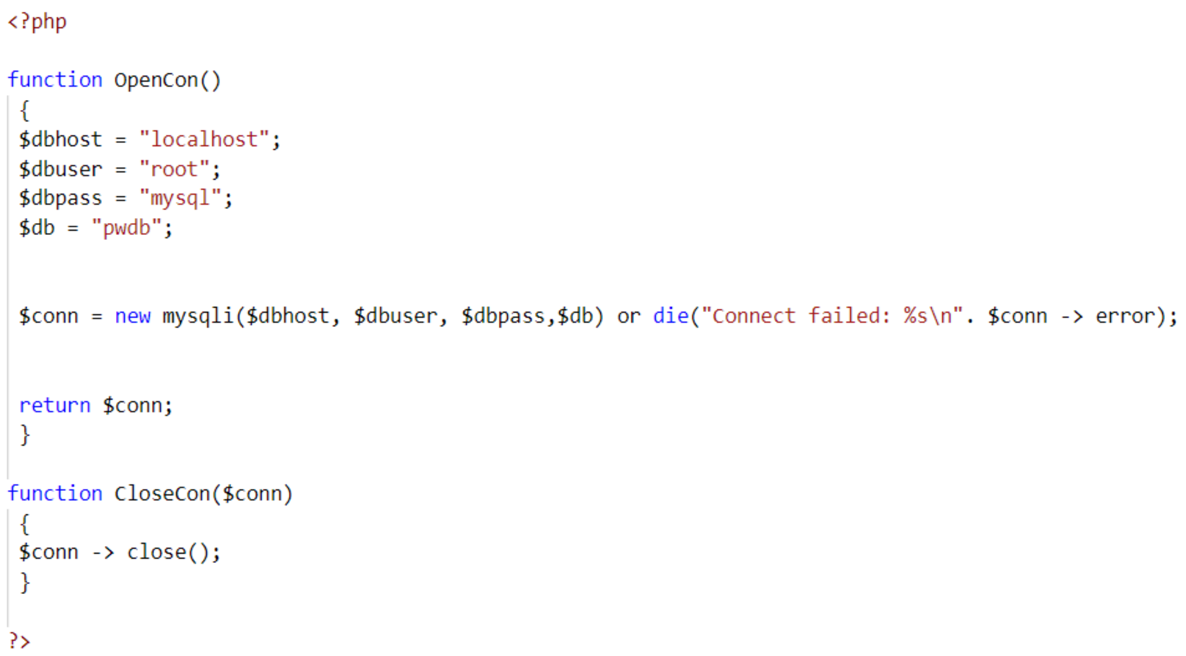
Return response to BLL

DAL function

Return response to BLL

Example : Signup

DAL/connection.php



```
<?php

function OpenCon()
 {
 $dbhost = "localhost";
 $dbuser = "root";
 $dbpass = "mysql";
 $db = "pwdb";


 $conn = new mysqli($dbhost, $dbuser, $dbpass,$db) or die("Connect failed: %s\n". $conn -> error);


 return $conn;
 }

function CloseCon($conn)
 {
 $conn -> close();
 }

?>
```

