PHP(Hypertext Preprocessor)

BY MADI

Khan Omar

https://github.com/KOMAR-7

INTRODUCTION

Khan Omar

- What is PHP?
- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

- ***What is a PHP File?***
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

INTRODUCTION

Khan Omar

- What Can PHP Do?
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

INTRODUCTION

Khan Omar

- Why PHP?
- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

INTRODUCTION

Khan Omar

What do I need?

- To start using PHP, you can:
- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

- If your server has activated support for PHP you do not need to do anything.
- Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
- You do not need to compile anything or install any extra tools.
- Because PHP is free, most web hosts offer PHP support.

INTRODUCTION

Khan Omar

Basic Php Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:

  ```
  <?php
  // PHP code goes here
  ?>
  ```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.

On Server

```
<html>
<head> <title>Welcome</title> </head>
<body>
<?
   echo "Hello";
   print "<br />";
   echo "<b>I'm here..</b>";
?>
</body>
</html>
```

```
<html>
<head> <title>Welcome</title> </head>
<body>
Hello<br /><b>I'm here..</b></body>

</html>
```

Khan Omar

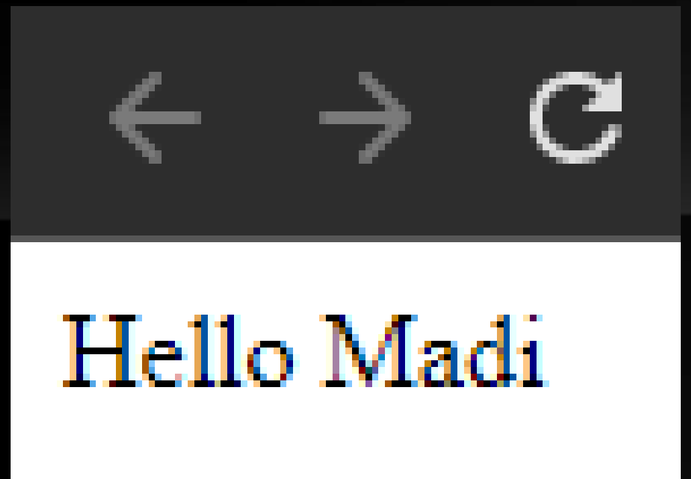
INTRODUCTION

Khan Omar

Example1:

```
<!doctype html>
<html>
   <body>
 <h1><?php echo "Hello Madi";?></h1>
   </body>
</html>
```

Save the file with .php extension



How to start with PHP

Khan Omar

STEP 1

Install XAMMP

STEP 2

Go to c drive, then xammp folder, and go to ht docs. C:\xampp\htdocs

STEP 3

Create a folder and create index.php and write your code

STEP 4

Start the xammp 'apache' and go to chrome and type: localhost/foldername

1

Step1

Go to chrome type Xammp download "apachefriends.org/download.html"

2

Step 2

Download XAMMP based on your computer specification and install

3

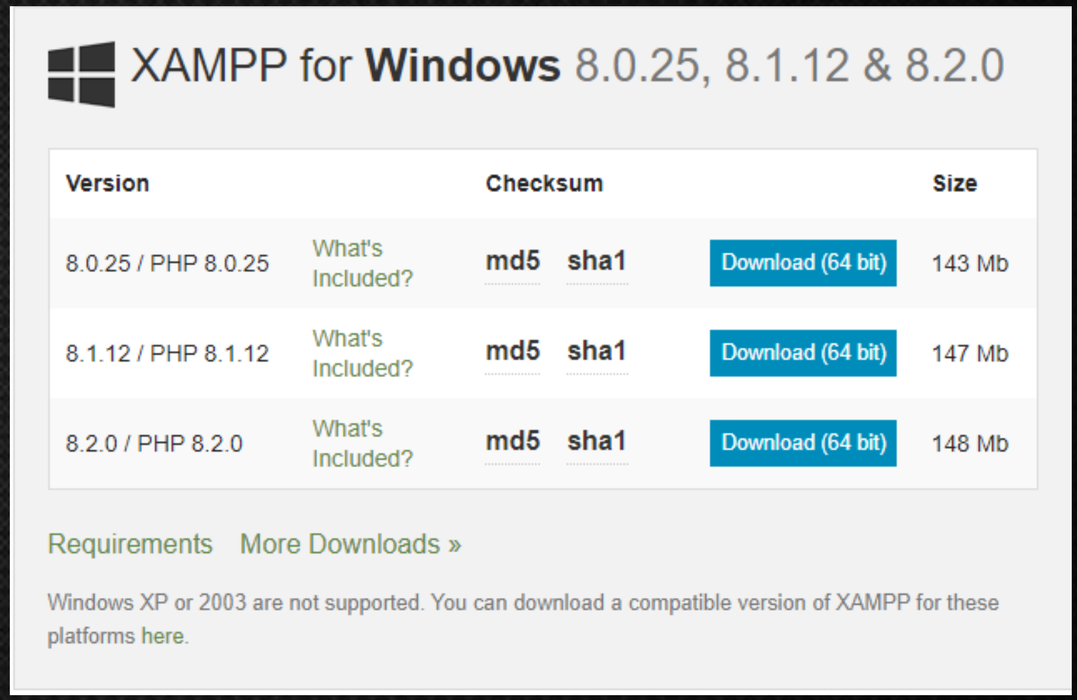
Step 3

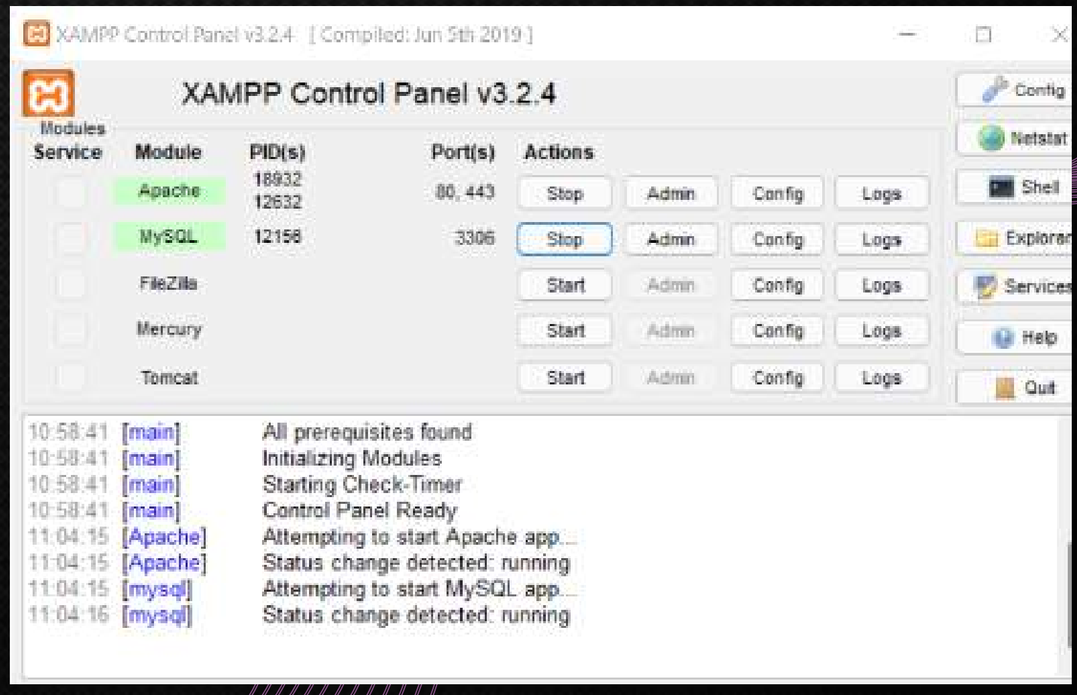
After completetion of XAMMP search Xammp in your Taskbar and open

3

Step 4

Start Apache and MySQL and start your journey!!!





Fasten your seat belt, we will start our journey to PHP!!!

Khan Omar

1

Lets start

Start the apache from xammp.
Open the htdocs folder C:\xampp\htdocs

2

Create a folder

Create a folder named: 'phpTut'
(You can name yours)

3

Create a file

Open the folder in your IDE and create a new file: index.php

FIRST CODE

Khan Omar

-filename: index.php
Code:

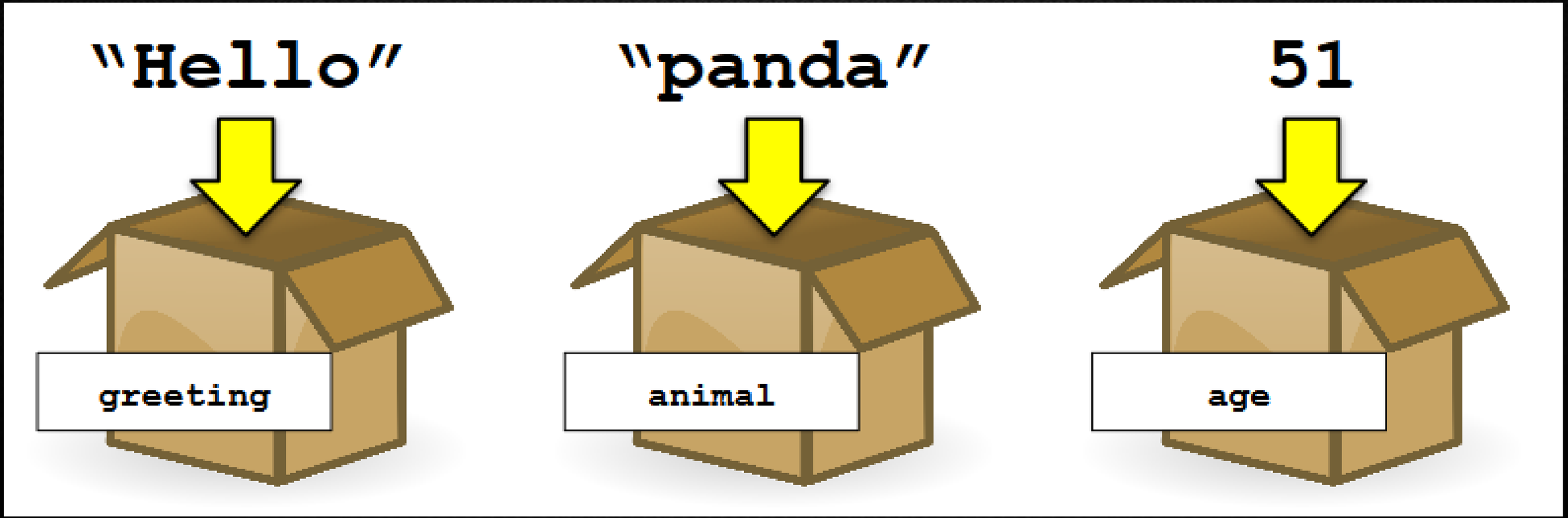
```
<!doctype html>
<html>
 <body>
 <h1><?php echo "Hello Madi";?></h1>
 </body>
</html>
```

-After writting the code save and go to chrome type:
'localhost/phpTut'

VARIABLE

Khan Omar

> Variables are used for storing values, like text strings, numbers or arrays.



VARIABLE

Khan Omar

> When a variable is declared, it can be used over and over again in your script.
> All variables in PHP start with a $ sign symbol.
> The correct way of declaring a variable in PHP: $var_name = value;
> In PHP, a variable does not need to be declared before adding a value to it.
> In the example above, you see that you do not have to tell PHP which data type the variable is.
> PHP automatically converts the variable to the correct data type, depending on its value.
> A variable name must start with a letter or an underscore "_" -- not a number
> A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString)

// COMMENT

Khan Omar

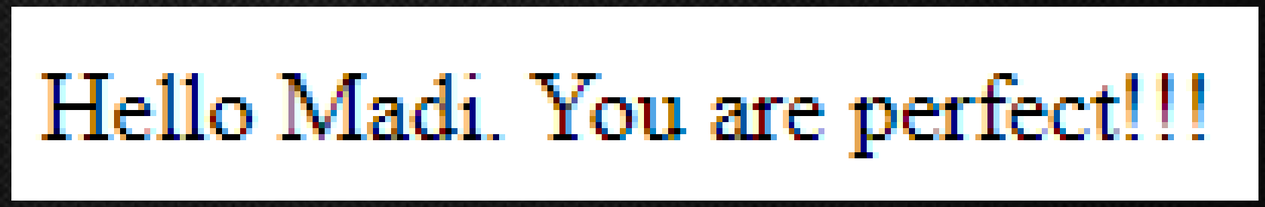
- In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters.
- In PHP, we use
  // to make a single-line comment or
  /* and */ to make a large comment block.
- Comment is not executed. It's just for making notes.
- Ex.
  // This is single line comment
  /*
  This is
  multiline comment */

CONCATI.NATION

Khan Omar

> The concatenation operator (.) is used to put two string values together.
> To concatenate two string variables together, use the concatenation operator:

```
<?php
    $txt1 = "Hello Madi.";
    $txt2 = "You are perfect!!!";
    echo $txt1 . " " . $txt2;
?>
```



If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings

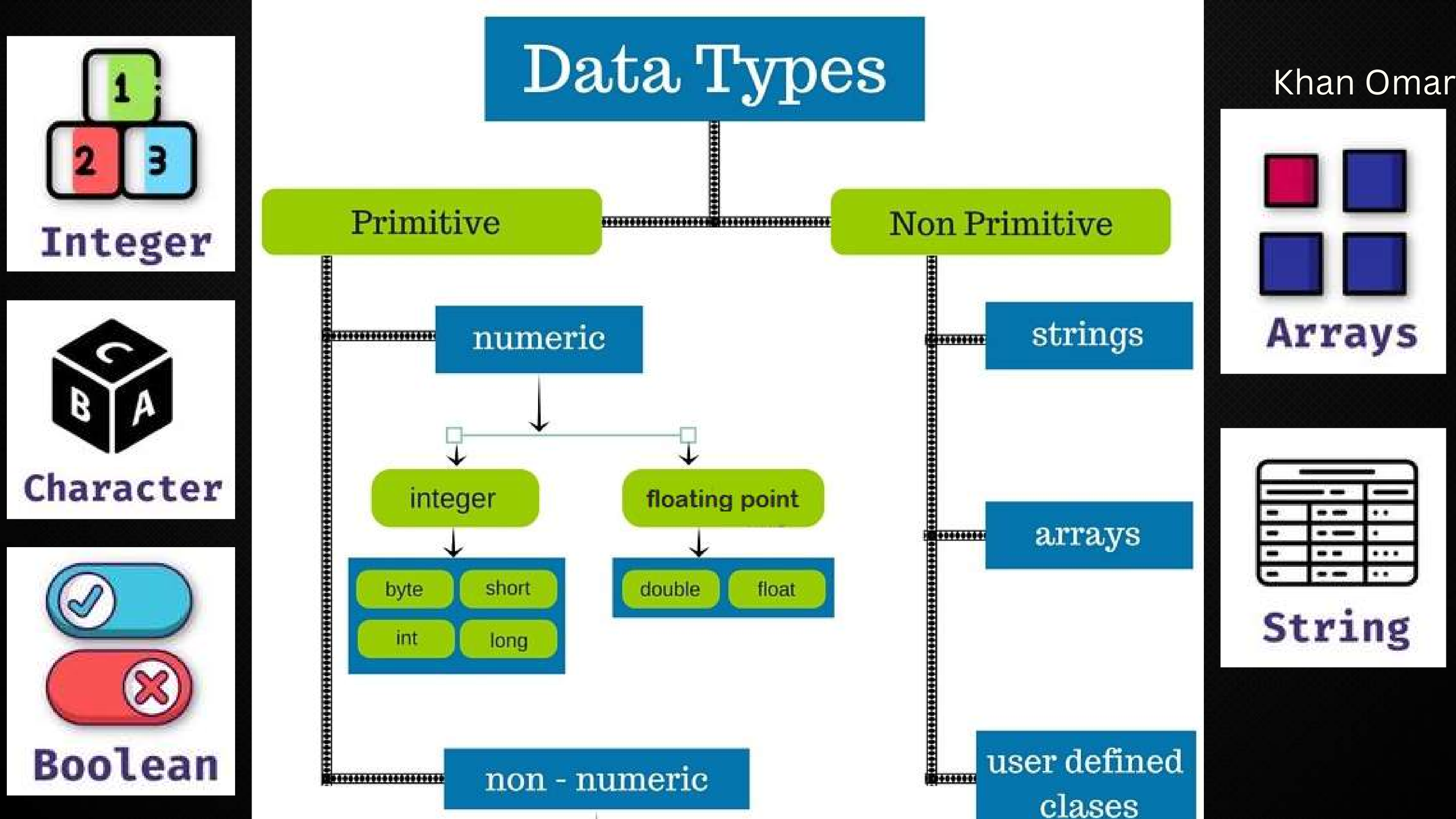
DATA TYPE

Khan Omar

- A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.

