Course Content (PHP Basics)

Author

Aadarsha Paudel < a4d4rsha@gmail.com >

If you have any questions, found any problems or errors, please feel free to email on given address. If you wish to use different medium, search for username `a4d4rsha` on most of the social media.

I'll try to respond as quick as possible.

Changelog (v1.1)

- Added Changelog, Missing PHP Features (Session, Requests, Cookies, Require), MySQL and PDO (1.1)
- Initial Version (1.0)

Extensions for Visual Studio Code

For General Ease of Use

- Auto Close Tag
- Auto Rename Tag
- Highlight Matching Tag
- Bracket Pair Colorizer 2

PHP Extensions

> Search for PHP, These will come up on suggestion

- PHP Intelephense
- PHP Namespace Resolver
- PHP Docblocker

Running PHP Files

Run using XAMPP

Put your files on `c:/xampp/htdocs/folder`. Change `folder` to any name you wish (just do not put space in between).

Open the XAMPP Control Panel from Start Menu and Start `Apache` and `MySQL` server.

You can now head to `http://localhost/folder` on your browser.

Run using Console

Write some PHP code on some directory. To open terminal there, press `Shift` + `Right Click` and click on `Open Powershell window Here` option. It will open powershell. There you can type:

```
php filename.php
```

Change `filename.php` to your desired file.

Run Local Server Using PHP Development Server

> Note: This step requires PHP on your path. If you have installed Composer with `Add PHP To Path` option checked, it is already added.

PHP (on latest versions) come with local development server. You can follow same steps as above to open Powershell on the working directory.

```
php -S localhost:8000
```

Now head to browser and go to `http://localhost:8000` to run PHP Code.

PHP File

PHP Code starts with `<?php` tag and ends with `?>` tag.

If you code has only PHP, you do not need `?>` at the end.

Yes, semi-colon `;` **is required** in PHP for all statement like C Programming Language.

Placing PHP Code

You can follow two logic: Put PHP inside HTML or put HTML inside PHP.

Example of PHP Inside HTML:

```
<?php $a = 10; $b = 20; ?>
<h1>
    <?php echo $a + $b; ?>
</h1>
```

Example of HTML Inside PHP:

```php
<?php

$a = 10; $b = 20;

echo "<h1>";
echo $a + $b;
echo "</h1>";
```

Comments

PHP supports inline and multi line (block) comments:

```php
<?php

// this is an inline comment

# Hash signs are also comments (but dont use them if possible)

/*
This is a
multi line
or block
comment
*/
```

Any thing written as comment is not parsed by PHP.

Print / Echo

In PHP, there are multiple ways of outputting some data back.

```php
<?php

echo "Hello";
print "Hello";
echo ("Hello");
print("Hello");
printf("Hello");
print_r("Hello");
```

All statements in above example should print "Hello". In total, there will be 6 "Hello" output.

While you have the option to use any of them, I would advise you to use the first one.

```php
<?php

$a = 10; $b = 20;

echo "Hello";
echo $a + $b;
```

Variables

Variable name in PHP follow same principle as other languages. It must be a valid identifier i.e A-z, 0-9 and Underscore (_) are allowed.

To make a variable you need to add dollar sign (`$`) infront of the identifier. It is required!

```php
<?php

$a = 20;            // integer
$name = "Ram";      // string
$salary = 555.55;   // float
```

You can perform various operations on the variable including arithmetic operations.

Here is an example of addition:

```php
<?php

$a = 10;
$b = 20;

echo $a + $b;
```

Data Types

PHP Comes with various data-types. While PHP is dynamic typed language (i.e you can change from one data type to another for one variable), you can explictly cast data types.

```php
<?php

$a = (int) "10";
$b = (float) "25.25";
var_dump($a, $b);
```

Here above code transforms string to integer using type casting. Likewise you can cast to other data types such as:

- Integer (`int`)
- Floating Point (`float`)
- String (`string`)
- Boolean (`bool`)

Strings

In PHP, you can "join" other variable's value or other string into a string.

We use dot operator (`.`) to join strings.

```php
<?php

$name = "Ram";
echo "Hello " . $name;      // Output: Hello Ram
```

This is one of the method, another method is to put variable directly inside the string.