Manage connection to DB

Session and cookies

Session and cookies

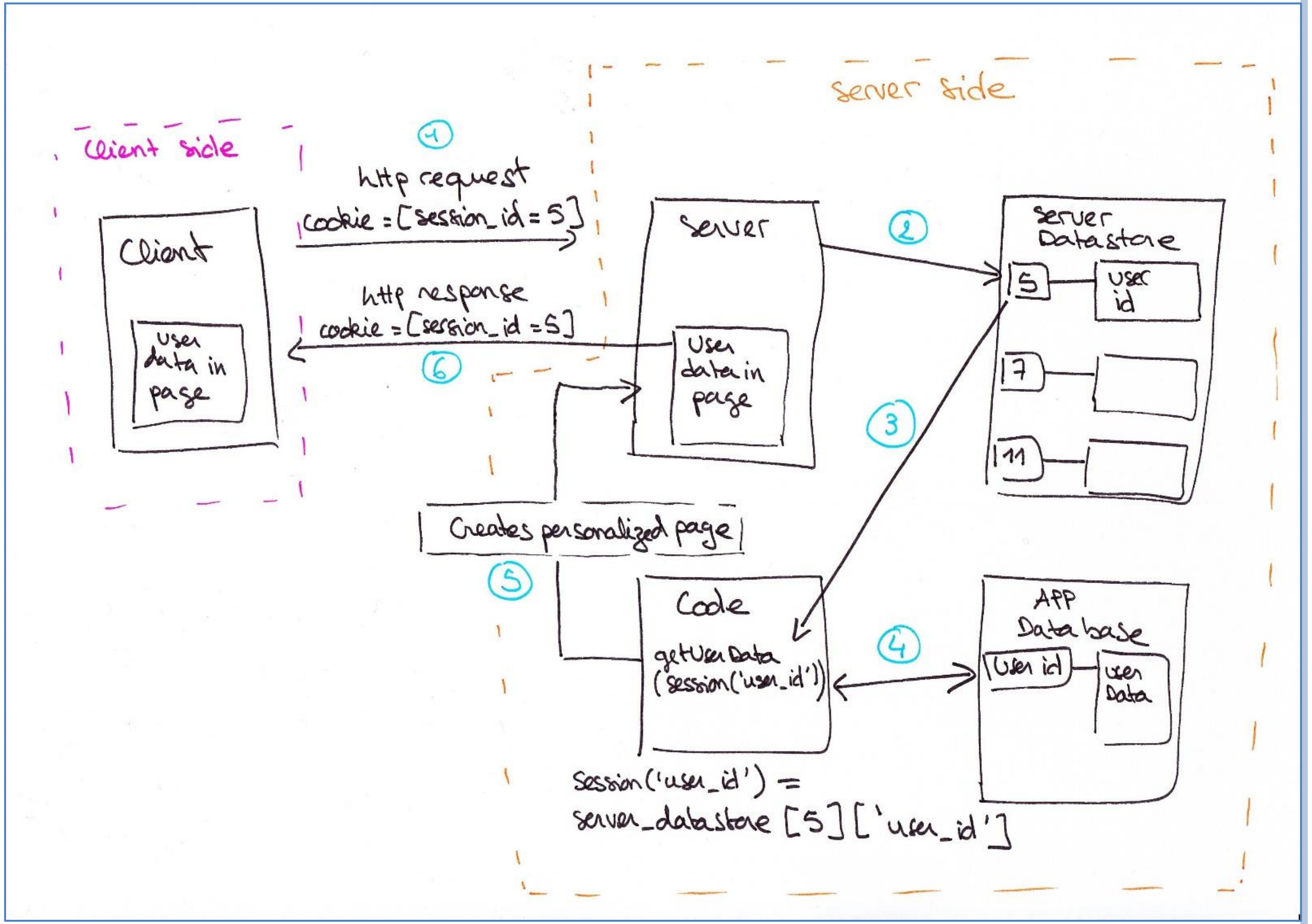
- A cookie
  - is a small file <u>stored by the browser</u> (stored on client-side) and sent to the server with each request
  - often used to identify a user
- A session
  - is a set of data <u>stored on the server</u> and associated with <u>a given user</u>
  - can be used to keep state information between page requests
- Session Ids are normally sent to the browser via session cookies and are used to retrieve existing session data.

SESSION AND COOKIES

- How?
  - A session is created after user authentication
  - The identifier of this session (session Id) is sent to the user during the creation of his session.
  - It is stored in a cookie (called, by default, PHPSESSID)
  - This cookie is sent by the browser to the server with each request
  - The server (PHP) uses this cookie, containing the session Id, to know which file corresponds to this user.