- Data type is the classification of data into a category.
  - Alphanumeric characters are classified as strings.
  - Whole numbers are classified as integers.
  - Numbers with decimal points are classified as floating points.
  - True or false values are classified as Boolean.
  - Collection of similar types of elements are called Array.
  - An object is a specific instance of a class that serves as templates for objects.

As told earlier, PHP determines the data types by analyzing the values of the data supplied.


Data Types
Primitive
Non Primitive
numeric
integer
floating point
byte
short
int
long
double
float
non - numeric
strings
arrays
user defined clases
Integer
Character
Boolean
Arrays
String
Khan Omar

DATA TYPE

Khan Omar

```php
<?php
// PHP String
$x = "Hello world!";
echo $x;

//PHP Integer
$x = 5985;
var_dump($x);

//PHP Float
$x = 10.365;
var_dump($x);

//PHP Boolean
$x = true;
$y = false;
```

```php
//PHP Array
$cars =
array("Volvo","BMW","Toyota");
var_dump($cars);

//PHP Object
 class Car {
 function Car() {
 $this->model = "VW";
 }
 }

 // create an object
 $herbie = new Car();
```

```php
// show object properties
 echo $herbie->model;

//PHP NULL Value
$x = "Hello world!";
$x = null;
var_dump($x);
```

The var_dump() function is used to dump information about a variable. This function displays structured information such as type and value of the given variable.

PHP CONSTANT

Khan Omar

Just by the name, we can understand that constant is something whose value cannot be changed. It is basically an identifier. A good practice is always to define it in the starting.

```
<?php
// Syntax to define a constant is Define("Name","Value","Case-insensitive")
// For case-insensitive, sensitive is default
define("Tau","6.28318530718","true");
echo tAU;
// Fun fact: Tau = Pi x 2
?> Output => 6.28318530718
```

OPERATORS

Khan Omar

Operators are used to perform operations on variables or values
Operators are used to operate on values. There are four classifications of operators:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators


```
$variable1 = 5;
$variable2 = 2;
echo $variable1;
echo $variable2;
```

OPERATORS

Khan Omar

// Arithmetic Operators

```php
<?php

$x = 10;
$y = 4;
echo "$x + $y" . "<br>"; // Outputs: 14
echo "$x - $y" . "<br>"; // Outputs: 6
echo "$x * $y" . "<br>"; // Outputs: 40
echo "$x / $y" . "<br>"; // Outputs: 2.5
echo "$x % $y" . "<br>"; // Outputs: 2

?>
```

OPERATORS

Khan Omar

```
// Assignment Operator
<?php

$x = 10;
echo $x . "<br>"; // Outputs: 10

$x = 20;
$x += 30;
echo $x . "<br>"; // Outputs: 50

$x = 50;
$x -= 20;
echo $x . "<br>"; // Outputs: 30

$x = 5;
$x *= 25;
echo $x . "<br>"; // Outputs: 125

$x = 50;
$x /= 10;
echo $x . "<br>"; // Outputs: 5

$x = 100;
$x %= 15;
echo $x . "<br>"; // Outputs: 10
?>
```

OPERATORS

Khan Omar

```
//Comparision Operator
  <?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z);  // Outputs: boolean true
echo "<br>";
var_dump($x === $z); // Outputs: boolean false
echo "<br>";
var_dump($x != $y);  // Outputs: boolean true
echo "<br>";
var_dump($x !== $z); // Outputs: boolean true
echo "<br>";
var_dump($x < $y);   // Outputs: boolean
true
echo "<br>";
var_dump($x > $y);   // Outputs: boolean
false
echo "<br>";
var_dump($x <= $y);  // Outputs: boolean
true
echo "<br>";
var_dump($x >= $y);  // Outputs: boolean
false
echo "<br>";
?>
```

OPERATORS

Khan Omar

```
//Logical Operator
<?php
 $a = $x >= $y
  $b = $x <= $y

  echo $a and $b . " this returns true if both, a and b are true <br>";
  echo $a or $b . " this returns true if either of them (or both) are true <br>";
  echo $a xor $b . " this returns true if either of them are true, but not both";
  echo $a && $b .  " this returns true if both, a and b are true <br>";
  echo $a || $b . " this returns true if either of them are true <br>";
  echo !$b . " this returns true if b is not true <br>";
?>
```

OPERATORS

Khan Omar

```
// Increment/Decrement operators
<?php

 $x = 24;
 $y = 3;
 echo ++$x . " this increments x by 1 and then returns x <br>";
 echo $x++ . " this returns x and increments x by 1 <br>";
 echo --$x . " this decrements x by 1 and then returns x <br>";
 echo $x-- . " this returns x and decrements x by 1 <br>";

?>
```

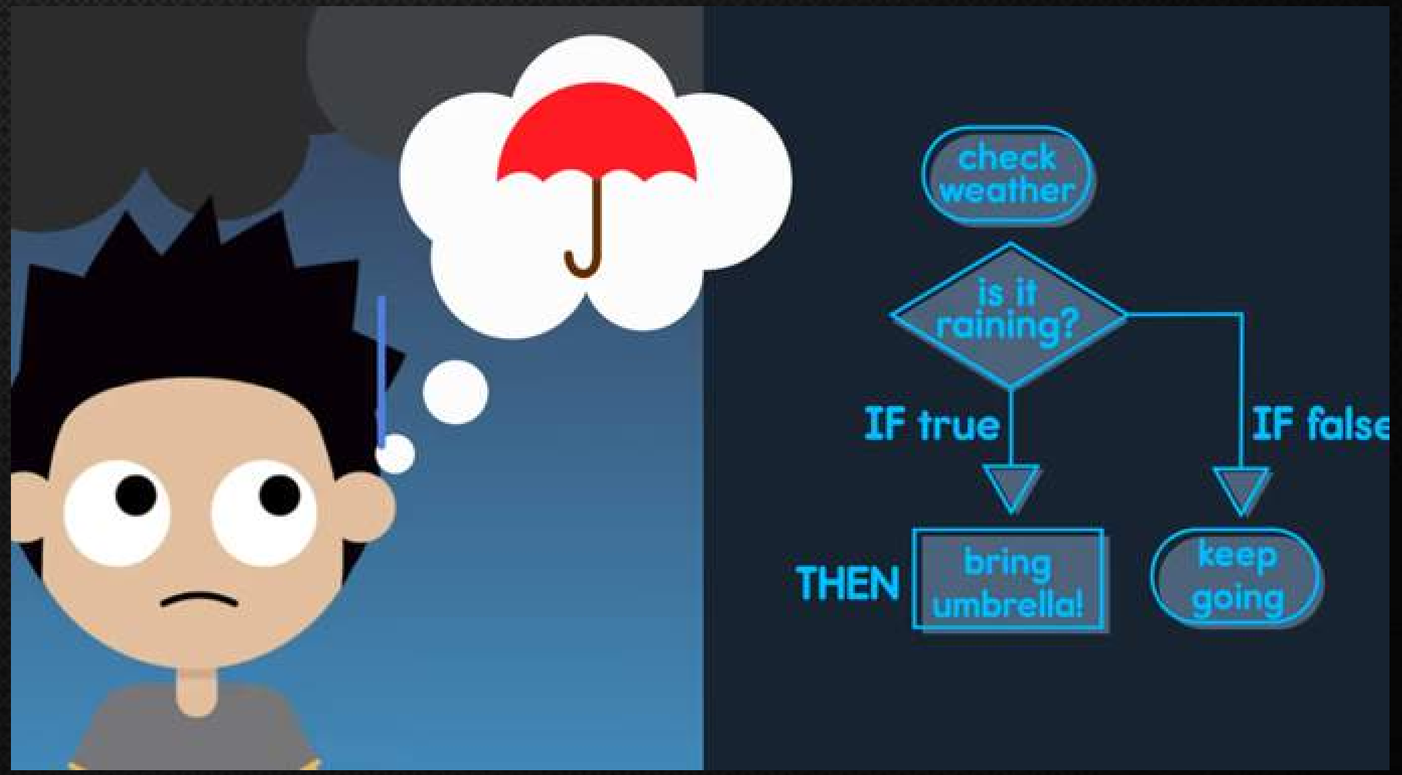
CONDITIONAL STATEMENT

Khan Omar

A conditional statement tells a program to execute an action depending on whether a condition is true or false.

> Very often when you write code, you want to perform different actions for different decisions.

> You can use conditional statements in your code to do this.



CONDITIONAL STATEMENT

Khan Omar

> if statement - use this statement to execute some code only if a specified condition is true

> if...else statement - use this statement to execute some code if a condition is true and another code if the condition is false

> if...elseif....else statement - use this statement to select one of several blocks of code to be executed

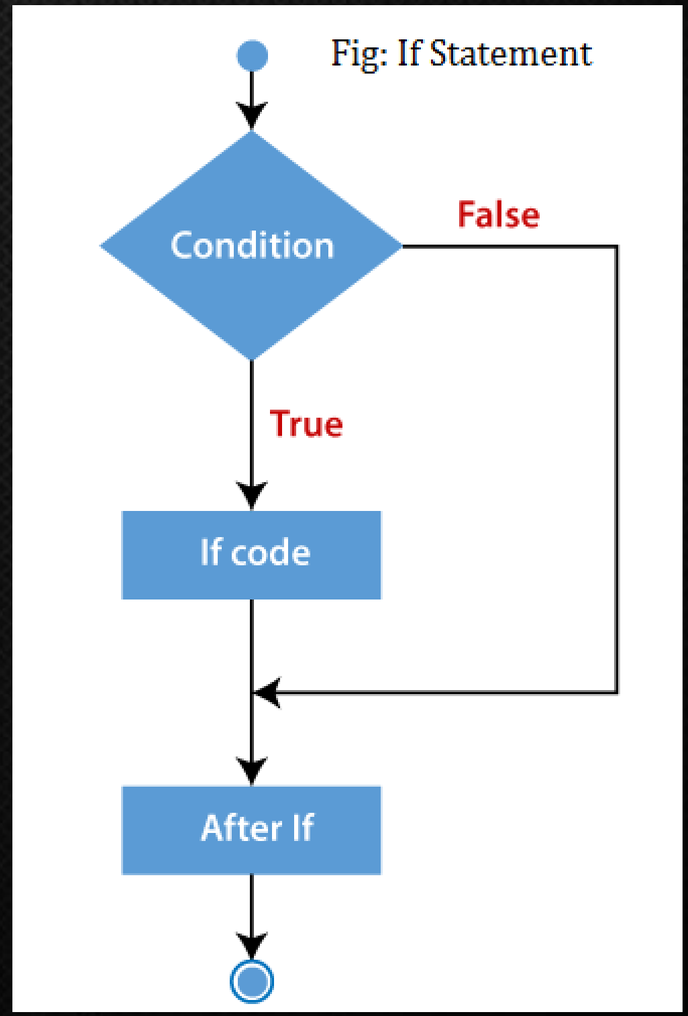
> switch statement - use this statement to select one of many blocks of code to be executed

if STATEMENT

Khan Omar

This statement executes the block of code inside the if statement if the expression is evaluated as True.

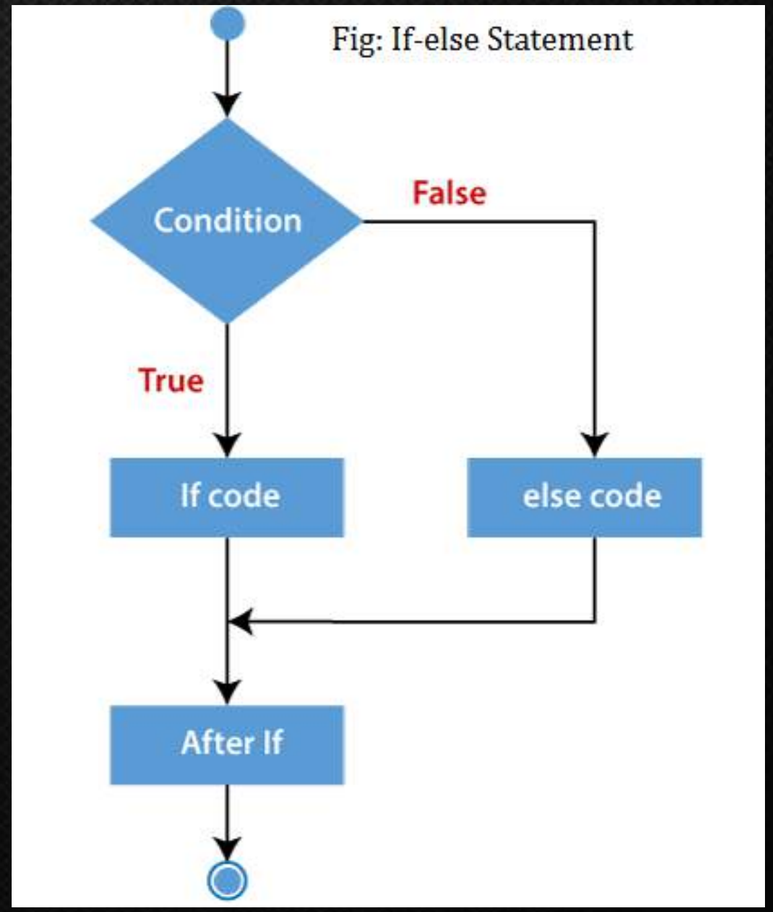
```
<?php
$x = "22";
if ($x < "20") {
  echo "Hello World!";
}
?>

Output =>
// Nothing will be print
```



if-else STATEMENT

Khan Omar

This statement executes the block of code inside the if statement if the expression is evaluated as True and executes the block of code inside the else statement if the expression is evaluated as False.




```
<?php
$x = "22";
if ($x < "20") {
  echo "Less than 20";
} else {
  echo "Greater than 20";
}
?>
```

Output => Greater than 20

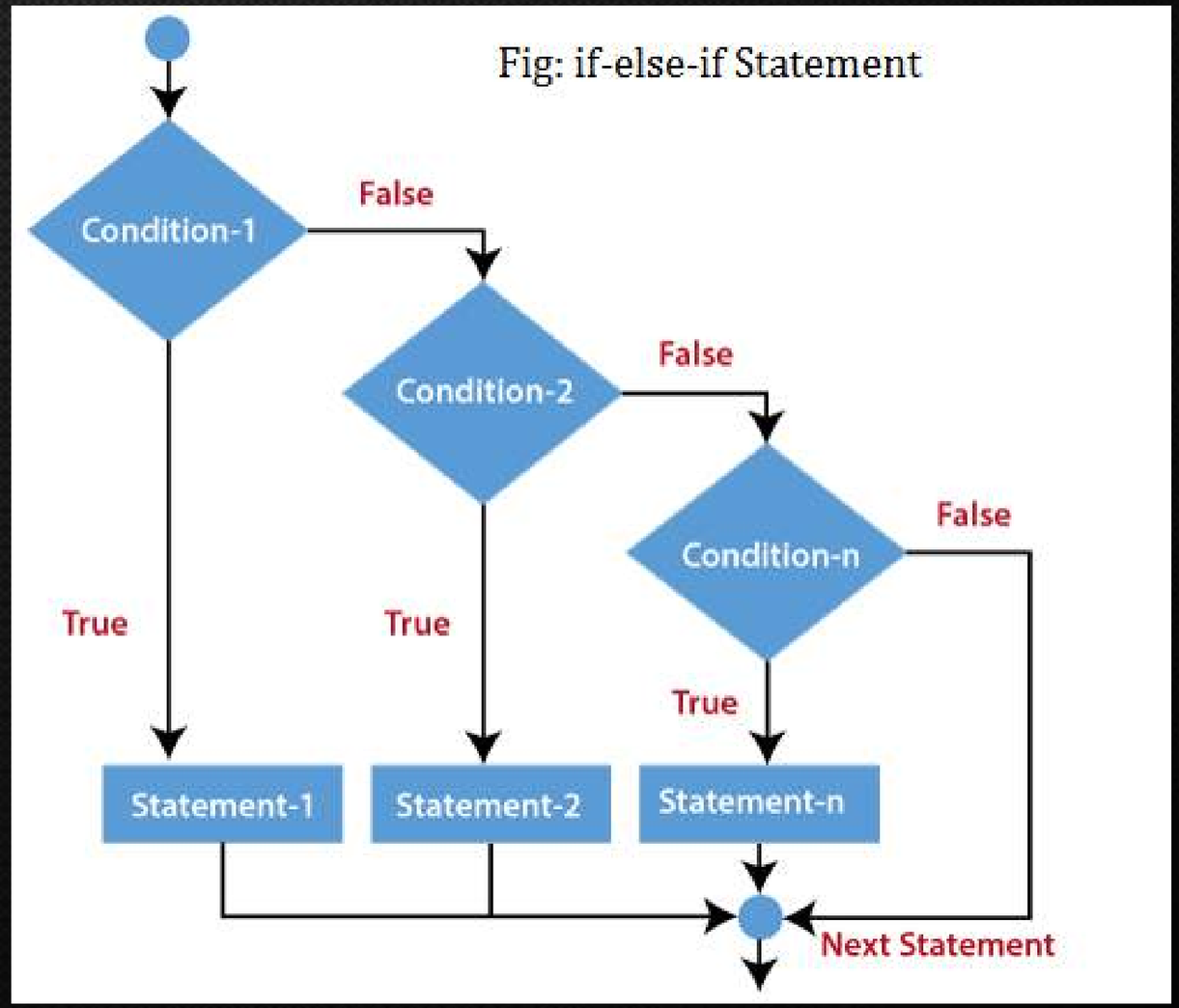
if-else if ladder STATEMENT

Khan Omar

This statement executes different expressions for more than two conditions.

```
<?php
$x = "22";
if ($x == "22") {
  echo "correct guess";
} else if ($x < "22") {
  echo "Less than 22";
} else {
  echo "Greater than 22";
}
?>
```

