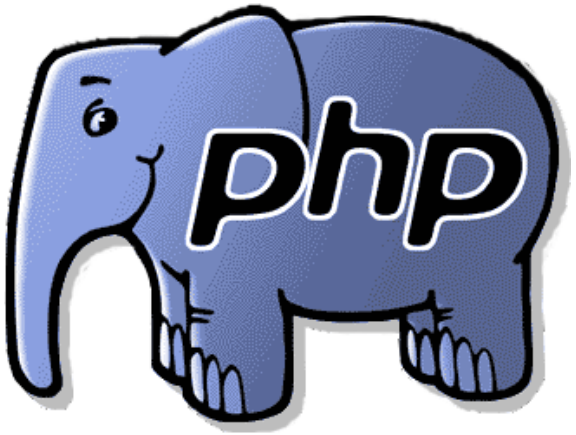
Partie V: PHP

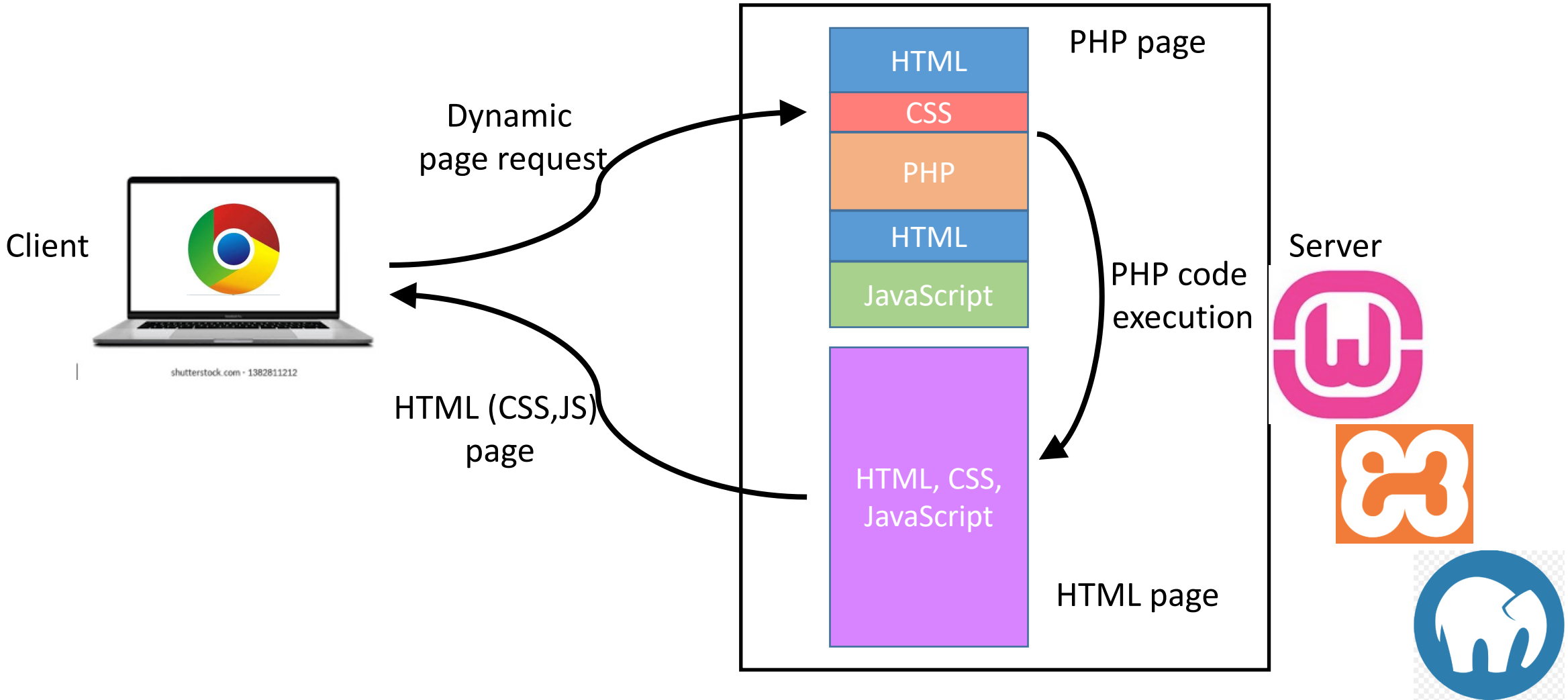
PHP – Introduction



- What? Server-side scripting language for creating interactive and dynamic web pages
  - Server-side like others: Python, Java, Perl, Ruby, ...
- Why?
  - To create dynamic pages (generated by the server on request)
  - To manage databases, collect data from forms
  - To manage files on the server
- A dynamic PHP page **(.php file)** can contain: HTML, CSS, JavaScript, and PHP (<?php ... code ... ?>)
- After execution on the server, an HTML file is generated and sent to the browser
- PHP tags can be placed anywhere in the HTML document

```
14 |     <img src="<?php $filePath ?>" alt="image">
```

PHP – Dynamic page generation



PHP – Basics (variables & types)

- Variables
  - **$** prefix denotes variables
  - **Globals**: outside functions (not recognized within functions)
  - Accessible in functions if preceded by **'global'** or **$GLOBALS['var']**
  - **Locals**: within functions (not recognized outside)
  - An uninitialized variable equals **NULL**
  - **Constants:** use **const** name=value; or **define**("name",val); (global)
- Types
  - string, int, float, bool, array, Object
  - var_dump(): gives the type and value of a variable
  - Type conversion is automatic

```
$x = "11";
$y = 5;
$y = $x / 2;
```

PHP – Basics (variables & types)

```
3   <body>
4   <?php
5   $a = "Hello world!";
6   $b = 5;
7   $c = 10.5;
8   define("d",10);
9   echo var_dump($a)."<br>"; //string(12) "Hello world!"
10  echo var_dump($b)."<br>"; #int(5)
11  echo var_dump($c)."<br>"; //float(10.5)
12  echo var_dump(d)."<br>"; //int(10)
13  echo var_dump($e)."<br>"; #NULL
14  $b = "*".$a;
15  echo var_dump($b)."<br>";//string(13) "*Hello world!"
16  $x = 5;
17  function myTest() {
18    $y = 3;
19    echo "<p>Variable x inside function is: $x</p>";//error