Session and cookies

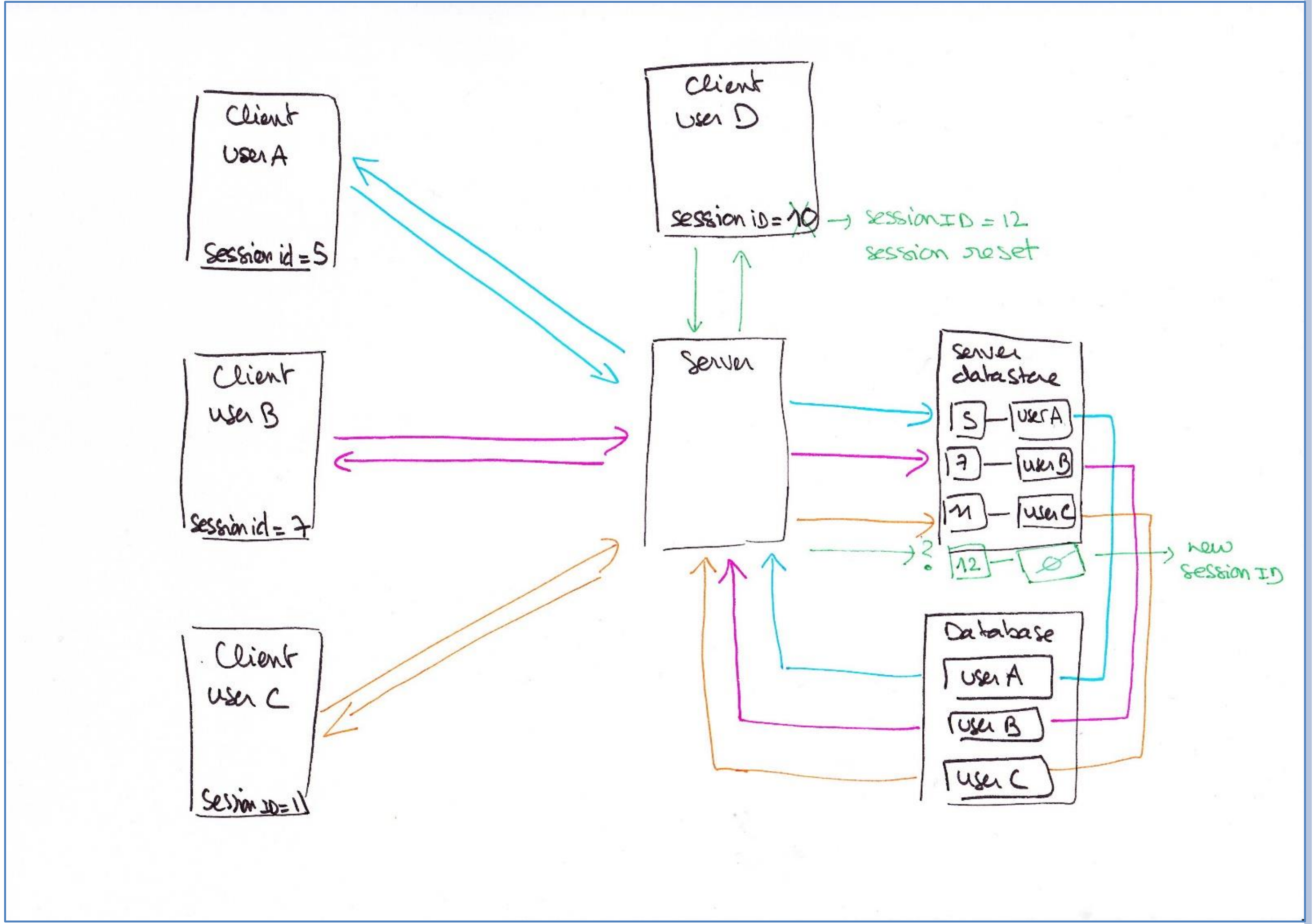
One Client



client side
server side
1
http request
cookie = [session_id = 5]
http response
cookie = [session_id = 5]
6
Client
user data in page
Server
User data in page
2
Server Datastore
5
user id
7
11
3
Creates personalized page
5
Code
getUserData (session('user_id'))
4
APP Database
User id
user Data
session('user_id') = server_datastore [5]['user_id']

Session and cookies

Multiple Clients



HTTP Methods

HTTP-GET

- GET requests
  - are only used to request data (not modify)
  - should never be used when dealing with sensitive data
  - have length restrictions
  - only used to send simple text data
- Data is sent as URL parameters that are usually strings of name and value pairs separated by &.

- Example
  www.example.com/action.php?**fname**=*John*&**lname**=Smith
  - The red parts in the URL are the GET parameters
  - and the green ones are the value of those parameters.