Output => correct guess



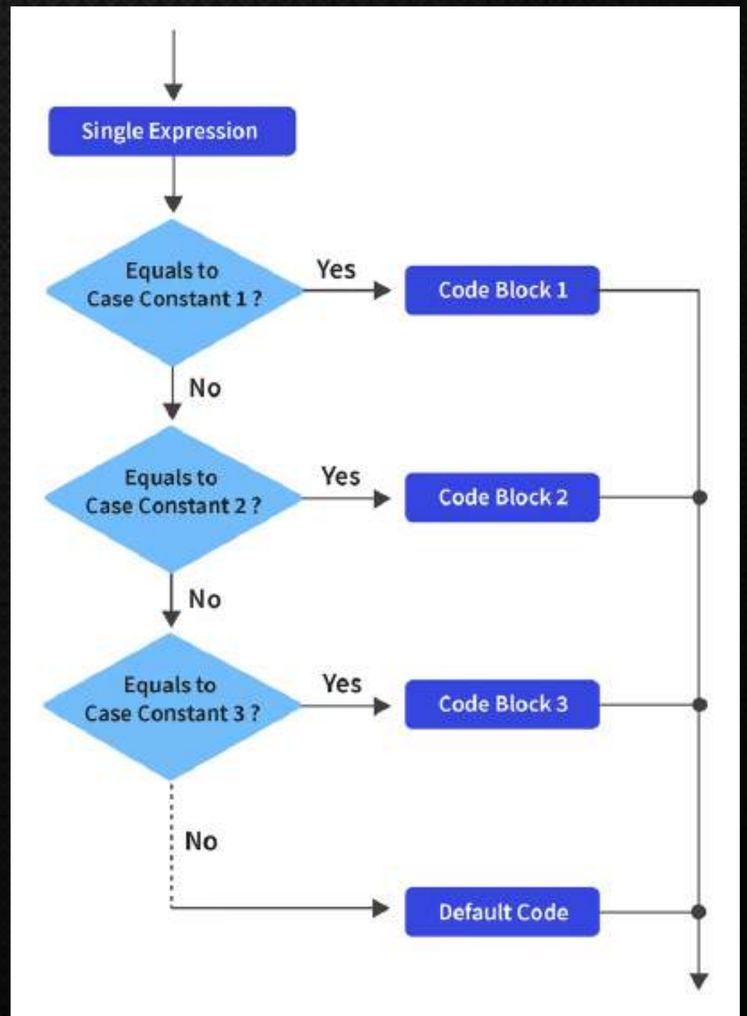
switch STATEMENT

Khan Omar

This statement allows us to execute different blocks of code based on different conditions. Rather than using if-elseif-if, we can use the switch statement to make our program.

```
<?php
$i = "2";
switch ($i) {
  case 0:
    echo "i equals 0";
    break;
  case 1:
    echo "i equals 1";
    break;
  case 2:
    echo "i equals 2";
    break;
  default:
    echo "i is not equal to 0,1, 2";
}
?>
```

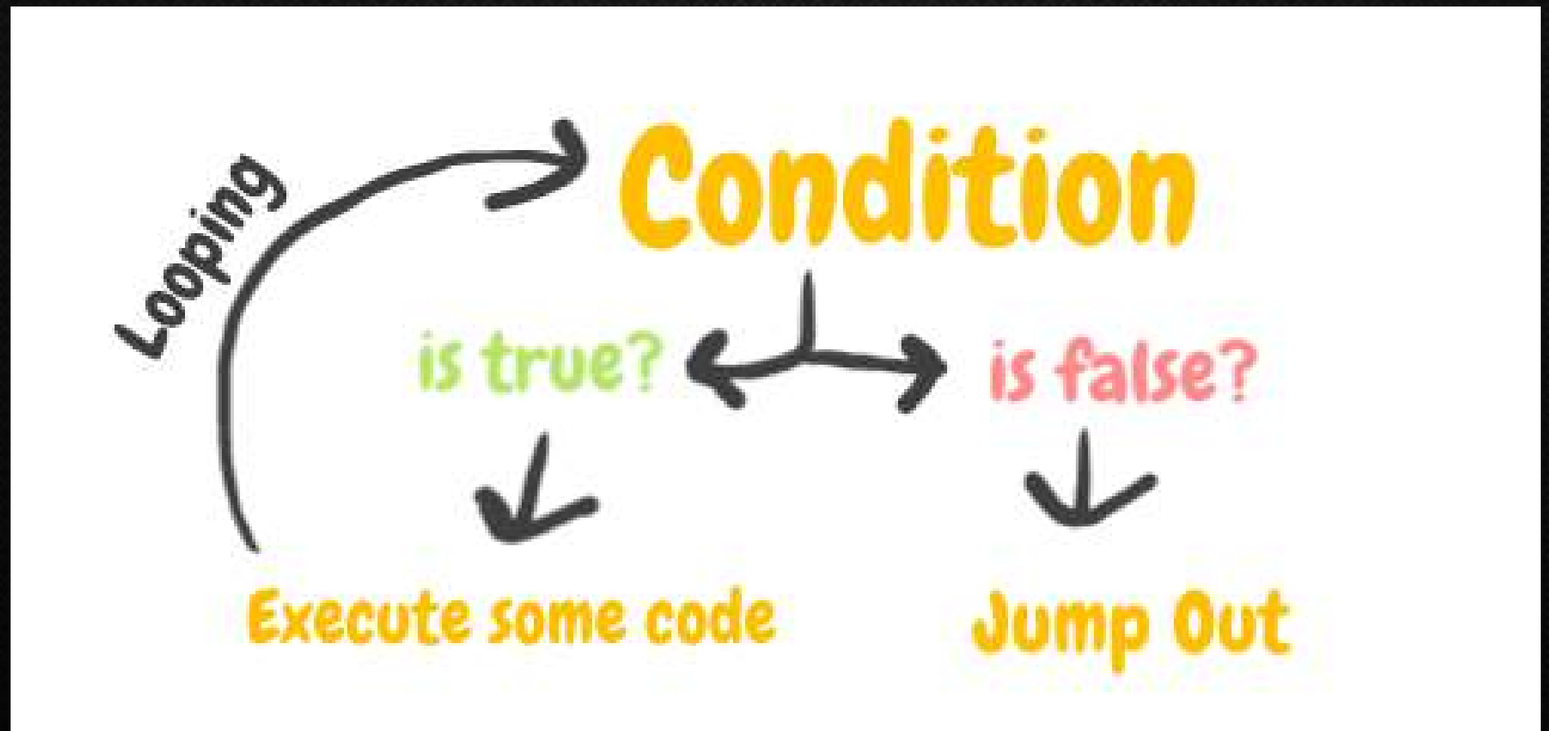
Output =>i equals 2



LOOPS

Khan Omar

In computer programming, a loop is a sequence of instruction s that is continually repeated until a certain condition is reached. Typically, a certain process is done, such as getting an item of data and changing it, and then some condition is checked such as whether a counter has reached a prescribed number.



LOOPS

Khan Omar

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

while LOOP

Khan Omar

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
  code to be executed;
}
```

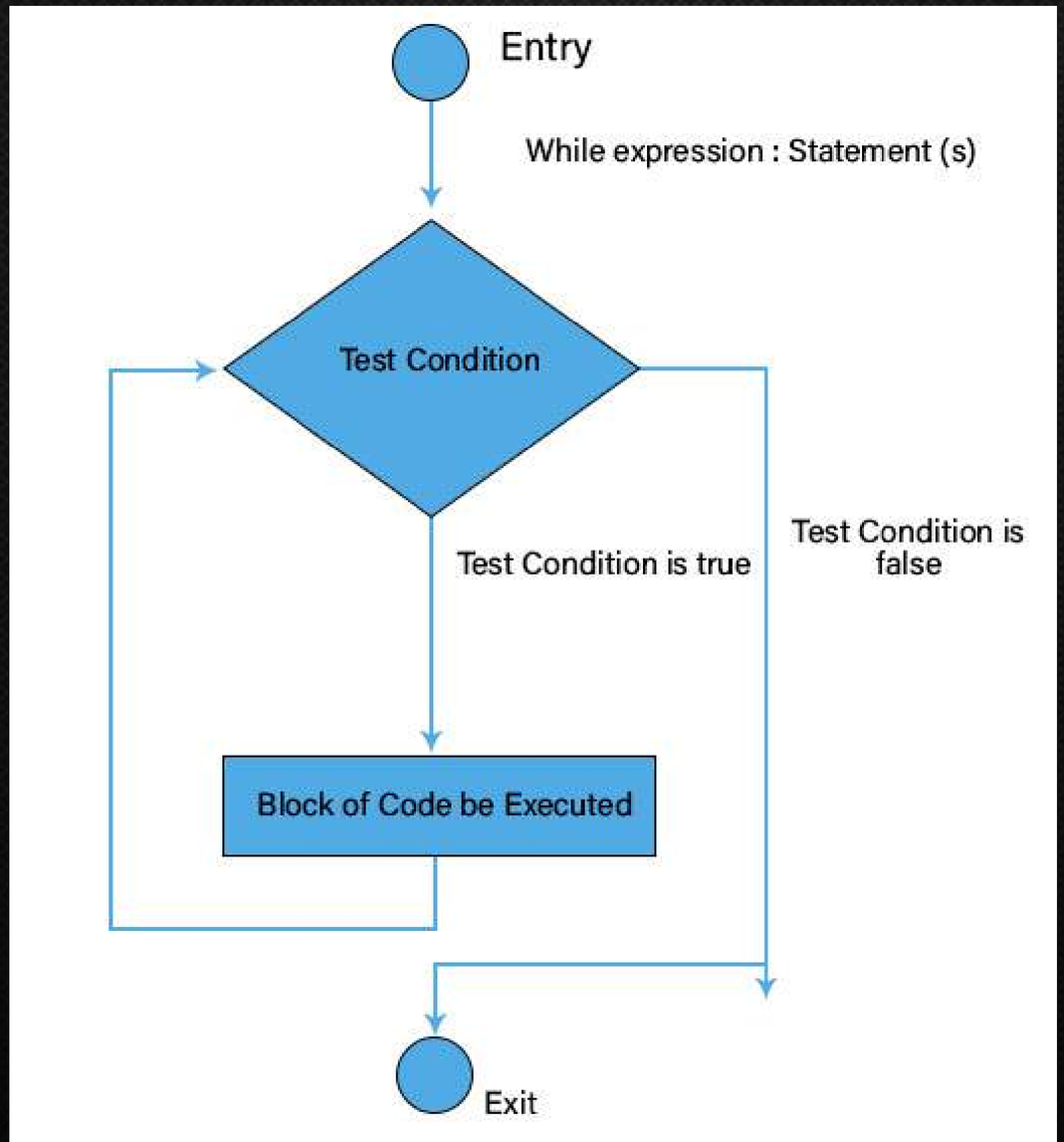
```
<?php
$x = 1;
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
} ?>
```

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

- $x = 1; - Initialize the loop counter ($x), and set the start value to 1
- $x <= 5 - Continue the loop as long as $x is less than or equal to 5
- $x++; - Increase the loop counter value by 1 for each iteration

while LOOP

Khan Omar



do-while LOOP

Khan Omar

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

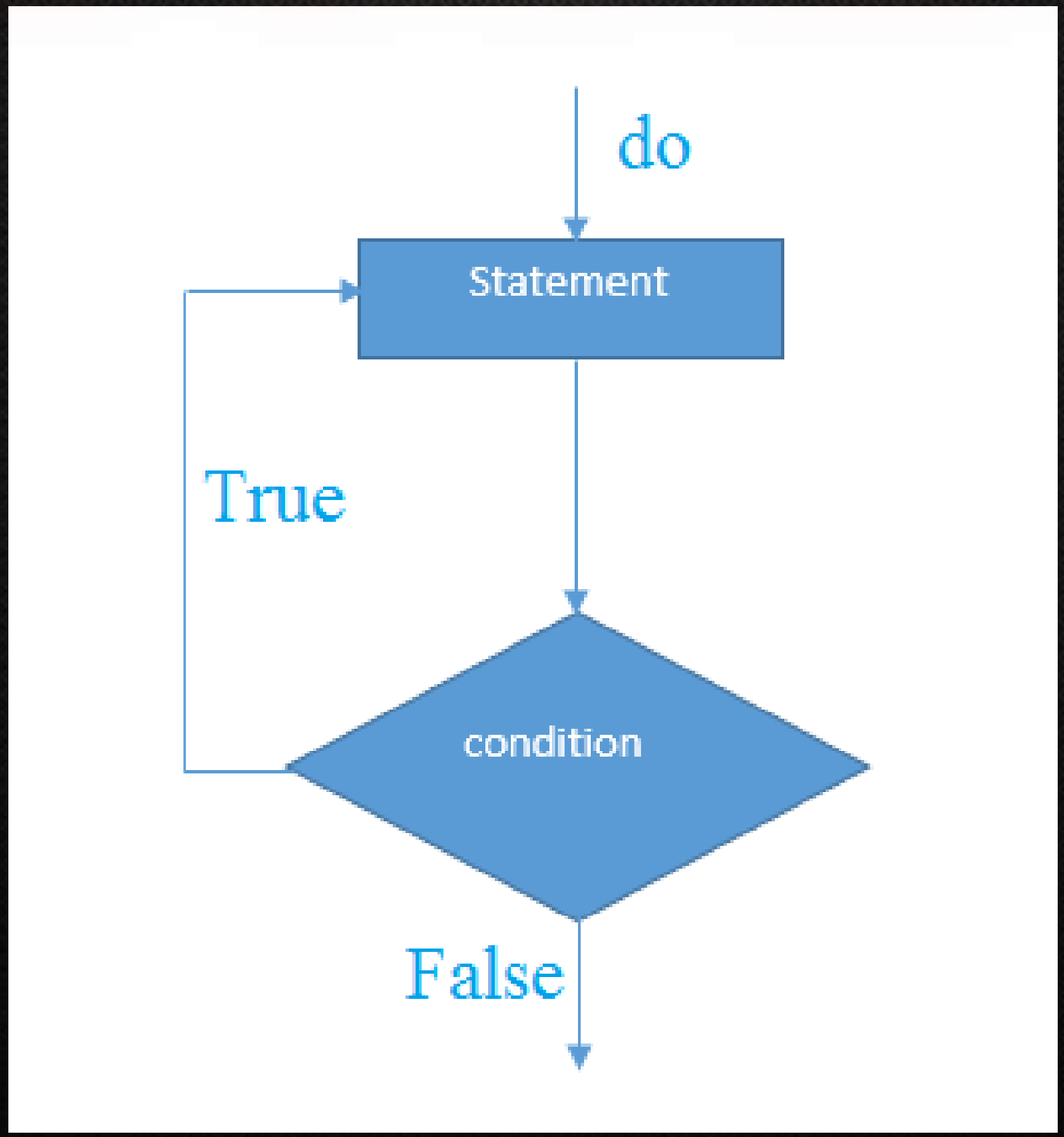
```
do {
 code to be executed;
} while (condition is true);
```