> Note: For this to work, you must be double quotes `"` instead of single.

```php
<?php

$name = "Ram";

echo "Hello $name";     // Output: Hello Ram
```

Boolean

In PHP boolean data types are written as `true/false` or `TRUE/FALSE`.

A important thing to remember in case of PHP. It follows, something or false logic i.e If the variable has value other than `0`, `FALSE` or `NULL`, it is false by default.

```php
<?php

$isAdmin = 0;   // false
$isAdmin = true;
$isAdmin = 12;  // it is true
$isAdmin = "";  // it is false
$isAdmin = false;
```

Array

In PHP, you can make dynamic arrays (i.e it does not have to be fixed size like C Programming Language.)

```php
<?php

$arr = array(1, 2, 3);
$arr2 = [1, 2, 3];
```

In the above example, we are making array using two different methods. While both works just fine, you can use the lower method of using `[ ]` to define array.

You can access array using their index. Array in PHP start with index of 0 (like any other language).

```php
<?php

$arr = ['Ram', 'Hari', 'Sita'];

echo $arr[1]; // this will print "Hari"
```

Array with Keys

While using index is okay for some operations, you may want to define your own index keys.

```php
<?php

$arr = [
    'first' => 'Ram',
    'second' => 'Hari',
    'third' => 'Sita',
];

echo $arr['second']; // this will print "Hari"
```

Mixing Data Types on Array

Unlike C, you can mix any data types into array.

```php
<?php

$arr = [1, 'Red', false];

echo $arr[2];   // it will print "Red"
```

In above example, you can see we have used data types: integer, string and boolean.

Multi Dimensional Arrays

In PHP, we can use multi-dimensional array i.e array have array inside.

```php
<?php

$arr = [
    [1, 2, 3],
    [4, 5, 6],
];

echo $arr[0][1];    // will print "2"
```

Flow Control (Conditions)

We can use following flow-control statements in PHP:

- If / Else
- If / Elseif / Else
- Switch

```php
<?php

$a = 10;
$b = 20;
$c = 30;

// If
if($a > 5) {
    echo "A bigger than 5 <br>";
}

// If Else
if($a > $b) {
    echo "A bigger than B <br>";
} else {
    echo "B bigger than A <br>";
}

// If Elseif Else
if($a > $b && $a > $c) {
    echo "A bigger than B and C <br>";
} elseif($b > $a && $b > $c) {
    echo "B bigger than A and C <br>";
} else {
    echo "C bigger than A and B <br>";
}

switch($a) {
    case 5:
        echo "A is 5";
        break;
    case 10:
        echo "A is 10";
        break;
    default:
        echo "A is not 5 or 10";
}
```

Omitting the Brackets

You can omitt the brackets `{}` if it has only one statement.

```
<?php
$a = 20;

if($a > 10)
    echo "A greater than 10";

if($a > 25)
    echo "A greater than 15";
    echo "Another statement";
```

> Note: Here `echo "Another statement";` is always run (even if that condition is false, which it is) since if we do not provide brackets, only the first statement after `if()` will run.

Loops

You can use `for`, `while`, `do-while` loops like any other language.

```
<?php

for($a = 0; $a < 10; $a++) {
    echo $a;
}
// output: 0 1 2 3 4 5 6 7 8 9 10

while($a < 10) {
    echo $a;
    $a++;
}
// output: 0 1 2 3 4 5 6 7 8 9 10
```

For dealing with arrays, PHP provides a handy loop called `foreach`. It loops for each items in array.

```php
<?php

$arr = ['red', 'green', 'blue'];
$arr2 = []; // empty array
foreach($arr as $item) {
    echo $item;
}
// Output: redgreenblue

foreach($arr2 as $item) {
    echo $item;
}

// No output since $arr2 is empty.
```

To use foreach, the variable passed must be an array. It can be empty, but it has to be array. PHP takes care of checking if it is empty or not.