20    echo "<p>Variable y inside function is: $y</p>";//3
21  }
22  myTest();
23  echo "<p>Variable x outside function is: $x</p>";//5
24  echo "<p>Variable y outside function is: $y</p>";//error
25  ?>
26  </body>
```

PHP – Basics (simple statements)

- Operators
  - Arithmetic/Logic : +, -, *, /, %, **, &&, and, ||, or, !, xor, …
  - Incrementation: ++i, --i, i++, i--
  - Comparaison: ==, >=, <=, !=, ===, !==, <=>
  - Assignement: +=, *=, …, .=
    - Multiple: $name = $email = $gender = $comment = $website = "";
  - Arrauys: +, ==, ===, !=, !==
  - Others: ., ?:
- Display & Insert
  - **echo/print**: displays a value (numeric, text, HTML tag, etc.) in the HTML document
  - **include/require 'f'**: inserts the contents of the php file 'f' into the current file for reuse reasons, ex: include 'header.php'; require 'menu.php';
- Coments: //coment, #coment, /* coment multiline*/

PHP – Basics (simple statements)

```php
<?php
$a = $b = $c = "";
$a = "Hello world!";
$b = 5;
$c = --$b**2/2;
echo $c;//8
$d = $b<=>$c;
echo "compare b & c: ".$d;//-1
$a .="...";
echo $a;//ello world!...
$x = array(7,"5",2,3.0);
$y = array("7","5",2,3);
$z = array(6,"zero");
echo $x==$y;//true
echo $x===$y;//false
echo var_dump($z+$x);//6,"zero",2,3.0
?>
```

PHP – The basics (control statements)

- Conditionnel
  - **if** cond inst: usual conditionnel if
  - **if** cond inst **else** Inst: if with usual else
  - **if** cond **i**nst **elseif** cond inst **else i**nst: nesting if-else with elseif »
  - **switch** exp {**case** val: inst break; … **default**: inst}: usual switch
- Repetition
  - **for**(init;cond;incr) Inst: usual for loop
  - **foreach**(tab **as** val)/**foreach**(tab **as** idx=>val): for loop to iterate arrays
  - **while** cond Inst: usual while loop
  - **do** Inst **while** cond: usual do-while loop