```
<?php
$x = 1;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5); ?>
```

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

do-while LOOP

Khan Omar



for LOOP

Khan Omar

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value

for LOOP

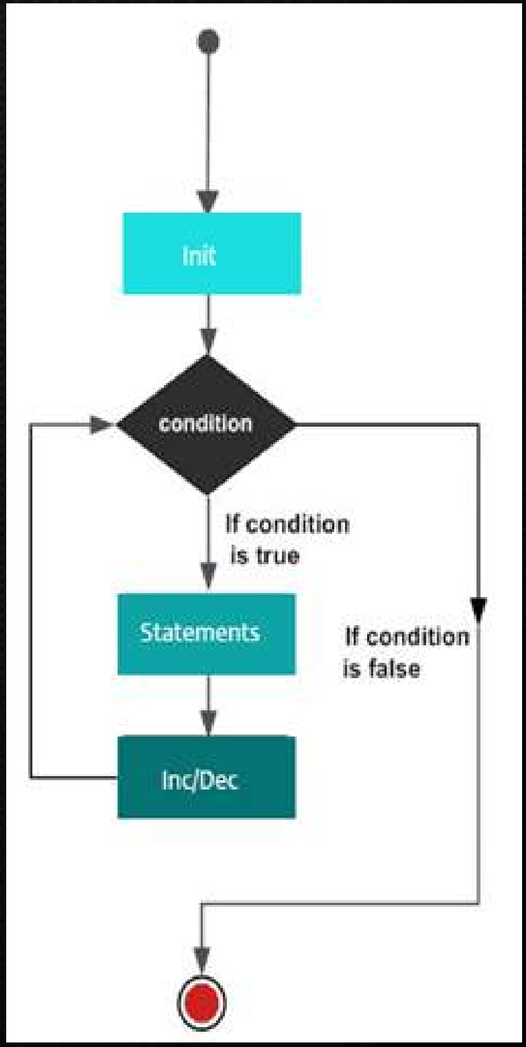
Khan Omar

Ex.

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10

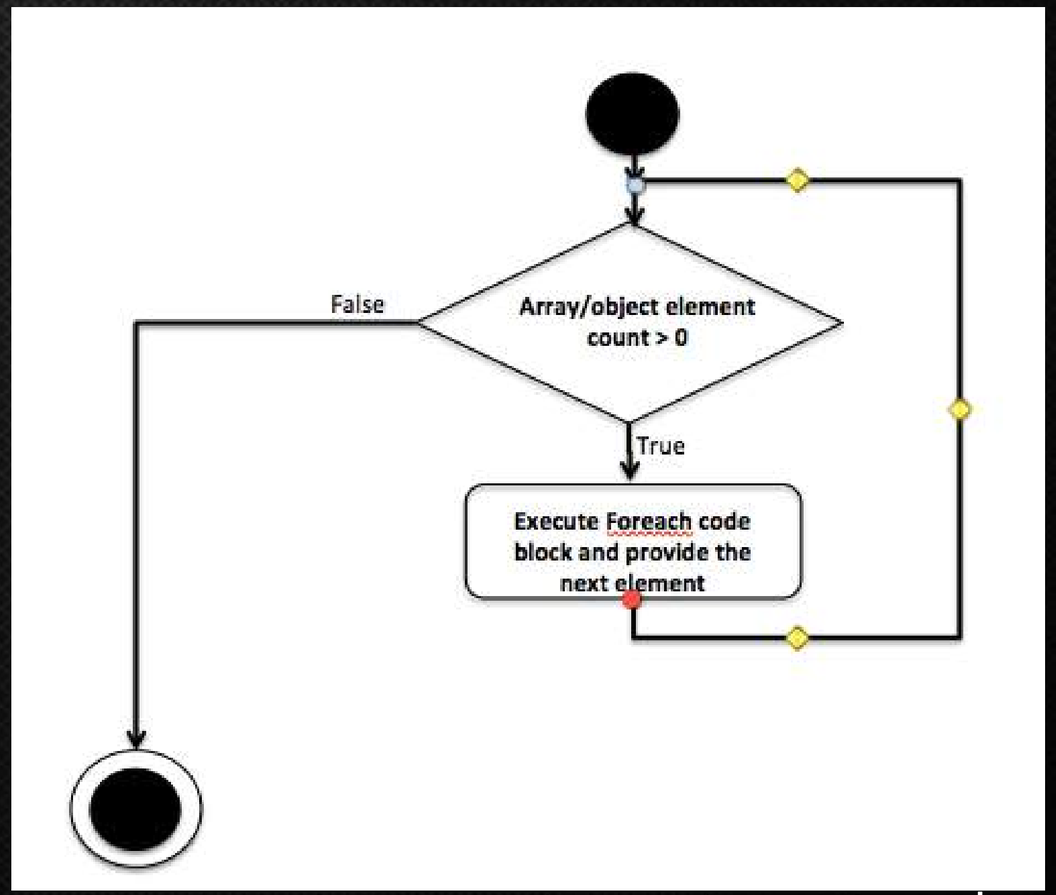
- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 100; - Continue the loop as long as $x is less than or equal to 100
- $x+=10 - Increase the loop counter value by 10 for each iteration



for-each LOOP

Khan Omar

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.



Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

for-each LOOP

Khan Omar

```
<?php
$colors = array("red", "green", "blue", "yellow");

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

Output =>

```
red
green
blue
yellow
```

BREAK & CONTINUE

Khan Omar

Break statement is used to "jump out" of a switch statement,out of a loop.

This example jumps out of the loop when x is equal to 1:

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 1) {
    break;
  }
  echo "The number is: $x <br>";
}?>
```

Output=> The number is: 0

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 1:

```
<?php
for ($x = 0; $x < 3; $x++) {
  if ($x == 1) {
    continue;
  }
  echo "The number is: $x <br>";
}?>
```

Output=> The number is: 0

The number is: 2

ARRAYS

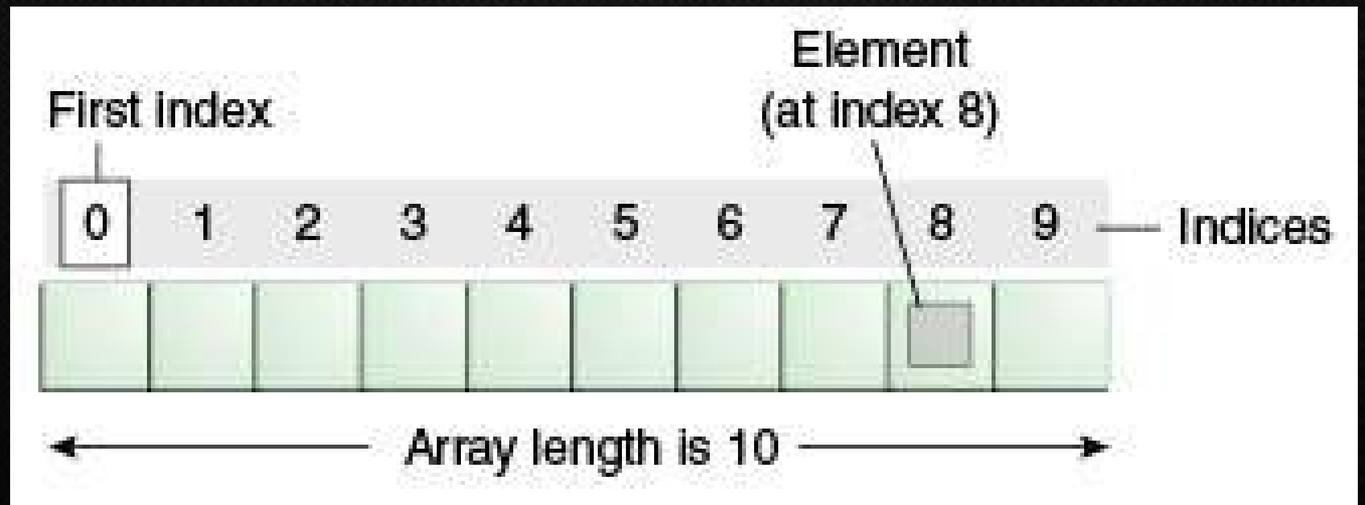
Khan Omar

An arrangement of objects, pictures, or numbers in rows and columns is called an array.

> An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

> An array is a special variable, which can store multiple values in one single variable.

> An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.



ARRAYS

Khan Omar

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Supra";
$cars2 = "BMW x7";
$cars3 = "Benz";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!
An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
Each element in the array has its own index so that it can be easily accessed.

ARRAYS

Khan Omar

Create an Array in PHP:
In PHP, the array() function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with a numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

INDEXED ARRAY

Khan Omar

There are two ways to create indexed arrays:
The index can be assigned automatically (index always starts at 0), like this:
$cars = array("Volvo", "BMW", "Toyota");

or the index can be assigned manually:
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

INDEXED ARRAY

Khan Omar

Ex.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Output => I like Volvo, BMW and Toyota.

Loop through array:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
  echo $cars[$x];
  echo "<br>";
}?>
```

Output=>
Volvo
BMW
Toyota

ASSOCIATIVE ARRAY

Khan Omar

Associative arrays are arrays that use named keys that you assign to them.
There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Associative Arrays

Key Name	Values
Naruto	Masashi Kishimoto
One piece	Eiichiro Oda
Bleach	Tite Kubo
Code Geass	Yoshihiro Togashi
Death Note	Yoshihiro Togashi

ASSOCIATIVE ARRAY

Khan Omar

Ex.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Output => Peter is 35 years old.

Loop through array:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}?>
```

Output =>
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

MULTIDIMENSIONAL ARRAY

Khan Omar

A multidimensional array is an array containing one or more arrays.
PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

MULTIDIMENSIONAL ARRAY

Khan Omar

Ex.

```
<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

Outut =>

Volvo: In stock: 22, sold: 18.

BMW: In stock: 15, sold: 13.

Saab: In stock: 5, sold: 2.

Land Rover: In stock: 17, sold: 15.

MULTIDIMENSIONAL ARRAY

Khan Omar

We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):

```
<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);

for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

MULTIDIMENSIONAL ARRAY

Khan Omar

Row number 0
- Volvo
- 22
- 18

Row number 1
- BMW
- 15
- 13

Row number 2
- Saab
- 5
- 2

Row number 3
- Land Rover
- 17
- 15

Output of previous exampe

FUNCTIONS

Khan Omar

> We will now explore how to create your own functions.
> To keep the script from being executed when the page loads, you can put it into a function.
> A function will be executed by a call to the function.
> You may call a function from anywhere within a page.
>A function will be executed by a call to the function.

Syntax:

```
function funcName(){
// code to be executed
}
```

> Give the function a name that reflects what the function does
> The function name can start with a letter or underscore (not a number)
> If you want to execute a function just call by it name.

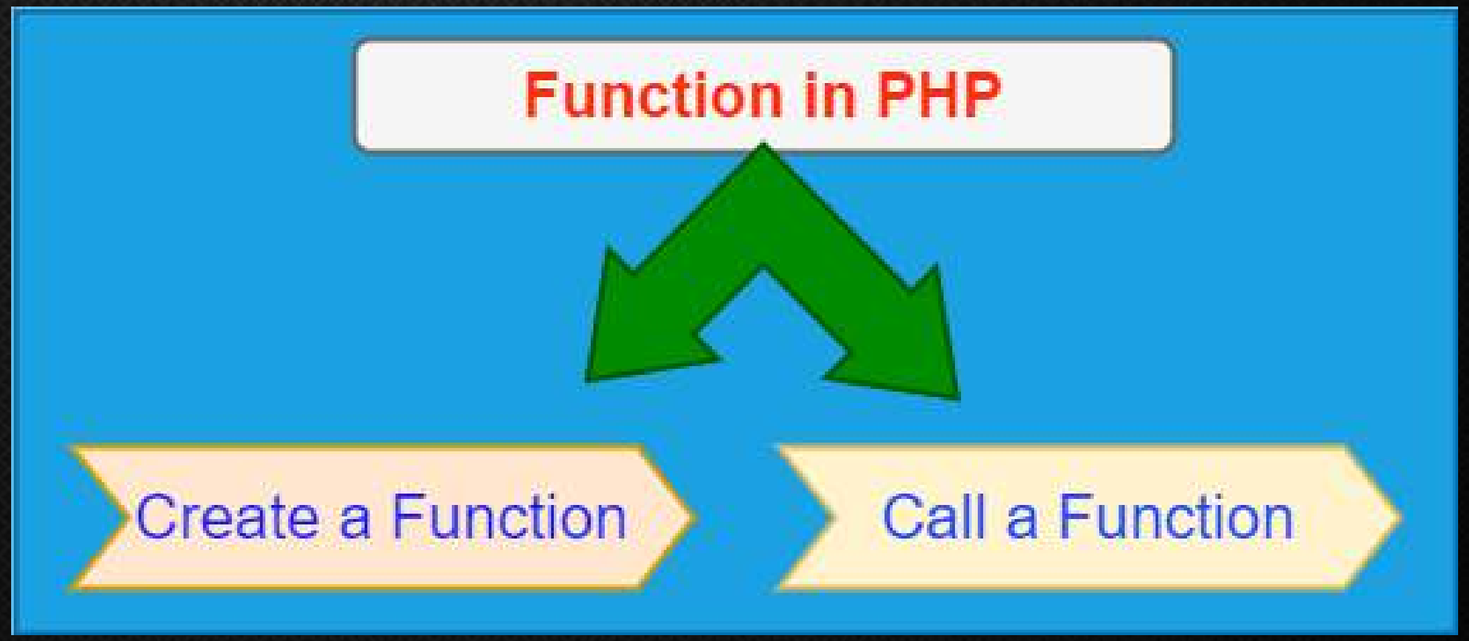
```
funcName();
```

FUNCTIONS

Khan Omar

```
<?php
function writeMsg() {
  echo "Hello Madi!";
}

writeMsg();
?>
```

Output:
Hello Madi!



FUNCTIONS

Khan Omar

PHP Function Arguments
Information can be passed to functions through arguments. An argument is just like a variable.
Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

FUNCTIONS

Khan Omar

```php
<?php
function familyName($fname) {
  echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.

FUNCTIONS

Khan Omar

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

```
<?php
function setHeight(int $minheight = 50) {
  echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>
```

```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

FUNCTIONS

Khan Omar

PHP Functions - Returning values

To let a function return a value, use the return statement:

```
<?php
function sum(int $x, int $y) {
  $z = $x + $y;
  return $z;
}

echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>
```

5 + 10 = 15

7 + 13 = 20

2 + 4 = 6

FUNCTIONS

Khan Omar

Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used:

```
<?php
function add_five(&$value) {
  $value += 5;
}
$num = 2;
add_five($num);
echo $num;?>
```

7

SUPER GLOBALS

Khan Omar

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

$GLOBALS

Khan Omar

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.

```
<?php
$x = 75;
$y = 25;
function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z; ?>
```

Output => 100

In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!

$_SERVER

Khan Omar

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

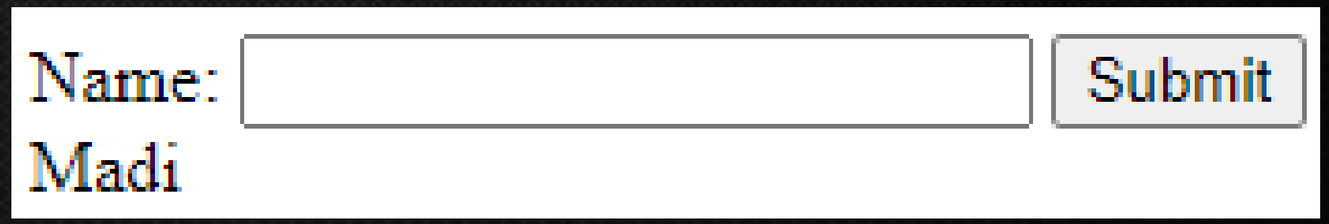
```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME']; ?>
```

$_REQUEST

Khan Omar

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = htmlspecialchars($_REQUEST['fname']);
   if (empty($name)) {
     echo "Name is empty";
   } else {
     echo $name; } }?>
```

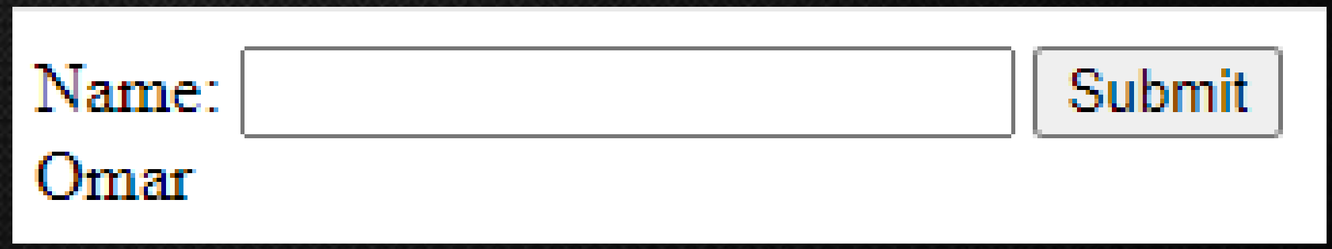


$_POST

Khan Omar

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = $_POST['fname'];
   if (empty($name)) {
     echo "Name is empty";
   } else { echo $name;}?>
```



$_GET

Khan Omar

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".
$_GET can also collect data sent in the URL.