Function

In PHP, Functions behave exactly same as on other language. It is a piece of code that you can call from anywhere once defined.

```php
<?php

function hello()
{
    echo "Hello";
}

hello();
```

You can also pass parameters to function:

```php
<?php

function hello($name)
{
    echo "Hello";
    echo $name;
}

hello("Hari");  // output: "Hari"
```

In PHP, you can specify optional parameters for function. If no data is passed, it will use that default value given.

```
<?php

function hello($name = "Hari")
{
    echo "Hello";
    echo $name;
}

hello();    // output: "Hari"
// Notice the parameter, we did not pass any name arguement.

hello("Ram");  // output: "Ram"
// when passed an arguement, it uses that instead of default one
```

Returning Values

```
<?php

function hello() {
    return "Hello";
}

echo hello();
```

Here in above example, we do not immediately print "hello" inside function, instead we return the value. The returned value can be set to variable or used anywhere else.

Oh, and since PHP is a dynamic typed language, it does not care what you return. You can return nothing, you can return a integer, string or whatever you wish to return.

System Functions

While we can define our own functions, PHP also provides us with various inbuilt functions.

Count

It counts the item in array.

Note: The count starts from 1

```
<?php

$arr = ['Red', 'Green', 'Blue']
count($arr);    // output: 3
```

Empty and Isset

Isset just check if the variable passed has been set or not.

Empty checks if the variable has been set or not AND the variable has any value.

```
<?php

$a = 10;
$b = "";

if(isset($a)) { echo "A is Set <br>"; }
if(isset($b)) { echo "B is Set <br> "; }
if(isset($c)) { echo "C is Set <br> "; } else { echo "C is not set <br>"; }

// Empty
if(isset($a))
    echo "A is Empty <br>";

if(isset($b)) { echo "B is Set <br> "; }
if(isset($c)) { echo "C is Set <br> "; } else { echo "C is not set <br>"; }
```

Unset

Unset is used to free variable from memory. While it is not used commonly, you can use it to avoid conflict with data on some cases.

```
<?php

$a = 10;

unset($a);

echo $a;    // will throw error about no variable $a
```

Var_Dump

It prints out the contents of given variable. It supports all data types (including arrays and objects)

```
<?php
$a = 10;
$arr = ['red', 'green', 'blue'];

var_dump($a);
var_dump($arr);
```

```
// or alternatively
var_dump($a, $arr);
```

You can pass any number of arguments to var_dump.

Die & Exit

Like it's name, it kills the PHP code execution when PHP runs into that statement.

`exit` is a nicer way than telling to `die`.

```php
<?php

echo "Hello";
die;
echo "World";
```

Here, `world` will not print as we told PHP to stop execution after `Hello` print.

String Functions

Explode

It helps to put items in string to array if there is any separator.

```php
<?php
$color = "red,green,blue";
$arr = explode(",", $color);
var_dump($arr); // we get 3 items: 0->red, 1->green, 2->blue as array
```

NL2BR (New Line to Break)

It prints `<br>` when it finds new line.

```php
<?php

$message = "hello

world";

echo nl2br($message); // output: hello<br/><br/>world
```

And no, just becase it has 2 in its name, it will not print 2 `<br>`. You can try putting multiple new line, it does its job correctly.

String Transform: Upper and Lower

```php
<?php

$str = "hElLo";
echo strtoupper($str);  // output: HELLO
echo strtolower($str);  // output: hello
```

String Length

```php
<?php

$str = "hello world";
echo strlen($str);  // output: 11
```

> Note: Like `count()` it counts from 1 and not 0

Trim

It removes whitespace (spaces, newline) from start and end of string.

```php
<?php

$str = "   hello   ";
echo trim($str);    // output "hello" with no space around
// (view page source for it to take effect)
```

String Replace

This helps you to replace some characters in string with another one.

```php
<?php

$str = "hello ram, hello hari, hello sita";
$str_rep = str_replace("hello", "hi", $str);
echo $str_rep;
```

This above code will replace all instance of "hello" on the `$str` variable with "hi".

By default, PHP will replace all instance not just first.