PHP – The basics (control statements)

```php
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
foreach ($colors as $i =>$val) {
    echo "$i = $val <br>";
}
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

PHP – Basics (internal functions)

- Strings
  - **strlen**(str): str length
  - **str_word_count**(str): number of words in str
  - **strrev**(str): reverse the order of str
  - **strpos**(str, a): position of the 1st occurrence of a in str or false
  - **str_replace**(a,b,str): replace parts a with b in str
- Numbers
  - NaN, isNaN(val): non-numeric value and test for NaN
  - (int), (integer), intVal(nbr): casting to the integer type
- Math
  - pi(): PI value
  - min/max(a,b,c, …): min/max of a values list
  - abs/sqrt(x): absolute/square root value of x
  - round(x): integer closest to x
  - rand(a,b): random number between a and b

PHP – Basics (internal functions)

```php
<?php
$txt = "  C'est un exemple de texte simple. ";
echo strlen($txt)."<br>";//36
echo str_word_count($txt)."<br>";//6
echo strrev($txt)."<br>";//.elpmis etxet ed elpmexe nu tse'C
echo strpos($txt,"ex")."<br>";//11
echo str_replace("ex","",$txt)."<br>";//C'est un emple de tte simple.
echo var_dump(sqrt(-1))."<br>";//float(NAN)
$a = rand(5,6); $b = rand(-1,1); $c = rand(8,9);
$a = round($a) * pi();
$b = abs($b);
$c = sqrt($c);
echo "max: ".max($a,$b,$c);//max: 15.707963267949
?>
```

PHP – Basics (functions & Arrays)

- Fonctions
  - Up than **1000** internal functions
  - **Declaration**: function fname($a,…){code}
  - **Default value**: function f($p=val) {…}, a call f() ⇔ f(val)
  - **return**: if it returns a value, function f(…) {… return exp}
  - Use the **“&“** prefix to pass parameters by address, function f(&$p) {…}
- Arrays
  - $a = array(v1, v2, …) / $a[0]=v1; $a[1]=v2; …
  - $a = array(k1=>v1, k2=>v1, …) / $a[k1]=v1; $a[k2]=v2; … (associative arrays)
  - Count(a): Size of the a array
  - sort(), rsort(), asort(), ksort(), arsort(), krsort(): ascending/descending numeric/alphanumeric sort according to key/value (associative)

PHP – Basics (functions & Arrays)

```php
<?php
$m = "Un exemple de message.";
function show(&$msg){
    echo $msg;
    $msg .= "*";
}
function msize($msg=""){
    show($msg);
    return strlen($msg);
}
echo "(".msize($m).")";//Un exemple de msg.(18)
echo "(".msize().")";//(1)
```

```php
$a = array(23,2,3.0,11);
$b[0]="Un"; $b[1]="exemple"; $b[2]="de"; $b[3]="msg";
$c = array("POO"=>4, "DW"=>2, "TL"=>3);
echo "courses weights: ".$c["POO"]." ".$c["DW"]." ".$c["TL"];
echo count($a);//4
echo count($b);//4
echo count($c);//3
sort($a);//[2,3,11,23]
asort($c);//["DW"=>2,"TL"=>3,"POO"=>4]
ksort($c)//["DW"=>2,"POO"=>4,"TL"=>3]
?>
```

PHP - Superglobals

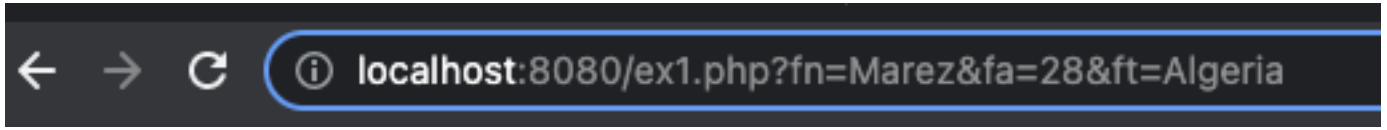
- What?
  - Set of predefined variables (associative arrays) in PHP always available and accessible everywhere
- Examples
  - **$GLOBALS**: used to access global variables from anywhere.
  - **$_SERVER**: information about the server, paths, and script locations (PHP_SELF, SERVER_NAME, SERVER_ADDR, REQUEST_METHOD, SERVER_PORT, SCRIPT_NAME, SCRIPT_URI, ...).
  - **$_REQUEST**: used to collect data after submitting a form.
  - **$_POST**: used to collect data after submitting a form with method="post". It is also widely used for passing variables.
  - **$_GET**: used to collect data after submitting a form with method="get". It can also collect data sent in the URL.
  - **$_COOKIE**: used to store cookies sent by the browser
  - **$_FILES**: used to access files uploaded via POST

PHP - Superglobals