```
<form action="/" method="get">
  <input type="text" name="name">
  <br>
  <input type="submit">
</form>
<?php
  echo $_GET["query"];
?>
```

The data would be appended to the the url
ex. https://myshopweb.netlify.app/?omar

GET & POST

Khan Omar

What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

HTTP Methods

- GET, POST, PUT, HEAD, DELETE, PATCH, OPTIONS, CONNECT, TRACE

The two most common HTTP methods are: GET and POST.

Get and Post methods are the HTTP request methods used inside the <form> tag to send form data to the server. HTTP protocol enables the communication between the client and the server where a browser can be the client, and an application running on a computer system that hosts your website can be the server.

Khan Omar

GET method

The **GET** method is used to submit the HTML form data. This data is collected by the predefined **$_GET variable** for processing.

The information sent from an HTML form using the GET method is visible to everyone in the browser's address bar, which means that all the variable names and their values will be displayed in the URL. Therefore, the get method is not secured to send sensitive information.

ex.

myshopweb.netlify.app/?user_id=1

GET

Khan Omar

This is gettest.html

```
<html> <body>
    <form action = "gettest.php" method = "GET">
       Username: <input type = "text" name = "username" /> <br>
       Blood Group: <input type = "text" name = "bloodgroup" /> <br>
       <input type = "submit" />
    </form>
  </body> </html>
```

The below code will display an HTML form containing two input fields and a submit button. In this HTML form, we used the method = "get" to submit the form data.

```
<html>
  <body>
    Welcome <?php echo $_GET["username"]; ?> </br>
    Your blood group is: <?php echo $_GET["bloodgroup"]; ?>
  </body>
</html>
```

if input username=Omar and bloodgroup=A+

When the user will click on Submit button after filling the form, the URL sent to the server could look something like this:

localhost/gettest.php?username=omar%bloodgroup=A+

GET

Khan Omar

Advantages of GET method (method = "get")

- You can bookmark the page with the specific query string because the data sent by the GET method is displayed in URL.
- GET requests can be cached.
- GET requests are always remained in the browser history.

Disadvantages of GET Method

- The GET method should not be used while sending any sensitive information.
- A limited amount of data can be sent using method = "get". This limit should not exceed 2048 characters.
- For security reasons, never use the GET method to send highly sensitive information like username and password, because it shows them in the URL.
- The GET method cannot be used to send binary data (such as images or word documents) to the server.

POST

Khan Omar

Similar to the GET method, the POST method is also used to submit the HTML form data. But the data submitted by this method is collected by the predefined superglobal variable $_POST instead of $_GET.
Unlike the GET method, it does not have a limit on the amount of information to be sent.
The information sent from an HTML form using the POST method is not visible to anyone.

ex.
myshopweb.netlify.app

POST

Khan Omar

This is posttest.php

```
<html> <body>
    <form action = "posttest.php" method = "post">
        Username: <input type = "text" name = "username" /> <br>
        Blood Group: <input type = "text" name = "bloodgroup" /> <br>
        <input type = "submit" />
    </form>
  </body> </html>
```

```
<html>
  <body>
    Welcome <?php echo $_POST["username"]; ?> </br>
    Your blood group is: <?php echo $_POST["bloodgroup"]; ?>
  </body>
</html>
```

if input username=Omar and bloodgroup=A+

When the user will click on Submit button after filling the form, the URL sent to the server could look something like this: localhost/posttest.php

POST

Khan Omar

Advantages of POST method (method = "post")

- The POST method is useful for sending any sensitive information because the information sent using the POST method is not visible to anyone.
- There is no limitation on size of data to be sent using the POST Method. You can send a large amount of information using this method.
- Binary and ASCII data can also be sent using the POST method.
- Data security depends on the HTTP protocol because the information sent using the POST method goes through the HTTP header. By using secure HTTP, you can ensure that your data is safe.

Disadvantages of POST Method

- POST requests do not cache.
- POST requests never remain in the browser history.
- It is not possible to bookmark the page because the variables are not displayed in URL.

REQUEST

Khan Omar

$_REQUEST variable

The $_REQUEST variable is a superglobal variable, which can hold the content of both $_GET and $_POST variable. In other words, the PHP $_REQUEST variable is used to collect the form data sent by either GET or POST methods. It can also collect the data for $_COOKIE variable because it is not a method-specific variable.

The example in the next slide shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

REQUEST

Khan Omar

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = htmlspecialchars($_REQUEST['fname']);
   if (empty($name)) {
     echo "Name is empty";
   }
else {
 echo $name;
 }
}
?>
</body>
</html>
```

REQUEST

Khan Omar

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   // collect value of input field
   $name = htmlspecialchars($_REQUEST['fname']);
   if (empty($name)) {
     echo "Name is empty";
   }
```

LETS KNOW ABOUT DATABASE

Khan Omar

- MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- MySQL software is Open Source.
- The MySQL Database Server is very fast, reliable, scalable, and easy to use.
- MySQL Server works in client/server or embedded systems.
- A large amount of contributed MySQL software is available.

The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way

LETS KNOW ABOUT DATABASE

Khan Omar

phpMyAdmin:
phpMyAdmin is a free and open source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services.

What is difference between MySQL and phpMyAdmin?
What is the difference between PHPMyAdmin and MySQL? MySQL is the database management system, or a database server. phpMyAdmin is the web application written primarily in PHP. It's used for managing MySQL database

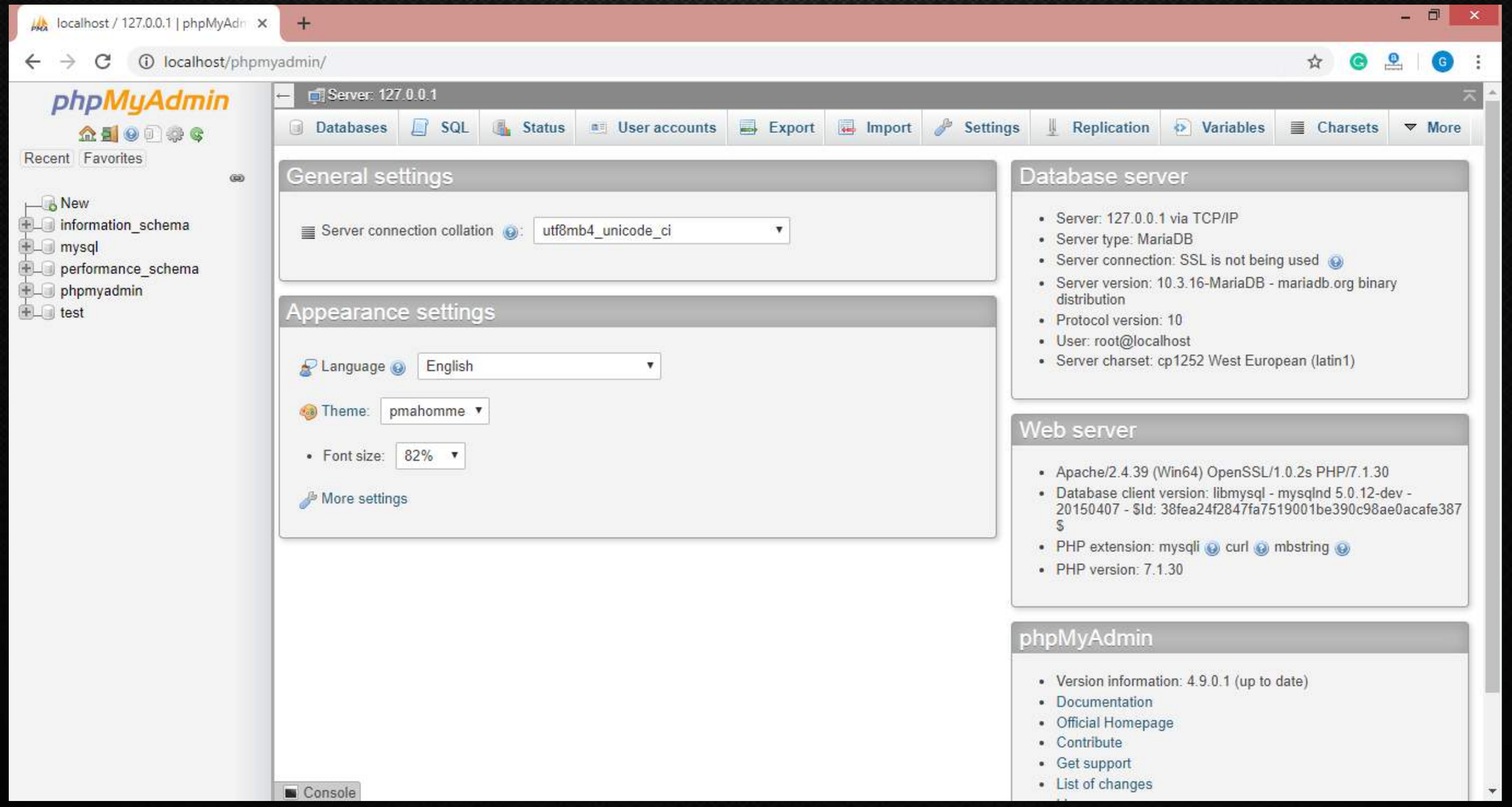
How to use:
after starting xammp apache and mysql got to chrome and type localhost/phpMyAdmin/

LETS KNOW ABOUT DATABASE

Khan Omar

After opening the phpMyadmin you will find this page:

1

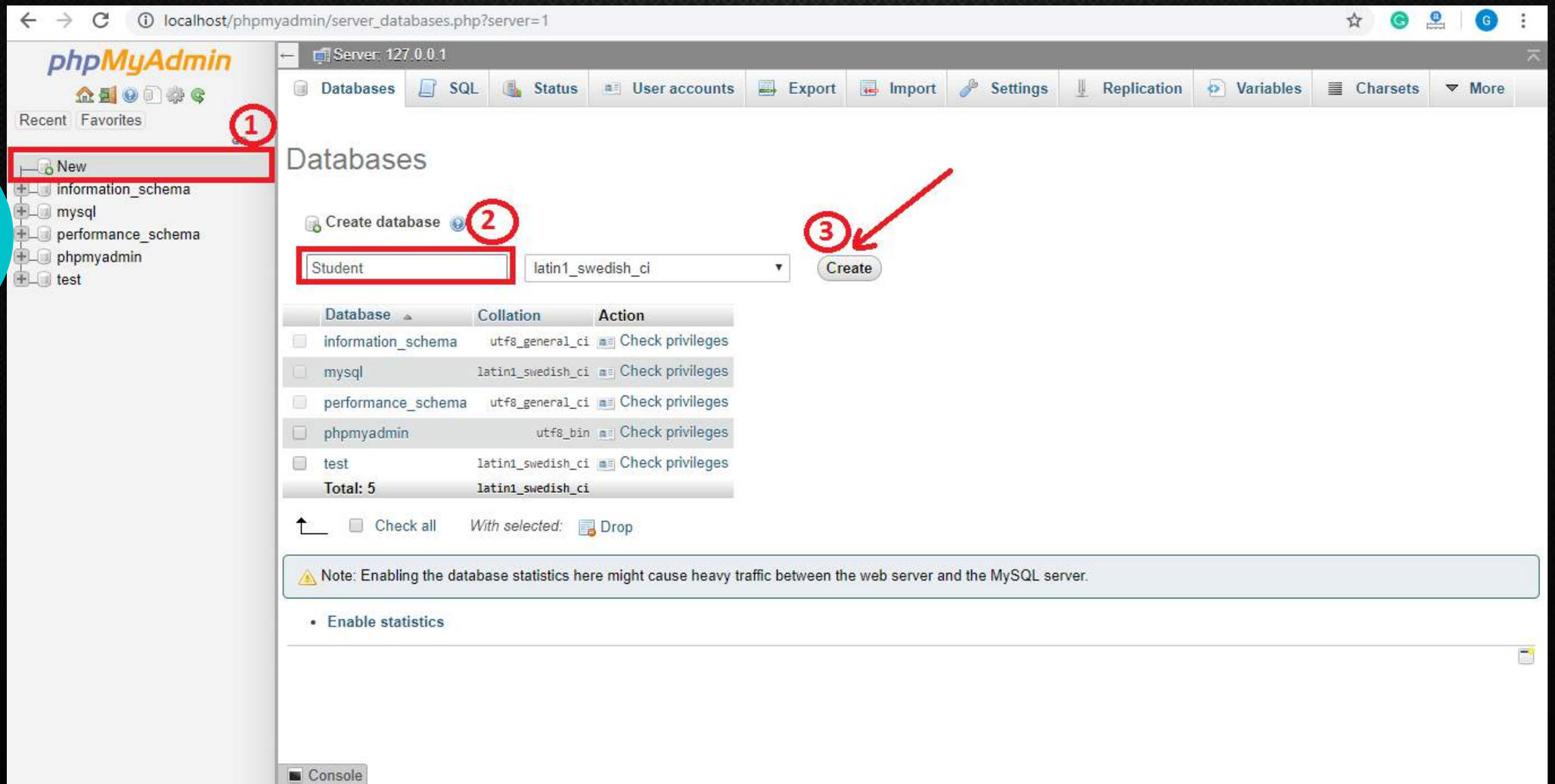


HOW TO WORK WITH PHPMYADMIN?

Khan Omar

Click on New (1) to create a database and enter the database name in Create database (2) field and then click on Create (3) button. We can create any number of databases.

2

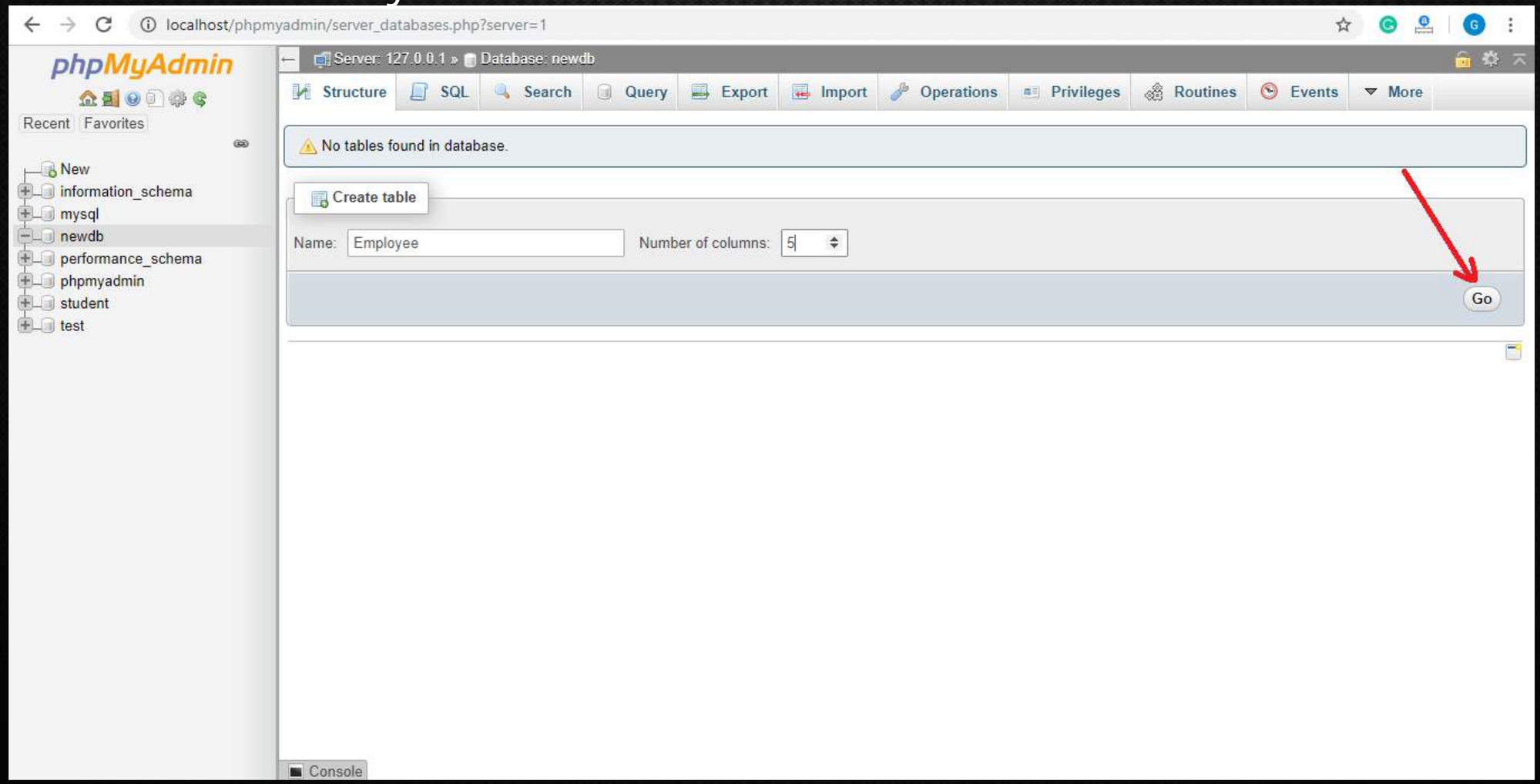


HOW TO WORK WITH PHPMYADMIN?

Khan Omar

Enter the table name, number of columns, and click on Go. A message will show that the table is created successfully.

3

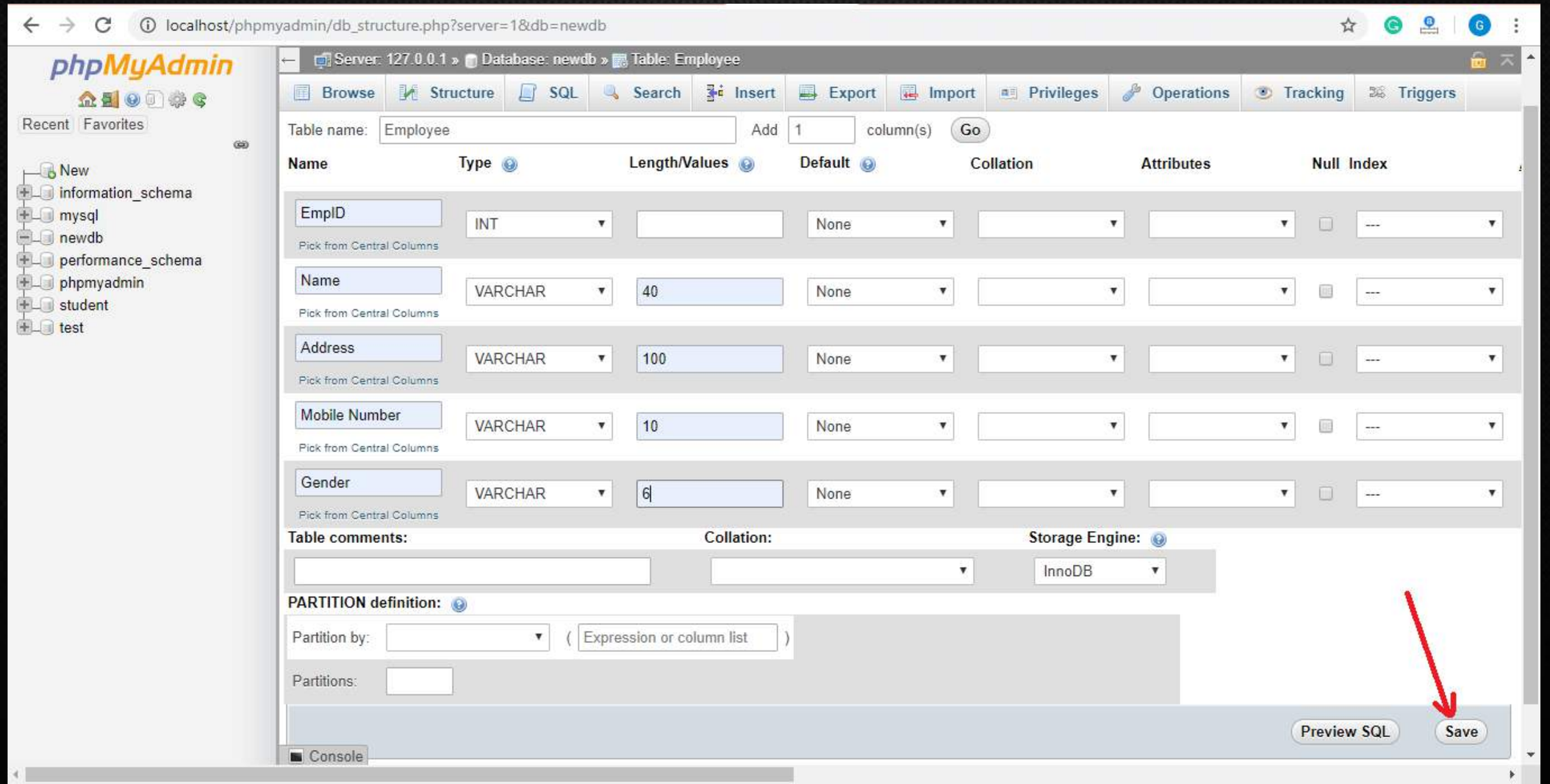


HOW TO WORK WITH PHPMYADMIN?

Khan Omar

Now enter the field name, type, their size, and any constraint here and save it.

4

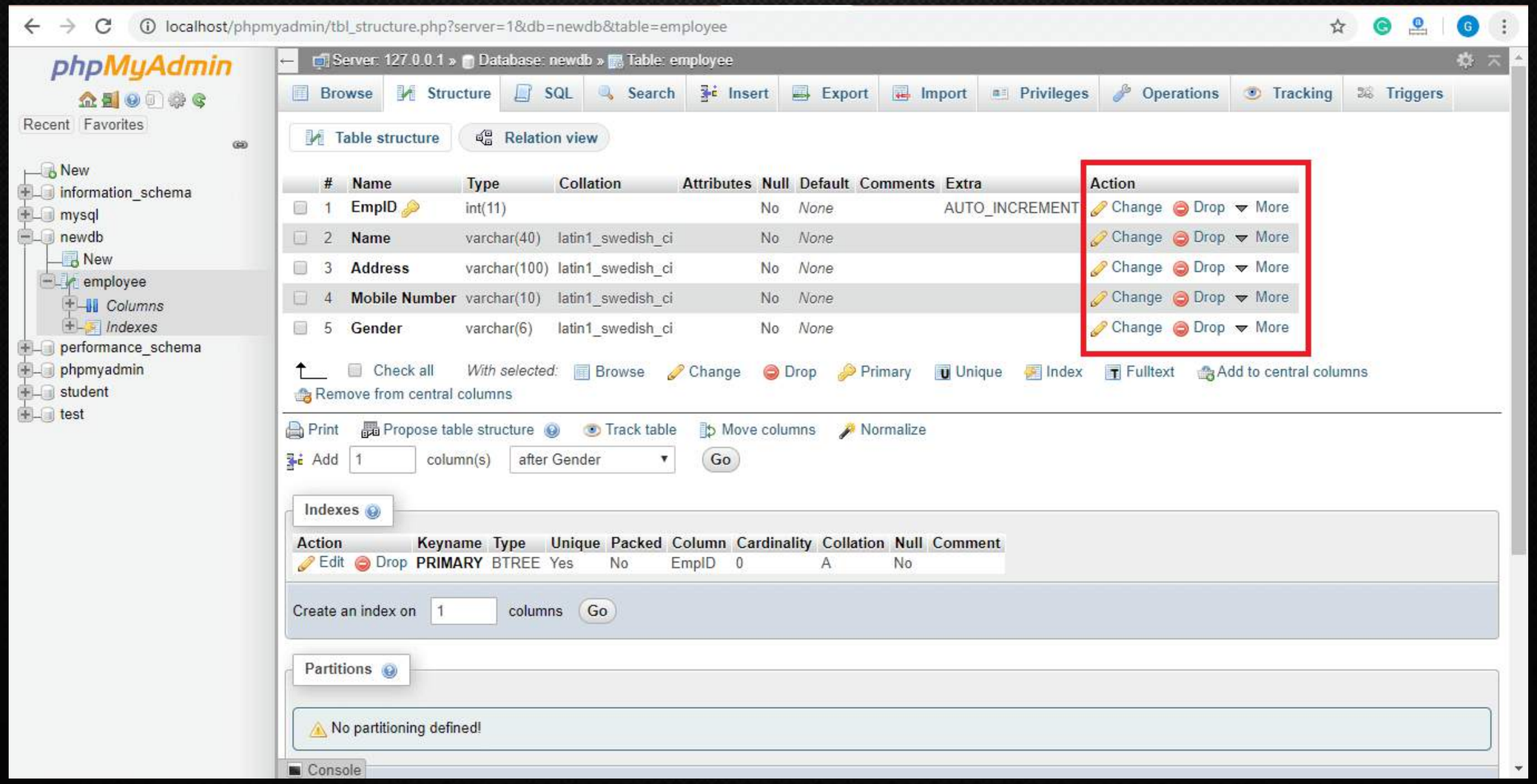


HOW TO WORK WITH PHPMYADMIN?

Khan Omar

The table is created successfully. We can make changes in the table from here.

5



DATABASE CREATION USING CODING WITH PHPMYADMIN

Khan Omar