Math Functions

Random Number Generation

```
<?php

echo rand(1, 20);   // output: random number between 1 to 20
echo mt_rand(1, 20);    // same but more random-ish
echo uniqid();  // alpha-numeric random string
```

Date Time

PHP uses Unix Time. TD;LR: it is basically number of seconds after Jan 1, 1970.

```
<?php

echo time();    // current time
echo time() + 5;    // since it is an integer, you can add seconds to it
// prints 5 seconds after

echo time() + (60 * 60 * 5) + (45 * 60);  // prints Nepali time (if it is on UTC clock)
// adds 5 hours 45 minutes
```

While `time()` function deals with computer time, we can make human readable time format using `date()` function.

It accepts 1 required and 1 optional parameter:

```
date($format, $time = time());
```

For help with formatting letters, go to https://www.php.net/manual/en/datetime.format.php to learn more.

Example: print like: `2020-Jan-1 10:25:32 AM`

```
<?php

echo date("Y-M-d H:i:s A");
```

Hashing

Normally, in PHP md5 and password_hash() are used.

Note: MD5 produces same has for given string, always. It is prune to brute-force attack. I would advise you not to use it for passwords.

```php
<?php

$str = "Hello";

echo md5($str);
echo "<br>";
echo md5($str); // notice, it prints same hash everytime
echo "<br>";

echo password_hash($str, PASSWORD_BCRYPT);
echo "<br>";
echo password_hash($str, PASSWORD_BCRYPT);
```

Compared to `md5` hasing, `password_hash()` is much secure.

Verify MD5 Hash

```php
<?php

$hash = "5d41402abc4b2a76b9719d911017c592"; // for "hello"
if(md5("hello") == $hash) {
    echo "Verified";
}
```

Verifying `password_hash`

```php
<?php

$hash = "$2y$10$dNq8Edvsh3WIlTJ5o.ztqeXS/iMn1OmRAXkodrFUzP7ofDDcT/jJO";
$password = "hello";

if(password_verify($password, $hash)) {
    echo "Verified";
}
```

Execute External Command

> NOTE: DO NOT USE THIS COMMAND ON YOUR CODE UNLESS ABSOLUTELY NECESSARY.

`exec()` command is used to run external programs outside PHP. It is useful on certain tasks PHP cannot do, or have hard times doing it.

Here is an example program to open notepad using PHP. (Linux users, change string to your editor name)

```php
<?php

exec("c:/windows/system32/notepad.exe");    // opens notepad
```

PHP Include and Require

You can include and execute code from other PHP file into another PHP file. This opens up possibility to separate code into multiple parts and also helps us to re-use various parts instead of repeting the code.

```php
<?php
# index.php

echo "Index ";
require "hello.php";
echo " Page";
```

```php
<?php
# hello.php

echo "Hello";
```

> Output: Index Hello Page

Here in above example we are using `require` feature to import code from another file.

Similarly, we can use `include` too.

Difference between `include` and `require` is that if there is error importing on `include`, the code below will run.

```php
<?php
# index.php
include "file.php";
```

```php
<?php
# file.php
echo "Included File";
```

> Output: Included File

Include and Require "Once"

If you wish to include or require certain piece of code only once, you can add `_once` at the end i.e `include_once()` and `require_once()`

```php
<?php

include_once "hello.php";
require_once "hello.php";
```

```php
<?php
# hello.php
echo "Hello ";
```

> Output: Hello

Notice the output, we only get one "Hello" instead of normal two. This behaviour is intended for include.

Get and Put Files

This is the most basic type of file read and write feature in PHP. While it is basic, it boasts much dynamic features.

`text.txt:`

```
Hello Ram
```

`index.php`

```php
<?php

// Get Contents from File into A String
$text = file_get_contents("text.txt");

// Replace "Ram" with "Hari"
$replaced = str_replace("Ram", "Hari", $text);

// Save File Contents with changed data
file_put_contents("text.txt", $replaced);
```

Requests: GET and POST

In PHP, you can accept user input data (form) using GET and POST request global variables.

```html
<form method="GET" action="get.php">
  Name: <input type="text" name="name" />
  <button type="submit">Send</button>
</form>
```

`get.php`

```php
<?php

$name = $_GET['name'];

echo "User's name is $name";
```

Similarly, for POST request types:

```html
<form method="POST" action="post.php">
  Name: <input type="text" name="name" />
  <button type="submit">Send</button>
</form>
```

`post.php`

```php
<?php

$name = $_POST['name'];

echo "User's name is $name";
```

Now difference in GET and POST Types;

- GET Request has its parameters in the URL. Example, for above form, if you submit it, your URL will look like this: `get.php?name=Ram`. POST requests do not append the data in URL.
- Since POST request do not append data in URL, the data contents wil not be shown in browser's history.
- POST request can handle file uploads and large set of data, while GET is limited by the size of browser's address bar.