HTTP-GET

```
<?php
if( $_GET["fname"] || $_GET["lname"] ) {
    echo "Welcome ". $_GET['lname']. "<br />";
    echo "You are ". $_GET['lname'];

    exit();
}
?>
<html>
<body>

    <form action = "<?php $_PHP_SELF ?>" method = "GET">
        First Name: <input type = "text" name = "fname" />
        Last Name: <input type = "text" name = "lname" />
        <input type = "submit" />
    </form>

</body>
</html>
```

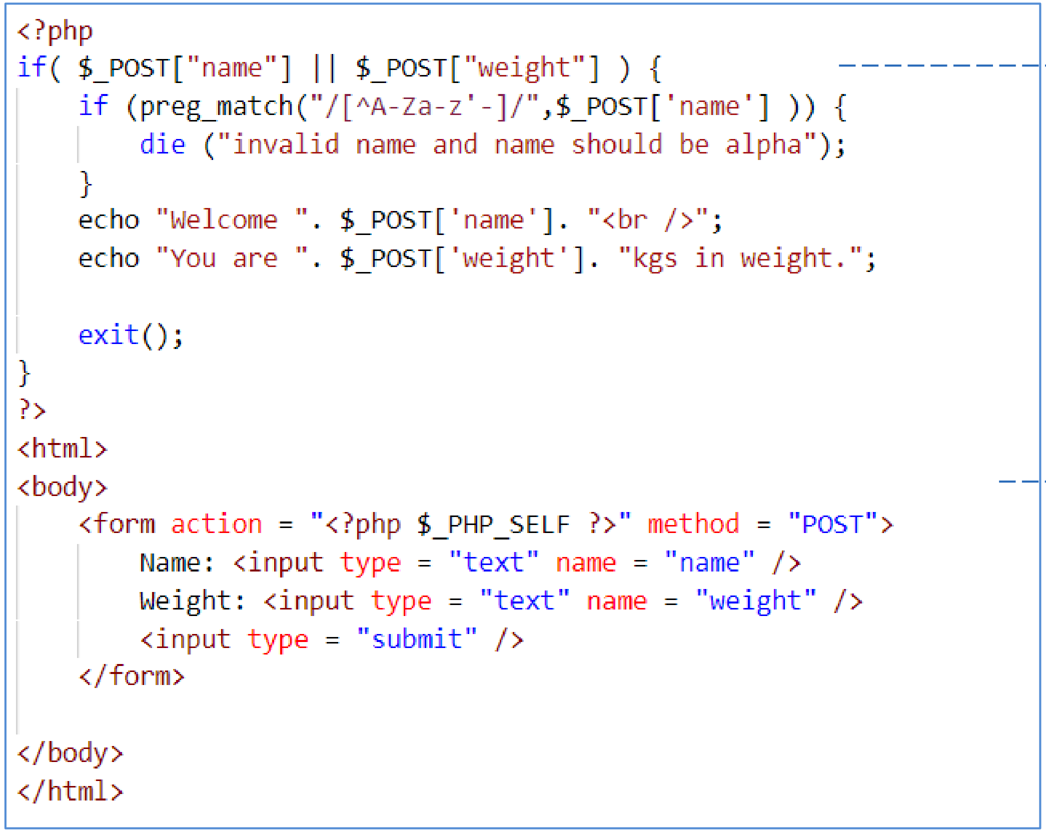
use $_GET to read data

method ="GET"

HTTP-POST

- POST requests
  - used to send data to a server to create a resource
  - not visible in the URL
  - have no length restrictions
  - data are stored in request body

HTTP-POST



```
<?php
if( $_POST["name"] || $_POST["weight"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
        die ("invalid name and name should be alpha");
    }
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['weight']. "kgs in weight.";

    exit();
}
?>
<html>
<body>
    <form action = "<?php $_PHP_SELF ?>" method = "POST">
        Name: <input type = "text" name = "name" />
        Weight: <input type = "text" name = "weight" />
        <input type = "submit" />
    </form>

</body>
</html>
```

use $_POST to read data

method ="POST"

HTTP-PUT

- PUT requests
  - used to send data to a server to update a resource
  - not visible in the URL
  - have no length restrictions
  - data are stored in request body

HTTP-DELETE

- DELETE requests
  - used to send data to a server to delete a resource or a file