```php
<?php
   $servername = "localhost";
   $username = "root";  //default user name is root
   $password = "";     //default password is blank
   $conn = mysqli_connect($servername, $username, $password);
   if(!$conn)
      die("Connection failed".mysqli_connect_error());
   else
      //echo "Successfully connected with database";
   $query = "CREATE DATABASE newDB";
   if (mysqli_query($conn, $query)) {
      echo "Database created successfully with the name newDB";
   } else {
      echo "Error creating database: " . mysqli_error($conn);
   }
   mysqli_close($conn);
?>
```

In phpMyAdmin, we create a database using a graphical user interface as well as by running queries.

INCLUDE AND REQUIRE

Khan Omar

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement.

Use require when the file is required by the application.
Use include when the file is not required and application should continue when file is not found.

INCLUDE AND REQUIRE

Khan Omar

1. create file 1 named (footer.php)

footer.php

```
<?php
$ph = 9152115523;
echo "Copyright@2023";
?>
```

2. Create a second file named (index.php)

```
<?php
echo `<h1> Hello Madi <h1> <br><br>`;
include(footer.php);
echo "My Phone Number is $a";
?>
```

Hello Madi

Copyright@2023
My Phone Number is 9152115523

This technique is useful so that if you want to change a single thing you have to change in one file not everyfile.

FUNCTIONS USED WHILE INERACTING WITH DB

Khan Omar

- mysqli_affected_rows => Get number of affected rows in previous mysql operation
- mysqli_client_encoding => Returns the name of the character set
- mysqli_close => Close mysql connection
- mysqli_connect => Open a connection to a mysql Server
- mysqli_create_db => Create a mysql database
- mysqli_data_seek => Move internal result pointer
- mysqli_db_name => Retrieves database name from the call to mysql_list_dbs
- mysqli_db_query => Selects a database and executes a query on it
- mysqli_drop_db => Drop (delete) a mysql database
- mysqli_errno => Returns the numerical value of the error message from previous mysql operation
- mysqli_error => Returns the text of the error message from previous mysql operation
- mysqli_escape_string => Escapes a string for use in a mysql_query
- mysqli_fetch_array => Fetch a result row as an associative array, a numeric array, or both

FUNCTIONS USED WHILE INERACTING WITH DB

Khan Omar

- mysqli_fetch_assoc => Fetch a result row as an associative array
- mysqli_fetch_field => Get column information from a result and return as an object
- mysqli_fetch_lengths => Get the length of each output in a result
- mysqli_fetch_object => Fetch a result row as an object
- mysqli_fetch_row => Get a result row as an enumerated array
- mysqli_field_flags => Get the flags associated with the specified field in a result
- mysql_field_len => Returns the length of the specified field
- mysqli_field_name => Get the name of the specified field in a result
- mysqli_field_seek => Set result pointer to a specified field offset
- mysqli_field_table => Get name of the table the specified field is in
- mysqli_field_type => Get the type of the specified field in a result
- mysqli_free_result => Free result memory
- mysqli_get_client_info => Get mysql client info
- mysqli_get_host_info => Get mysql host info
- mysqli_get_proto_info => Get mysql protocol info

FUNCTIONS USED WHILE INERACTING WITH DB

Khan Omar

- mysqli_get_server_info => Get mysql server info
- mysqli_info => Get information about the most recent query
- mysqli_insert_id => Get the ID generated in the last query
- mysqli_list_dbs => List databases available on a mysql server
- mysqli_list_fields => List mysql table fields
- mysqli_list_processes => List mysql processes
- mysqli_list_tables => List tables in a mysql database
- mysqli_num_fields => Get number of fields in result
- mysqli_num_rows => Get number of rows in result
- mysqli_pconnect => Open a persistent connection to a mysql server
- mysqli_ping => Ping a server connection or reconnect if there is no connection
- mysqli_query => Send a mysql query
- mysqli_real_escape_string => Escapes special characters in a string for use in an SQL statement

FUNCTIONS USED WHILE INERACTING WITH DB

Khan Omar

- mysqli_result => Get result data
- mysqli_select_db => Select a mysql database
- mysqli_set_charset => Sets the client character set
- mysqli_stat => Get current system status
- mysqli_tablename => Get table name of field
- mysqli_thread_id => Return the current thread ID
- mysqli_unbuffered_query => Send an SQL query to mysql without fetching and buffering the result rows

LETS ROCK!!! (CREATE CONNECTION)

Khan Omar

Create a file named connectDb.php

```
<?php
echo "let's connect with database";
// Connecting to Database
$serverName = "localhost";
$username = "root";
$password = "";
// Create a connection object
$conn = mysqli_connect($serverName, $username, $password);
echo "<br>";
// die if connection is not successfully
if (!$conn) {
  die("Sorry we failed to connect" . mysqli_connect_error());
} else {
  echo "connection was successful";
?>
```

Here we just created a connection without adding database

Output:

Connection was successfull

LETS ROCK!!! (CREATE DATABSE)

Khan Omar

2. create a database file createDb.php

```
<?php
include "connectDb.php";
// Create a database
$sql = "CREATE DATABASE madi";
$result = mysqli_query($conn, $sql);
// echo var_dump($result);
if ($result) {
 echo "db created successfully";
} else {
 echo "db not created" . mysqli_error($conn);
}
// you can directly create from phpMyAdmin
?>
```

You can also create from the GUI. of phpMyAdmin

Output:

db created successfully

LETS ROCK!!! (CREATE TABLE)

Khan Omar

```
<?php
// Connecting to Database
$serverName = "localhost";
$username = "root";
$password = "";
$database = "madi";
// this is for local host

// Create a connection object
$conn =
mysqli_connect($serverName,$username,
$password,$database);
// die if connection is not successfully
if(!$conn){
  die("Sorry we failed to
connect".mysqli_connect_error());
}

$sql = "CREATE TABLE Persons
(
PersonID int Primary Key,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);";
$result=mysqli_query($conn, $sql);
if ($result) {
  echo "Table created successfully";
} else {
echo "Table not
created".mysqli_error($conn);
}
?>
```

Output:

Table created successfully

LETS ROCK!!! (INSERT VALUE)

Khan Omar

```
<?php
$serverName = "localhost";
$username = "root";
$password = "";
$database = "madi";
// Create a connection object
$conn =
mysqli_connect($serverName,$username,
$password,$database);
echo "<br>";
// die if connection is not successfully
if(!$conn){
  die("Sorry we failed to
connect".mysqli_connect_error());}
```

```
$sql = "INSERT INTO Persons (PersonId,
LastName, FirstName, Address, City)
VALUES ('1','Khan',
'Madistic','Heart','Mumbai');
";
$result=mysqli_query($conn, $sql);

if ($result) {
 echo "Value inserted successfully";
} else {
echo "Value not inserted.
".mysqli_error($conn);
}
?>
```

←T→	PersonID	LastName	FirstName	Address	City
☐ Edit Copy Delete	1	Khan	Madistic	Heart	Mumbai

LETS ROCK!!! (DISPLAY VALUE)

Khan Omar

```
<?php
$serverName = "localhost";
$username = "root";
$password = "";
$database = "madi";