> Note: If you have a form, please use POST request when possible. GET request should be used on simple stuff like passing a page number.

Session

Session is a mechanism of remembering small piece of data for that browser. Session's data is stored in server side.

We can use session to pass data from one PHP file to another. File A store data in session, file B get data from session. This session data is stored per-user basis.

`set-session.php`

```php
<?php
session_start();

$_SESSION['name'] = "Ram";
```

`get-session.php`

```php
<?php
session_start();

echo "Hello" . $_SESSION['name'];
```

`delete-session.php`

```php
<?php
session_start();

// This unsets the variable
unset($_SESSION['name']);
```

Notice the `session_start()` at the start of PHP file. This mechanism tells PHP to use session data.

> Note: I would advise you not trust the session data. Make sure you sanitize before using it.

Cookies

Cookies are piece of data that is stored on browser, that later can be accessed by PHP. This is helpful to remember User's login state and track certain things. We can also identify which user is who based on cookie data (if implemented).

`set-cookie.php`

```php
<?php

$name = "name";
$value = "Ram";
$expire = time() + 60 * 60 * 24; // 1 day from today
$path = "/";

setcookie($name, $value, $expire, $path);
```

`get-cookie.php`

```php
<?php

echo $_COOKIE['name'];
```

Deleting cookie is a tricky part. We cannot just ask PHP to remove the cookie from browser, that feature is not available. We need to ask PHP to modify the cookie with negative time (i.e time before current time) as expiry date.

This makes cookie update and get deleted instantly as it has expired a long ago.

`delete-cookie.php`

```php
<?php

setcookie("name", "", time() - 100, "/")
```

OOP

You know what OOP means.

Class and Objects

Like other OOP supported language, PHP also has almost same features.

Here is an example class and object:

```php
<?php

class Animal {
    public $name = "Puppy";

    public function printName() {
        echo $this->name;
    }
}

$an = new Animal;
$an->printName(); // prints "Puppy"
```

Note the keyword `$this->`. That is used to call internal attributes and methods.

Unlike C++ and Java, we use arrow `->` to denote the methods and properties.

Example java code:

```
class Animal {
    public name = "Puppy";

    main() {
        Animal an = new Animal();
        sout(an.name);
    }
}
```

Same code on PHP:

```
<?php
class Animal{
    public $name = "puppy";
}

$an = new Animal();
echo $an->name;
```

Notice the difference on how we use `->` instead of dot (`.`) to call methods and attributes.

Visibility

PHP supports all visibility types that other language commonly use.

- public
- private
- protected

PHP also supports `static` method and attributes.

Static Methods and Attributes

Static methods are those which is not bound to an object. It is lifetime and is copied over to all objects.

Unlike regular attributes (variables) or methods (functions) inside class, `static` variables can be called without making new object.

```
<?php

class Animal {
    static $name = "puppy";
    static function sayName(){
        echo self::$puppy;
    }
}
echo Animal::$name; // call name directly
echo Animal::sayName(); // call function directly
```

Limitations: You cannot use `$this->` keyword inside static variable since we do not make an object to use static functions. `$this->` is only useful when referencing method or attribute of that object.

Constructor & Destructor

Constructor runs when new object is created or the class is first called.