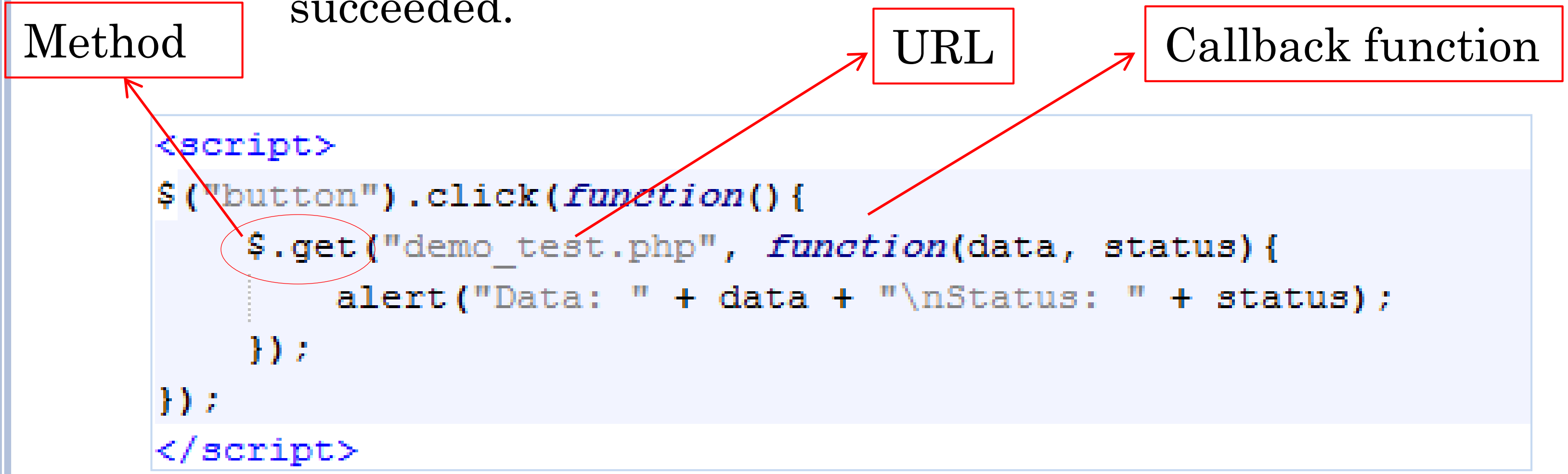
AJAX

AJAX

- AJAX = Asynchronous JavaScript and XML.
- AJAX is not a programming language; it is a technique used for accessing web servers from a web page.
- AJAX is mainly used to
  - Send data to a web server - in the background
  - Update a web page without reloading the page
- Examples of applications using AJAX:
  - Gmail, ie
    - send a new email
  - Youtube, ie
    - upload a new video
  - Facebook, ie
    - Post, like, …

AJAX

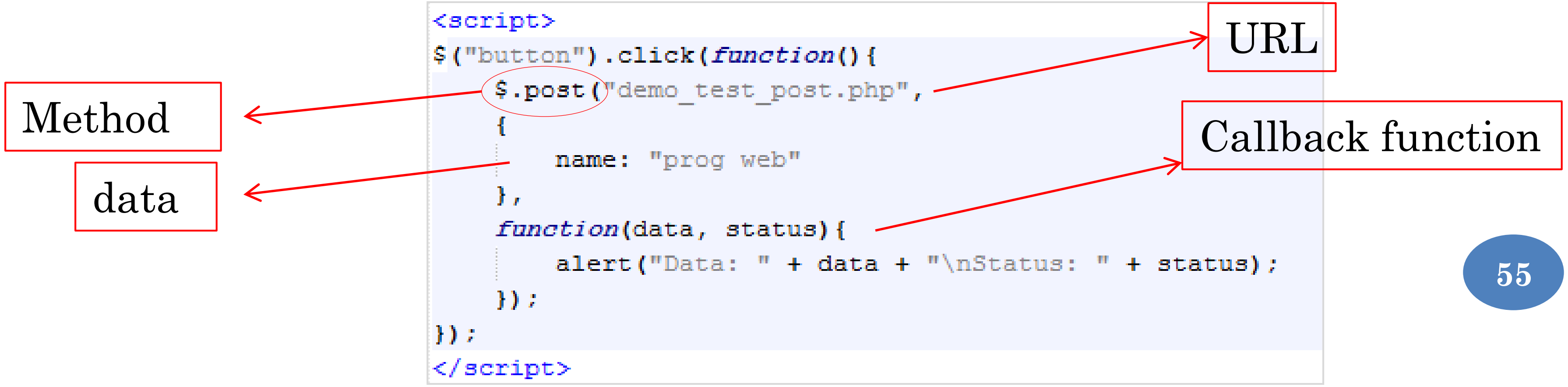
- Get: Request data from the server using an HTTP GET request
  - $.get(*URL*,*callback*);
    - *URL:* specifies the requested URL (link)
    - *callback:* is the name of a function to execute if the request succeeded.



```
<script>
$("button").click(function(){
    $.get("demo_test.php", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
</script>
```