$conn = mysqli_connect($serverName,
$username, $password, $database);
$sql = "SELECT * FROM persons";
$res = mysqli_query($conn, $sql);
$numRow = mysqli_num_rows($res);
echo "Number of rows in Table:
".$numRow;
echo "<br>";
// display row
// if($numRow>0){
// $row = mysqli_fetch_assoc($res);
// echo var_dump($row);
// }
// the above example fetches only one row
while($row = mysqli_fetch_assoc($res)){
 // echo var_dump($row);
 echo $row['PersonID']."". Hello ".
$row['LastName'] ." ". $row['FirstName'] . "
.Your Address is '".$row['Address']."' and
your city is ".$row['City'];
 echo "<br>";
}
?>
```

Number of rows in Table: 1
1. Hello Khan Madistic .Your Address is 'Heart' and your city is Mumbai

LETS ROCK!!! (UPDATE VALUE)

Khan Omar

```
<?php
$serverName = "localhost";
$username = "root";
$password = "";
$database = "madi";

$conn = mysqli_connect($serverName,
$username, $password, $database);

// update
$sqlU = "UPDATE `persons` SET FirstName
= 'Madi' WHERE First Name='Madistic'";
$resU = mysqli_query($conn,$sqlU);
$affRow = mysqli_affected_rows($conn);
echo "Number of rows affected are: " .
$affRow;
echo "<br>";
if($resU){
 echo "Data updated sucessfully";
}
else{
 echo "Problem in updating
data".mysqli_error($conn);
}
?>
```

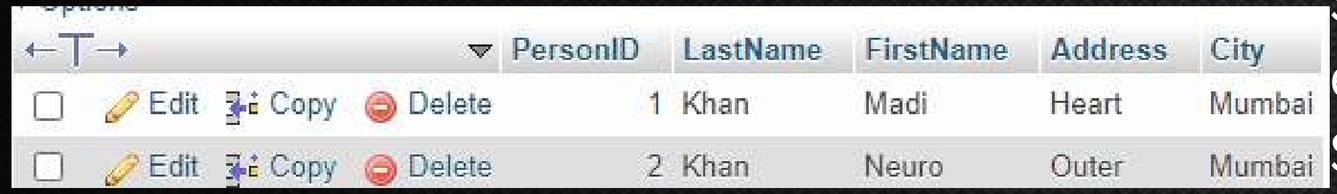
Number of rows affected are: 1
Data updated sucessfully

	PersonID	LastName	FirstName	Address	City
Edit Copy Delete	1	Khan	Madi	Heart	Mumbai

LETS ROCK!!! (UPDATE VALUE)

Khan Omar

Before Delete:



	PersonID	LastName	FirstName	Address	City
Edit Copy Delete	1	Khan	Madi	Heart	Mumbai
Edit Copy Delete	2	Khan	Neuro	Outer	Mumbai

```
<?php
$serverName = "localhost";
$username = "root";
$password = "";
$database = "madi";

$conn = mysqli_connect($serverName,
$username, $password, $database);
// delete only row limit specified
$sqlD = "DELETE FROM `person` WHERE
FirstName='Neuro' LIMIT 1";
$resD = mysqli_query($conn,$sqlD);
$affRow = mysqli_affected_rows($conn);
echo "Number of rows affected are: " .
$affRow;
echo "<br>";
if($resD){
 echo "Data deleted sucessfully";
}else{
 echo "Problem in deleting data.
".mysqli_error($conn);
}
?>
```

Number of rows affected are: 1
Data deleted sucessfully

After delete:

+ Options

	PersonID	LastName	FirstName	Address	City
Edit Copy Delete	1	Khan	Madi	Heart	Mumbai

Check all *With selected:* Edit Copy Delete Export

FILE HANDLING

Khan Omar

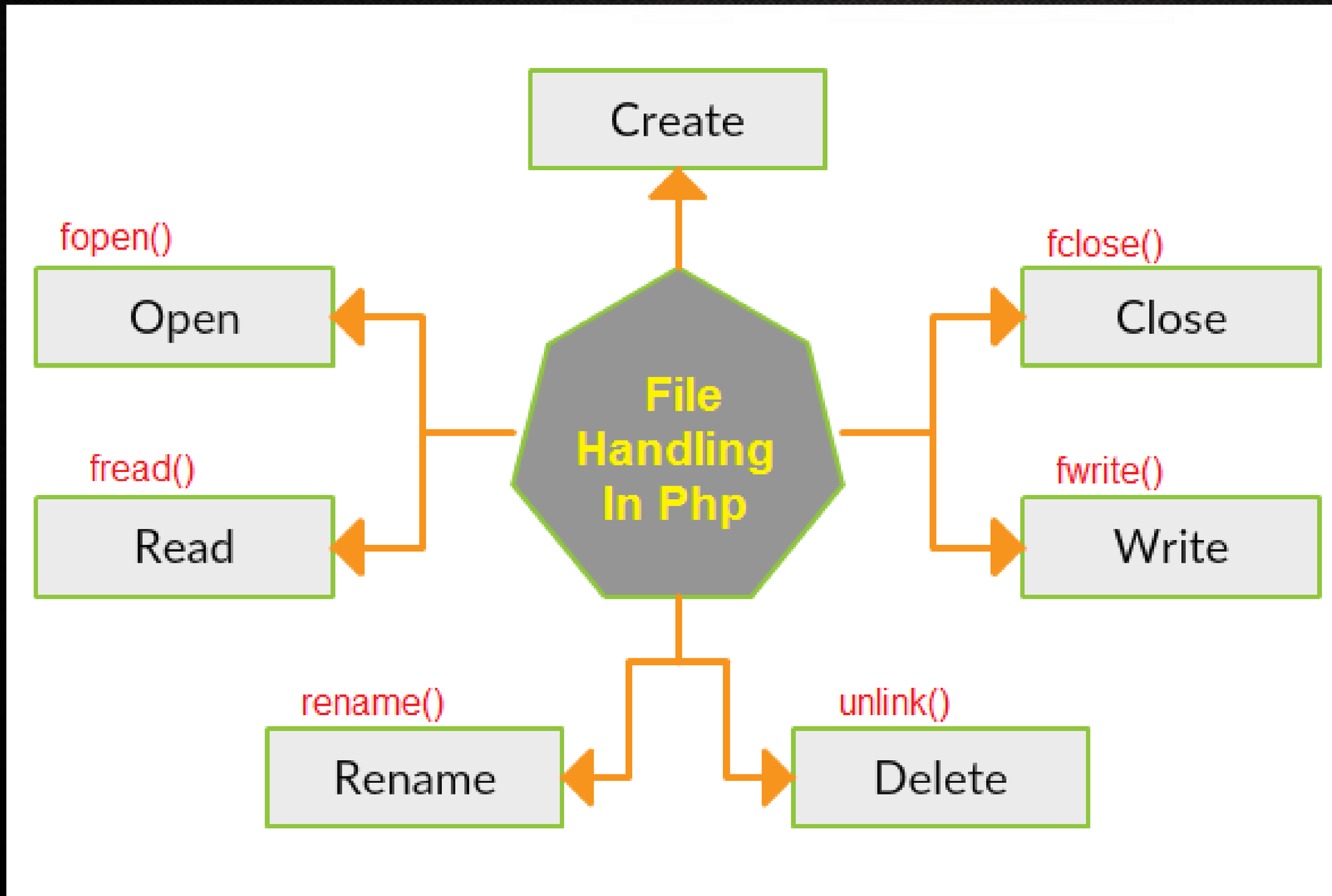
File handling is needed for any application. For some tasks to be done file needs to be processed. File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files.

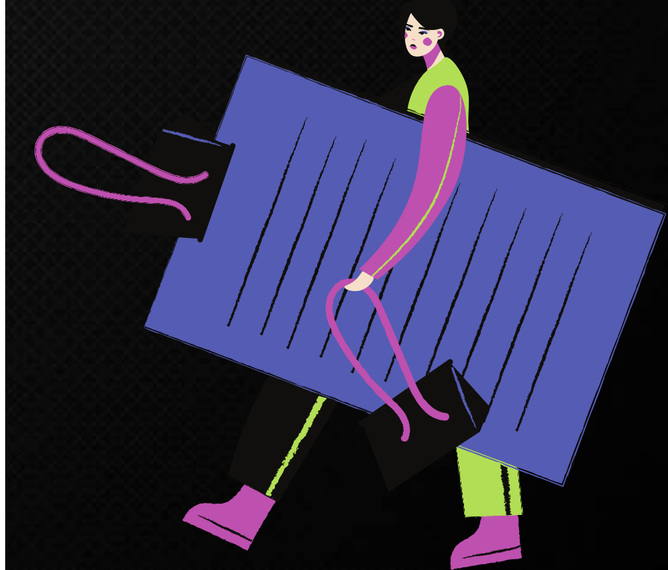
I have a txt file named as myFile.txt

```
<?php
// $t = readfile("myFile.txt");
// echo $t;
// The above code will show the content of file with the number of characters appended at the end.

// If you don't want number to be appended at the end just do readfile(path);
readfile("myFile.txt");
// you can also do this for html, png(will return in text format) and many more.
echo "<br>";
echo "Madi";
?>
```

FILE HANDLING

Khan Omar





FILE=> fopen()

Khan Omar

PHP fopen() function is used to open a file. First parameter of fopen() contains name of the file which is to be opened and second parameter tells about mode in which file needs to be opened.

```
<?php $file = fopen(“myFile.txt”,'w'); ?>
```

Files can be opened in any of the following modes :

- “w” – Opens a file for write only. If file not exist then new file is created and if file already exists then contents of file is erased.
- “r” – File is opened for read only.
- “a” – File is opened for write only. File pointer points to end of file. Existing data in file is preserved.
- “w+” – Opens file for read and write. If file not exist then new file is created and if file already exists then contents of file is erased.
- “r+” – File is opened for read/write.
- “a+” – File is opened for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.
- “x” – New file is created for write only.

FILE=> fread()

Khan Omar

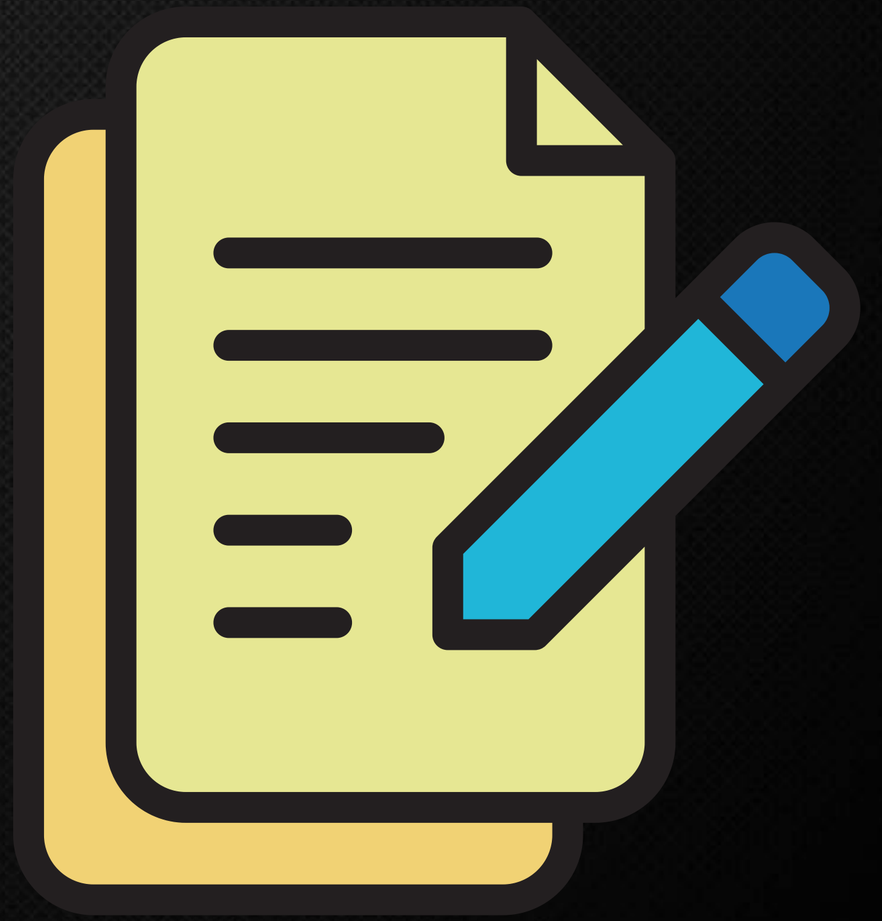
After file is opened using fopen() the contents of data are read using fread(). It takes two arguments. One is file pointer and another is file size in bytes,

```
fread($myfile,filesize("fileName"));
```

FILE => fwrite()

Khan Omar

New file can be created or text can be appended to an existing file using fwrite() function. Arguments for fwrite() function are file pointer and text that is to written to file. It can contain optional third argument where length of text to written is specified.



The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

```
fwrite($myfile, $ptr);
```

FILE=> fwrite()

Khan Omar

You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.

$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");

FILE=> fappend()

Khan Omar

You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.

```
$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");

// $fptr = fopen("myFile2.txt","w");
// // open file name if not created on fopen it will create in write mode
// // if you open a file in w/ write mode it will make the file empty
```

```
<?php

// Appending into file
$fptr = fopen("myFile2.txt","a");
fwrite($fptr,"The First Append. ");
fwrite($fptr,"The Second Append.");
fclose($fptr);
?>
```

Output:
The First Append. The second append

FILE=> fclose()

Khan Omar

The fclose() function is used to close an open file.

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
The fclose() requires the name of the file (or a variable that holds the filename) we want to close

FILE getc()

Khan Omar

The fgetc() function returns a single character from an open file.
Note: This function is slow and should not be used on large files. If you need to read one character at a time from a large file, use fgets() to read data one line at a time and then process the line one single character at a time with fgetc().

```
<?php
$file = fopen("test.txt","r");
echo fgetc($file);
fclose($file);
?>
```

Output: H

```
<?php
$file = fopen("test.txt","r");
while (! feof($file)) {
  echo fgetc($file);
  }
fclose($file);
?>
```

Output: Hello, this is a test file. There are three lines here. This is the last line.

FILE gets()

Khan Omar

The fgets() function returns a line from an open file.

```
<?php
$file = fopen("test.txt","r");
echo fgets($file);
fclose($file);
?>
```

Output:

Hello, this is a test file.

```
<?php
$file = fopen("test.txt","r");
while(! feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

Output:

Hello, this is a test file.
There are three lines here.
This is the last line.

COOKIE

Khan Omar

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP
A cookie is created with the setcookie() function.

Syntax
setcookie(name, value, expire, path, domain, secure, httponly);

Only the name parameter is required. All other parameters are optional.