```php
<?php
    $cname = "user";
    $cval = "BESNACI";
    setcookie($cname, $cval);
?>
<html>
<body>
    <form method="post" action="phpex.php">
        Name: <input type="text" name="fn">
        <input type="submit">
    </form>
<?php
$x = 7; $y = 2;
function myF() {
    $GLOBALS['x']++;
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];}
myF();
echo $z."<br>";//10
echo $_SERVER['PHP_SELF']."<br>";///phpex.php
echo $_SERVER['SERVER_NAME']."<br>";//localhost
echo $_SERVER['GATEWAY_INTERFACE']."<br>";//CGI/1.1
$fname = $_REQUEST['fname'];
if (!empty($fname)) echo $fname."<br>";//value
if(isset($_COOKIE[$cname]))
    echo "$cname: " . $_COOKIE[$cname];//user: BESNACI
?>
```

PHP – Forms

- After form data is submitted, it will be accessible on the server through the associative arrays **$_GET** and **$_POST**.
- Their keys are the names of the form elements (the '**name**' **attribute**), and the values are the user's input data.
- The content of **$_GET** is passed to the current script via URL parameters, while that of **$_POST** is passed via the HTTP POST method.
- Data sent via **GET** is visible to everyone (sending non-sensitive data), and its size is limited to 2000 characters.



- Data sent via **POST** is not visible (embedded in HTTP requests), hence secure, and its size is unlimited.

PHP – Send/recieve form data

form.html

```
<!DOCTYPE html>
<html>
<body>
    <h2>Add a new player</h2>
    <form method="post" action="ex1.php">
    <label for="i1">Name:</label>
        <input type="text" id="i1" name="pn"><br>
    <label for="i2">Age:</label>
        <input type="text" id="i2" name="pa"><br>
    <label for="i3">Team:</label>
        <input type="text" id="i3" name="pt"><br>
        <input type="submit" name="ps" value="Add player">
</form>
</body>
</html>
```

Form elts names

ex1.php

```
<!DOCTYPE html>
<html>
<body>
    <?php
        $name = $_POST["pn"];
        $age = $_POST["pa"];
        $team = $_POST["pt"];
    ?>
    <h1>The player <?php echo $name?></h1>
    <p><?php echo $name?> aged of <?php echo $age?> is one of the
    best players in the world participating in world championships.
    He is currently playing for the <?php echo $team?> team</p>
</body>
</html>
```

Key	'pn'	'pa'	'pt'
Value	'Marez'	30	'Algeria'

$_POST content

Add a new player

Name: Marez
Age: 30
Team: Algeria
Add player

Form elts entries

The player Marez

Marez aged of 30 is one of the best players in the world participating in world championships. He is currently playing for the Algeria team

Dynamic PHP parts

PHP – Forms validation

- Clean and process form data before any use for security reasons by calling these functions:
  - **trim**(d): removes leading and trailing spaces, tabs, newlines
  - **stripslashes**(d): removes backslashes '\'
  - **htmlspecialchars**(d): prevents the effect of malicious insertions
- Check for empty fields with **empty**(d)
- Check syntax with **preg_match**(model, d) → regex functions
- Access field values via **$_REQUEST**['cname'], **$_GET**['cname'], or **$_POST**['cname']
- Use **isset**('name') to check if an element has been set before use

PHP – Forms validation

form.html

```html
<!DOCTYPE html>
<html>
<body>
    <h2>My Form</h2>
    <form method="post" action="ex1.php">
        Name: <input type="text" id="i1" name="ft"><br>
        File: <input type="file" id="i2" name="ff"><br>
        <input type="radio" id="i3" name="fr">Choice1<br>
        <input type="radio" id="i4" name="fr">Choice2<br>
        <input type="radio" id="i5" name="fr">Choice3<br>
        Category: <select id="s1" name="fsel">
            <option value="val1">Value1</option>
            <option value="val2">Value2</option>
            <option value="val3">Value3</option>
        </select><br>
        <input type="checkbox" id="i6" name="fc1" value="option1">Option1<br>
        <input type="checkbox" id="i7" name="fc2" value="option2">Option2<br>
        Comment:<br>
        <textarea id="ta1" name="fta" cols="30" rows="5"></textarea><br>
        <input type="submit" id="i9" name="fs" value="Submit">
</form>
</body>
</html>
```

ex1.php