AJAX

- Post: requests data from the server using an HTTP POST request
  - $.post(*URL,data,callback*);
    - *URL:* specifies the requested URL (link)
    - *data:* specifies the parameters to send with the request
    - *callback: i*s the name of a function to execute if the request succeeded

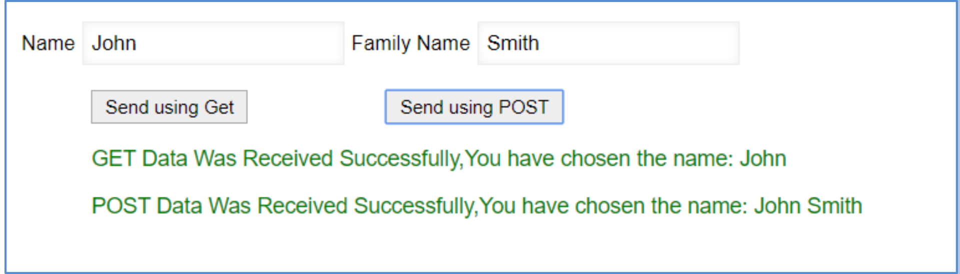


```
<script>
$("button").click(function(){
    $.post("demo_test_post.php",
    {
        name: "prog web"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
</script>
```

AJAX- Example

HTML

```
<head>
    <title>Simple Ajax Form</title>
    <script src="Scripts/jquery.min.js"></script>
    <script src="Scripts/ajax-test.js"></script>
</head>
<body>
    <form method="post" name="postForm">
        <ul>
            <li>
                <label for="name">Name</label>
                <input type="text" name="name" id="name" />
                <label for="name">Family Name</label>
                <input type="text" name="fname" id="fname" />
                <span class="error"></span>
            </li>
        </ul>
        <input type="button" value="Send using Get"  id="btnCheckNameGet"/>
        <input type="submit" value="Send using POST"  id="btnCheckNamePost"/>
    </form>
    <div id="success"></div>
</body>
```