We do not need to call constructor function ourselves. PHP does that automatically for us.

```php
<?php

class Animal {
    public function __construct() {
        echo "Calling Constructor";
    }
}

$an = new Animal(); // output: "Calling Constructor"
$an2 = new Animal(); // output: "Calling Constructor"
```

Constructor are useful when we want to perform certain action (like setting default values) when creating new object.

```php
<?php

class Animal {
    public $name;
    public function __construct($name) {
        $this->name = $name;
    }
}

$an = new Animal("Puppy");
echo $an->name; // prints: "Puppy"
```

Destructor are similar to constructor, they run automatically but destructor runs only when the class is free from memory using `unset()` or if the code execution ends for that file.

```php
<?php

class Animal {
    public function __destruct() {
        echo "Animal object destructed!";
    }
}

$an = new Animal();
unset($an); // free $an variable from memory
```

Destructor is not that commonly used but you need to know that it exists.

> Note: Unlike other methods, both constructor and destructor cannot `return` any values.

Inheritance

You can inherit properties (attributes) and methods (functions) from other class.

You may also override or overload them when needed.

```php
<?php

class A {
    public $age = 10;
    public function sayAge() {
        echo $this->age;
    }
}

class B extends A {
    public function __construct() {
        $this->age = 20;
    }
}

$b = new B;
$b->sayAge();   // Output: 20
```

Here, we are using `$age` property inside B and overriding its value with 20. We are also using `sayAge()` function from A class, which is not present in B.

> Note: You can extend only one parent class.

Implementation

When you want to define a set of functions beforehand to let other know what features it will have later, you can use the implementation.

```php
<?php

Interface AnimalImpl {
    public function eat();
}
```

```php
class Animal implements AnimalImpl {
    public function eat() {
        echo "Eating...";
    }
}
```

> Note: You need to implement ALL methods from interface when implementing it.

Traits

Traits are new concept from most of the other language. It allows us to inherit methods from multiple sources. This make our code more re-usable.

Unlike extending class with Inheritance, you can use multiple traits.

> Note: Traits are *used* not extended.

```php
trait PersonTrait {
    public function eat() {
        echo "Eating...";
    }
    public function walk(){
        echo "Walking...";
    }
}

class Student {
    use Person;
}

$s = new Student;
$s->eat();  // output: eating...
$s->walk(); // output: walking...
```

As said before you can use multiple traits at once.

```php
<?php

trait AnimalTrait {
    public function eat() {
        echo "Eating...";
    }
}

trait HumanTrait {
    public function speak() {
        echo "Speaking...";
    }
}
```

```php
class Person {
    use AnimalTrait, HumanTrait;
}

$p = new Person;
$p->eat();      // output: eating...
$p->speak();    // output: speaking...
```

Singleton

While we can create new object of class when we need, sometimes that is a bad practice. To avoid making duplicate object every time, we can follow singleton pattern. It is a bit advanced topic and there are debates whether you should follow it or not. My suggestion, use it when absolutely necessary or it provides clear benefit over making objects normally.

Singleton pattern is not PHP-specific. You can use similar method on other language that supports OOP.

We use static variable and method to store object (made once) and re-use it afterwards. While we are working with objects, we need to note that we are working with single object, no matter how many times we call it.

```php
<?php

class DB {
    static $instance = null;
    public function __construct() {
        echo "init... <br>";
    }

    static function getInstance() {
        if(self::$instance == null) {
            self::$instance = new DB;
        }

        return self::$instance;
    }
}

$db1 = new DB();
$db2 = new DB();
$db3 = new DB();

// Notice 3 "init" in output
echo "<hr>";
// Using Singleton
$db4 = DB::getInstance();
$db5 = DB::getInstance();
$db6 = DB::getInstance();
```

```
// We only get 1 this time.
// Object is made once and reused.
```

You can do all other stuff you can with class using this singleton pattern. It is just a method to create new object and how we use them.

If you want to learn more, you can google about this "Singleton Pattern in PHP" for more examples.

Composer

Composer is a tool made for PHP that allows us to use third-party packages and keep track of such packages.

Composer also helps us with autoloading class from different files than current.

You can get composer from https://getcomposer.org > Download > Download: composer-setup.exe

Open that file, choose "For all Users" and proceed.

When asked for PHP Path, it should be selected automatically with `c:/xampp/php/php.ini` if not, click on the `Open` button besides that and go to `Local Disk (C:)` > `xampp` > `php` > Choose `php.exe` file.