```php
<?php
if (!empty($_POST["ft"])) {
    $name = test_input($_POST["ft"]);
    if (preg_match("/^[a-zA-Z-' ]*$/",$name))
        echo "Name: $name";}
if (!empty($_POST["ff"])) {
    $file = test_input($_POST["ff"]);
    echo "File: $file";}
$choice = test_input($_POST["fr"]);
echo "File: $file";
$categ = test_input($_POST["fsel"]);
echo "Category: $categ";
$opt = array('option1'=>FALSE, 'option2'=>FALSE,
'option3'=>FALSE);
$o = test_input($_POST["fc1"]);
if(isset($o)) $opt[$o] = TRUE;
$o = test_input($_POST["fc2"]);
if(isset($o)) $opt[$o] = TRUE;
echo "Options: <br>";
foreach($opt as $k=>$v)
    echo "$k=$v";
$comment = test_input($_POST["fta"]);
echo "Comment: $comment";

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;}
?>
```

PHP – Data bases

- Connexion: $c = mysqli_connect(server, user, pw)
- Closing: mysqli_close($c)
- Running SQL queries: mysqli_query($c,$sql), tel que:
  - DB Creation: $sql = "**CREATE DATABASE** database"
  - Table Creation: $sql = "**CREATE TABLE** table (key1:type, …)"
  - Insertion: $sql = "**INSERT INTO** table (key1, …) **VALUES** (val1, …)"
  - Deletion: $sql = "**DELETE FROM** table **WHERE** condition"
  - Updating: $sql = "**UPDATE** table **SET** key=val **WHERE** condition"
  - Selecting: $sql = "**SELECT** key1, … **FROM** table1, … **WHERE** condition"

PHP – Data bases (exemple)

```php
<?php
    $server = "localhost";
    $user = "user";
    $password = "";
    $c = mysqli_connect($server, $user, $password);
    if (!$c) {die("Connection failed: ".mysqli_connect_error());}
    $sql = "CREATE DATABASE myDB";
    if (mysqli_query($c, $sql)) {
    $sql = "CREATE TABLE Person (id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
            name VARCHAR(30) NOT NULL,email VARCHAR(50),)";
    if (mysqli_query($c, $sql)) {
    $sql = "INSERT INTO Person(name, email)VALUES('Ali', 'ali@ex.com')";
    mysqli_query($c, $sql);
    $sql = "INSERT INTO Person(name, email)VALUES('Moh', 'moh@ex.com')";
    mysqli_query($c, $sql);
    $sql = "INSERT INTO Person(name, email)VALUES('Sami', 'sami@ex.com')";
    mysqli_query($c, $sql);
    $sql = "UPDATE Person SET name='Rami' WHERE id=2";
    mysqli_query($c, $sql);
    $sql = "DELETE FROM Person WHERE id=3";
    mysqli_query($c, $sql);
    $sql = "SELECT name, email, FROM Person";
    $result = mysqli_query($c, $sql);
    if (mysqli_num_rows($result) > 0) {
      while($row = mysqli_fetch_assoc($result)) {
        echo "Name: ".$row["name"]." - Email: ".$row["email"]."<br>";
      }
    }}}
    else echo "Error creating database: ".mysqli_error($conn);
    mysqli_close($c);
?>
```

PHP – Data bases (SELECT query)

- Due to certain incidents and issues, queries may fail to execute, and in such cases **mysqli_query**() returns **FALSE**.
- Unlike other queries, **mysqli_query**() produces a list of data rows with the **SELECT** query.
- After storing it, **$result = mysqli_query($c, $sql)**, it can be used as follows:
  - Check if the result is not empty: mysqli_num_rows($result) > 0
  - Iterate through the result list: while($row = mysqli_fetch_assoc($result))
  - Each result row is then accessible via $row['name']

```
$sql = "SELECT name, email, FROM Person";
$result = mysqli_query($c, $sql);
if (mysqli_num_rows($result) > 0) {
  while($row = mysqli_fetch_assoc($result)) {
    echo "Name: ".$row["name"]." - Email: ".$row["email"]."<br>";
  }
```