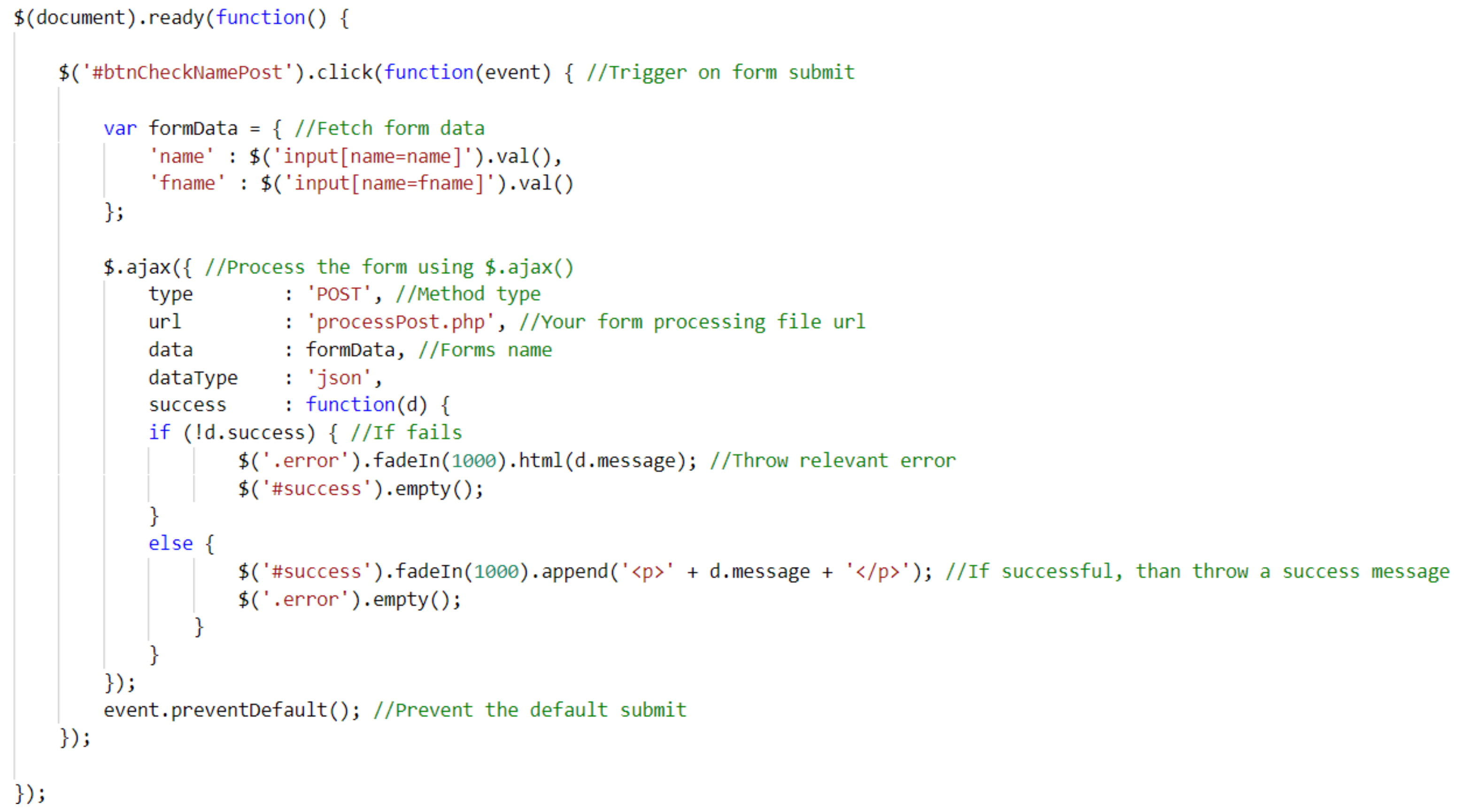


GET Data Was Received Successfully,You have chosen the name: John

POST Data Was Received Successfully,You have chosen the name: John Smith

AJAX- Example

jQuery - POST



```
$(document).ready(function() {

    $('#btnCheckNamePost').click(function(event) { //Trigger on form submit

        var formData = { //Fetch form data
            'name' : $('input[name=name]').val(),
            'fname' : $('input[name=fname]').val()
        };

        $.ajax({ //Process the form using $.ajax()
            type        : 'POST', //Method type
            url         : 'processPost.php', //Your form processing file url
            data        : formData, //Forms name
            dataType    : 'json',
            success     : function(d) {
            if (!d.success) { //If fails
                    $('.error').fadeIn(1000).html(d.message); //Throw relevant error
                    $('#success').empty();
            }
            else {
                    $('#success').fadeIn(1000).append('<p>' + d.message + '</p>'); //If successful, than throw a success message
                    $('.error').empty();
                }
            }
        });
        event.preventDefault(); //Prevent the default submit
    });

});
```