Create/Retrieve a Cookie

Khan Omar

The following example creates a cookie named "user" with the value "Madi". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).
We then retrieve the value of the cookie "user" (using the global variable $_COOKIE).
We also use the isset() function to find out if the cookie is set:

```
<?php
$cookie_name = "user";
$cookie_value = "Madi";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name]; }?>
```

To modify a cookie, just set (again) the cookie using the setcookie() function

Output:
Cookie 'user' is set!
Value is: Madi

Delete a Cookie

Khan Omar

To delete a cookie, use the setcookie() function with an expiration date in the past

```
<!DOCTYPE html>
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

Output:
Cookie 'user' is deleted

SESSION

Khan Omar

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the users computer.

What is a PHP Session?
When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
So; Session variables hold information about one single user, and are available to all pages in one application.

start SESSION

Khan Omar

A session is started with the session_start() function.
Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

```
<?php
// Start the session
session_start();

// Set session variables
$_SESSION["favp"] = "Madi";
$_SESSION["favanimal"] = "Panda";
echo "Session variables are set.";
?>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Output:
Session variables is set.

print SESSION

Khan Omar

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").
Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).
Also notice that all session variable values are stored in the global $_SESSION variable:

```
<?php
session_start();

// Echo session variables that were set on previous page
echo "Favorite p is " . $_SESSION["favp"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
```

Output:
Favorite p is Madi.
Favorite animal is Panda.

destroy SESSION

Khan Omar

To remove all global session variables and destroy the session, use session_unset() and session_destroy()

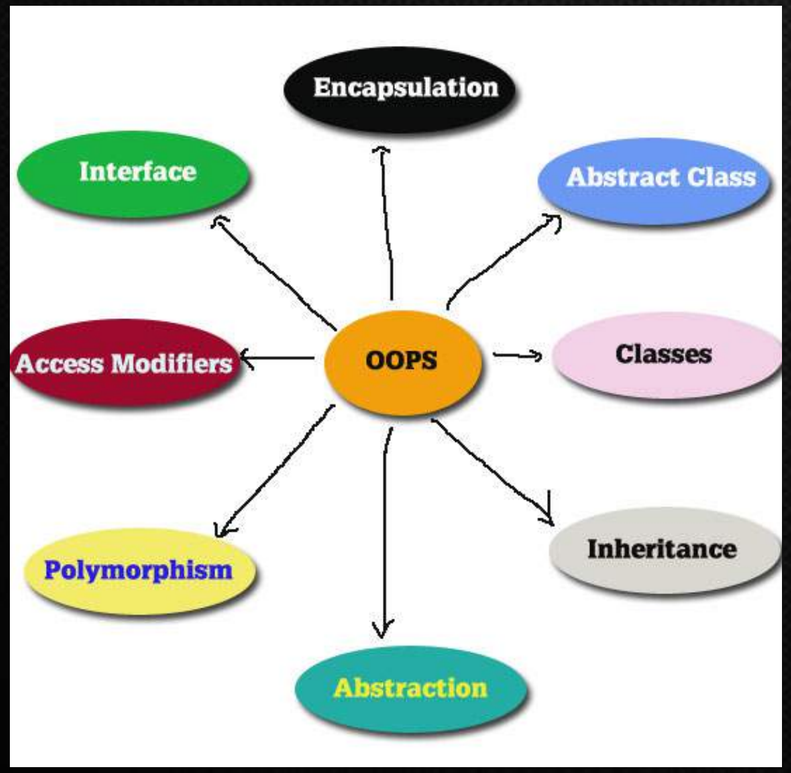
```
<?php
session_start();

// remove all session variables
session_unset();

// destroy the session
session_destroy();

echo "All session variables are now removed, and the session is destroyed."
?>
```

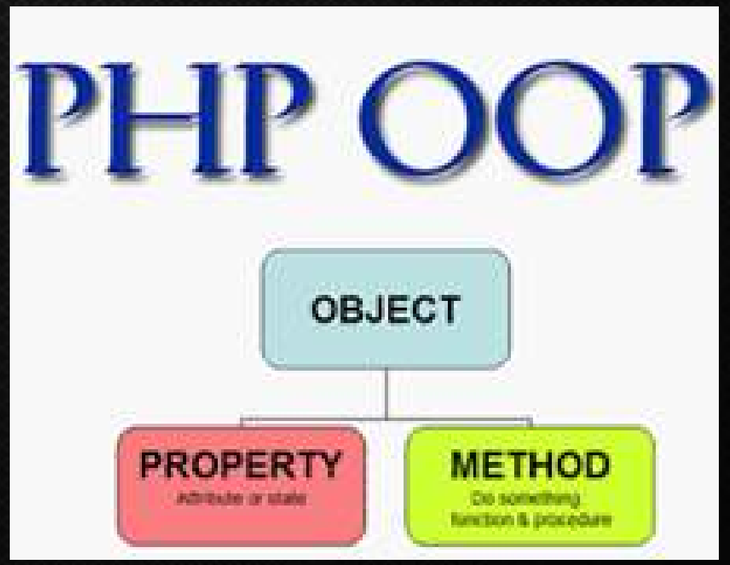
All session variables are now removed, and the session is destroyed.

LETS START WITH OOP

Khan Omar

Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data and code. The data is in the form of fields, and the code is in the form of procedures. A common feature of objects is that procedures are attached to them and can access and modify the object's data fields.



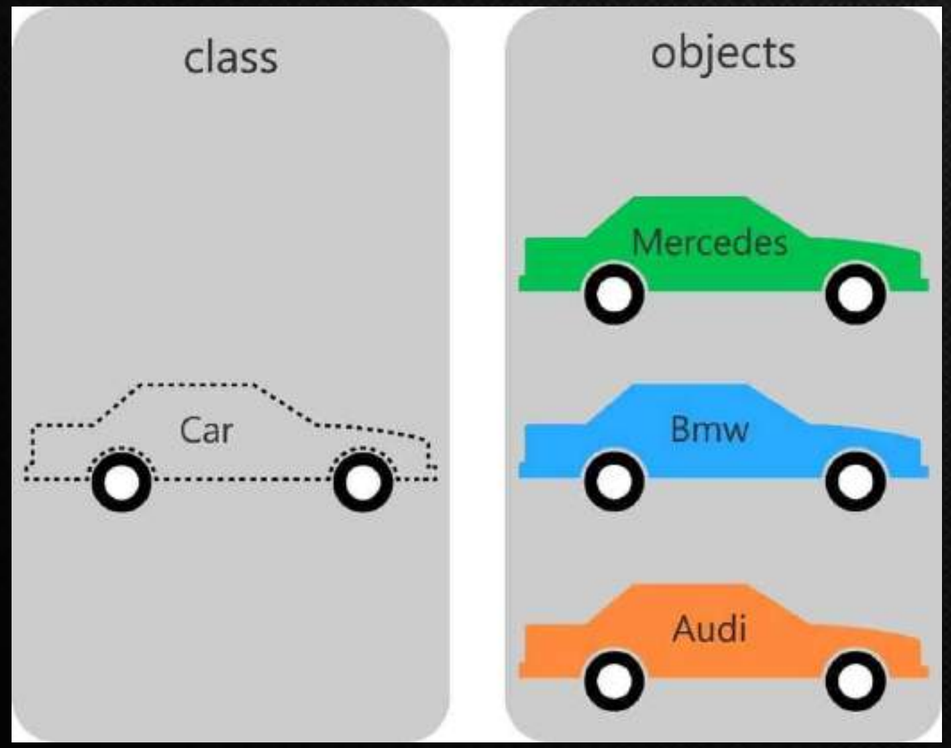


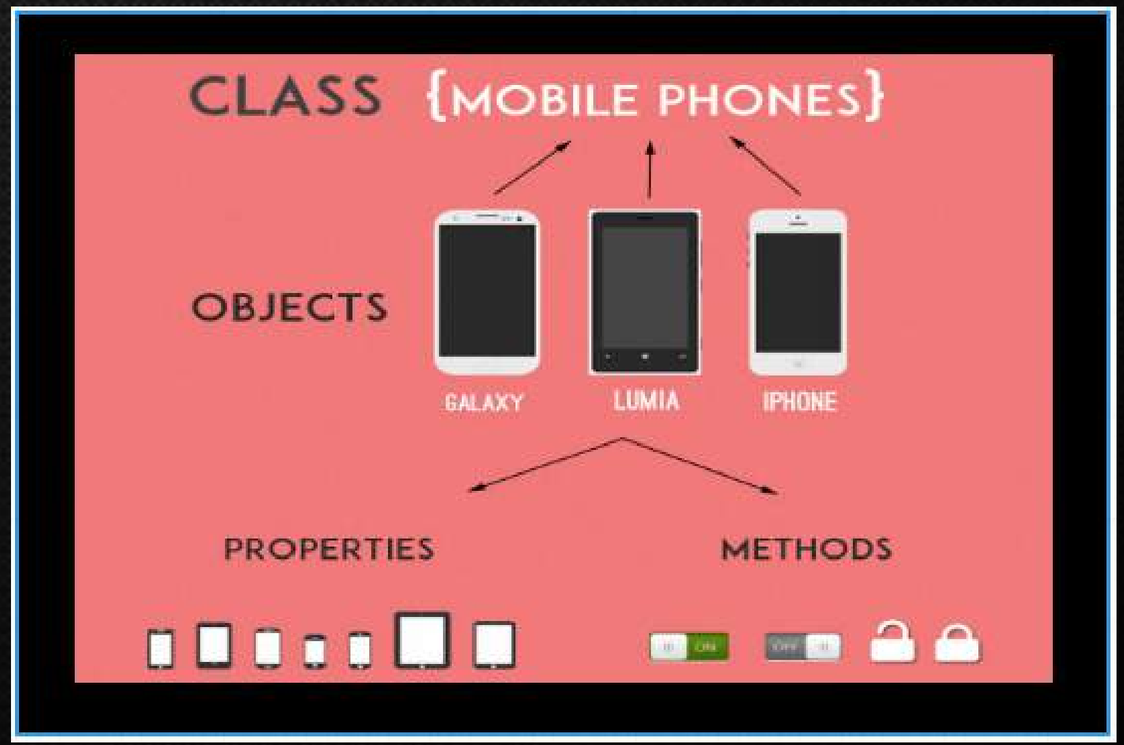
OOP CLASSES AND OBJECTS

Khan Omar

What is classes and objects?

=>Classes are user-defined data types that act as the blueprint for individual objects, attributes and methods. Objects are instances of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity.

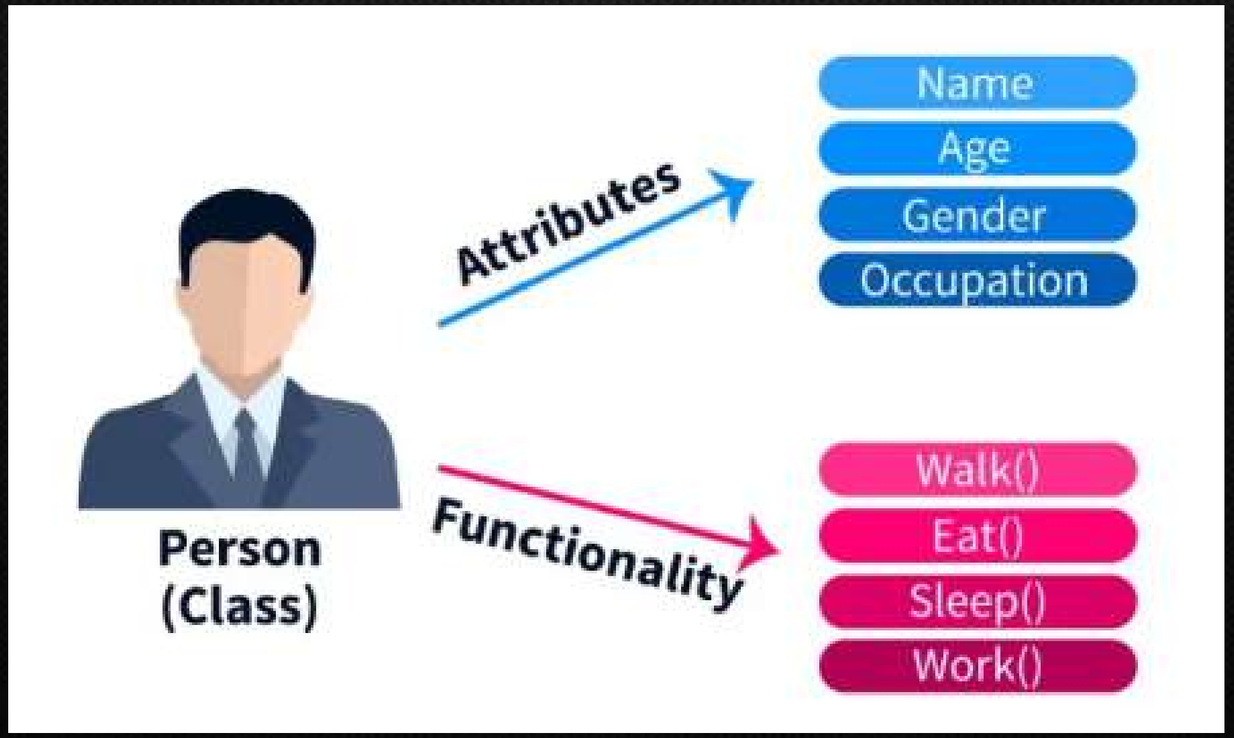




OOP CLASS

Khan Omar

Class Contain:
Class Name, properties(attributes), methods(functionalities).
This everything can be access with the help of object.



OOP

Khan Omar

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
  function set_color($color) {
  $this->color = $color;
  }
  function get_color() {
  return $this->color;
  }
}
$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " . $apple->get_color(); ?>
```

Output:
Name: Apple
Color: Red

The $this keyword refers to the current object, and is only available inside methods.

OOP CLASS

Khan Omar

Outside the class (by directly changing the property value):

```
<?php
class Fruit {
  public $name;
}
$apple = new Fruit();
$apple->name = "Apple";

echo $apple->name;
?>
```

Output: Apple

You can use the instanceof keyword to check if an object belongs to a specific class:

```
<?php
class Fruit {
  public $name;
  public $color;
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name; } }

$apple = new Fruit();
var_dump($apple
instanceof Fruit);
?>
```

Output: bool(true)

OOP CONSTRUCTOR

Khan Omar

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (__)!

```php
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function get_name() {
  return $this->name;
  }
}

$apple = new Fruit("Apple");
echo $apple->get_name();
?>
```

Output: Apple

OOP DESCTRUCTOR

Khan Omar

A destructor is called when the object is destructed or the script is stopped or exited.

If you create a __destruct() function, PHP will automatically call this function at the end of the script.

Notice that the destruct function starts with two underscores (__)!

```php
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function __destruct() {
  echo "The fruit is {$this-
>name}.";
  }
}

$apple = new Fruit("Apple");
?>
```

Output:
The fruit is Apple.

OOP ACCESS MODIFIERS

Khan Omar

Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

- public - the property or method can be accessed from everywhere. This is default
- protected - the property or method can be accessed within the class and by classes derived from that class
- private - the property or method can ONLY be accessed within the class

Specifiers	Same Class	Drived Class	Outside Class
Public	Yes	Yes	Yes
Protected	Yes	Yes	No
Private	Yes	No	No

OOP ACCESS MODIFIERS

Khan Omar

In the following example we have added three different access modifiers to three properties (name, color, and weight). Here, if you try to set the name property it will work fine (because the name property is public, and can be accessed from everywhere). However, if you try to set the color or weight property it will result in a fatal error (because the color and weight property are protected and private):

```php
<?php
class Fruit {
  public $name;
  protected $color;
  private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '300'; // ERROR
?>
```

OOP ACCESS MODIFIERS

Khan Omar

In the next example we have added access modifiers to two functions. Here, if you try to call the set_color() or the set_weight() function it will result in a fatal error (because the two functions are considered protected and private), even if all the properties are public:

```
<?php
class Fruit {
  public $name;
  public $color;
  public $weight;

  function set_name($n) {  // a public function (default)
    $this->name = $n;
  }
  protected function set_color($n) { // a protected function
    $this->color = $n; }
  private function set_weight($n) { // a private function
   $this->weight = $n;
  }
}

$mango = new Fruit();
$mango->set_name('Mango'); // OK
$mango->set_color('Yellow'); // ERROR
$mango->set_weight('300'); // ERROR
?>
```

OOP INHERITANCE

Khan Omar

Inheritance in OOP = When a class derives from another class.
The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.
An inherited class is defined by using the extends keyword.

OOP INHERITANCE

Khan Omar

```php
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the
color is {$this->color}.";
  }
}
```

```php
// Strawberry is inherited from
Fruit
class Strawberry extends Fruit {
  public function message() {
  echo "Am I a fruit or a berry? ";
  }
}
$strawberry = new
Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

OOP INHERITANCE(PROTECTED)

Khan Omar

```php
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color)
{
   $this->name = $name;
   $this->color = $color;
 }
  protected function intro() {
    echo "The fruit is {$this->name} and the
color is {$this->color}.";
 }
}

class Strawberry extends Fruit {
 public function message() {
 echo "Am I a fruit or a berry? ";
 }
}

// Try to call all three methods from outside
class
$strawberry = new Strawberry("Strawberry",
"red"); // OK. __construct() is public
$strawberry->message(); // OK. message() is
public
$strawberry->intro(); // ERROR. intro() is
protected
?>
```

OOP INHERITANCE(PROTECTED)

Khan Omar

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color)
{
    $this->name = $name;
   $this->color = $color;
  }
  protected function intro() {
    echo "The fruit is {$this->name} and the
color is {$this->color}.";
 }
}
class Strawberry extends Fruit {
 public function message() {
 echo "Am I a fruit or a berry? ";
 // Call protected method from within
derived class - OK
 $this -> intro();
 }
}

$strawberry = new Strawberry("Strawberry",
"red"); // OK. __construct() is public
$strawberry->message(); // OK. message() is
public and it calls intro() (which is protected)
from within the derived class
?>
```

OOP INHERITANCE(OVERIDING)

Khan Omar

```php
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color)
{
   $this->name = $name;
   $this->color = $color;
 }
  public function intro() {
    echo "The fruit is {$this->name} and the
color is {$this->color}.";
 }
}
```

```php
class Strawberry extends Fruit {
 public $weight;
 public function __construct($name, $color,
$weight) {
 $this->name = $name;
 $this->color = $color;
 $this->weight = $weight;
 }
 public function intro() {
 echo "The fruit is {$this->name}, the color is
{$this->color}, and the weight is {$this-
>weight} gram.";
 }
}
$strawberry = new Strawberry("Strawberry",
"red", 50); $strawberry->intro(); ?>
```

The fruit is Strawberry, the color is red, and the weight is 50 gram.

KHAN OMAR
MADI

Get In Touch:

+91 9152115523

https://github.com/KOMAR-7

khanomar0417@gmail.com