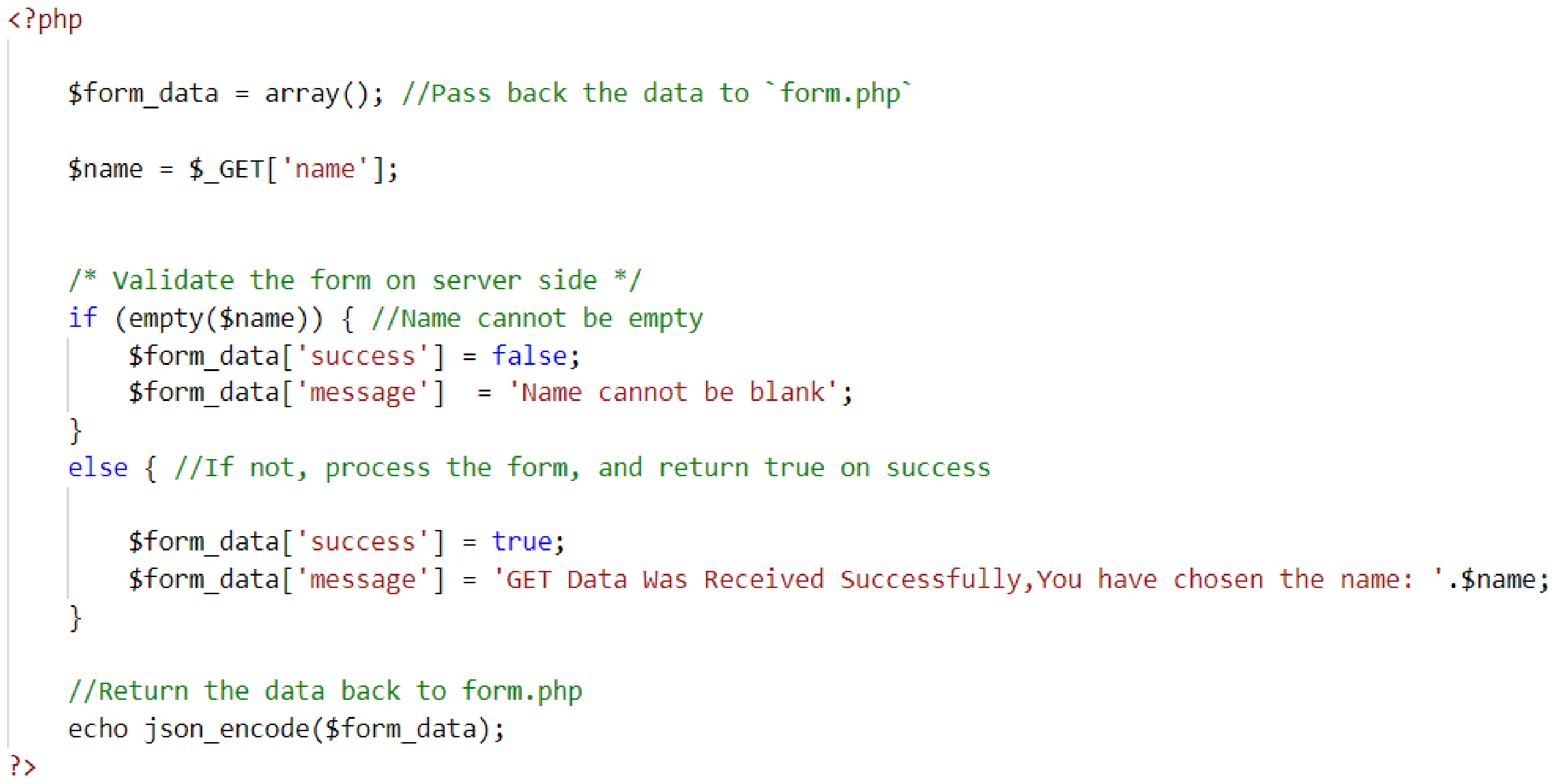
AJAX- Example

jQuery - GET

```
$('#btnCheckNameGet').click(function(event) {

    $.ajax({ //Process the form using $.ajax()
        type        : 'GET', //Method type
        url         : 'processGet.php?name='+$('input[name=name]').val(), //Your form processing file url
        //data      : formData, //Forms name
        dataType    : 'json',
        success     : function(data) {
            if (!data.success) { //If fails
                $('.error').fadeIn(1000).html(data.message); //Throw relevant error
                $('#success').empty();
            }
            else {
                $('#success').fadeIn(1000).append('<p>' + data.message + '</p>'); //If successful, than throw a success message
                $('.error').empty();
            }
        }
    });
});
```

AJAX- EXAMPLE

processGet.php



```
<?php

    $form_data = array(); //Pass back the data to `form.php`

    $name = $_GET['name'];


    /* Validate the form on server side */
    if (empty($name)) { //Name cannot be empty
        $form_data['success'] = false;
        $form_data['message']  = 'Name cannot be blank';
    }
    else { //If not, process the form, and return true on success

        $form_data['success'] = true;
        $form_data['message'] = 'GET Data Was Received Successfully,You have chosen the name: '.$name;
    }

    //Return the data back to form.php
    echo json_encode($form_data);
?>
```

AJAX- Example

processPost.php

```php
<?php

    $form_data = array(); //Pass back the data to `form.php`

    $name = $_POST['name'];
    $fname = $_POST['fname'];

    /* Validate the form on server side */
    if (empty($name)) { //Name cannot be empty
        $form_data['success'] = false;
        $form_data['message']  = 'Name cannot be blank';
    }
    else if (empty($fname)) { //FName cannot be empty
        $form_data['success'] = false;
        $form_data['message']  = 'Family Name cannot be blank';
    }
    else { //If not, process the form, and return true on success

        $form_data['success'] = true;
        $form_data['message'] = 'POST Data Was Received Successfully,You have chosen the name: '.$name.' '.$fname;
    }

    //Return the data back to form.php
    echo json_encode($form_data);
?>
```