YOU NEED TO CHECK THE `ADD PHP TO PATH` OPTION

Other option are default. Make sure you have working internet connection when you are installing composer.

Init

Go to any folder you wish to work on, (empty folder if possible) and `Shift` + `Right Click` and choose `Open Powershell Window Here`.

Now to init current directory as composer project type:

```
composer init
```

It will ask 10 or so questions, just hit `Enter` on all of them. (Don't type enter, press the key on keyboard).

MySQL

MySQL is a relational database. It supports all commonly used SQL queries and work how a regular database to work.

There are two versions of MySQL commonly available out there: MySQL and MariaDB.

MySQL - Original Version, Open source, but later sold to Oracle. People do not trust Oracle.

MariaDB - Fork of MySQL. Made and Maintained by community. This is drop-in replacement for MySQL. It has almost same features and function almost same.

PHPMyAdmin

It is a tool bundled with XAMPP program (or you can download and run standalone) that helps us work with database.

There are other programs that have similar features to PMA. HeidiSQL and Dbeaver are two commonly used external database management tools.

Queries

- CREATE: Database, Table
- INSERT
- SELECT
- UPDATE
- DELETE
- DROP: Table, Database

CREATE DATABASE

```sql
CREATE DATABASE `testing`;
```

CREATE TABLE

```sql
CREATE TABLE `users` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `name` VARCHAR(50) NOT NULL,
    `email` VARCHAR(50) NOT NULL UNIQUE,
);
```

INSERT DATA

```sql
INSERT INTO `users`(name, email) VALUES('Ram', 'ram@gmail.com');
```

SELECT DATA

```
SELECT * FROM `users`;
```

UPDATE DATA

```
UPDATE FROM `users` SET `name` = 'Hari' WHERE `id` = 5;
```

DELETE DATA

```
DELETE FROM `users` WHERE `id` = 5;
```

PHP to Database Connections

In PHP, there are multiple ways to connecting a database.

Commonly used methods are mysqli_*, or MySQLi class or PDO.

We will be using PDO as it is secure by default and have support for various database types besides MySQL.

Connection

For connecting to a database, you will need following things:

- Database Name
- Database Hostname (commonly `localhost`)
- Database Username
- Database Password

In XAMPP, following is the default:

```
DB_USER     = "root"
DB_PASSWORD = ""
DB_HOST     = "localhost"
```

You are expected to make your own database for each project.

PDO

PDO is an acronym for PHP Data Objects. PDO is a lean, consistent way to access databases. PDO supports multiple database types, including MySQL, SQLite, PostgreSQL and many others via additional driver. For all database connection, it provides same API for developers to use.

Making a Connection

`db.php`

```
<?php

// Change DB_NAME's value to your
// db's name
$DB_NAME = "test";
$DB_USER = "root";
$DB_PASS = "";
$DB_HOST = "localhost";

// Some Configuration
$_DB_DSN="mysql:host=".$DB_HOST.";dbname=".$DB_NAME.";charset=utf8";
$_DB_OPT=[
    PDO::ATTR_ERRMODE               =>  PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE    =>  PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES      =>  false
];

// Try Connection
try{
    $pdo = new PDO($_DB_DSN, $DB_USER, $DB_PASS, $_DB_OPT);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
        die("Cannot Connect to Database!");
}
```

Make a new file `db.php` and put the contents of above there.

Running a Query

```
<?php

require_once "db.php";

$stmt = $pdo->query("SELECT * FROM `users`");
$stmt->execute();
$data = $stmt->fetchAll();

var_dump($data);
```

Fetching Data

There are two methods:

- `fetchAll()`

- `fetch()`

You use `fetchAll()` when you expect multiple rows of data example: data of 10 different users. Data will be returned as multi-dimensional array (nested)

You use `fetch()` when you expect to get only 1 data. You will data as array.

Prepare Statement

PDO comes with secure Prepared Statements. This will sanitize the data and minimize the problem of SQL injection.

```php
<?php

require_once "db.php";

$id = 10;

$stmt = $pdo->prepare("SELECT * FROM `users` WHERE `id` = ? ");
$stmt->execute([$id]);
$data = $stmt->fetch();

var_dump($data);
```

Notice the `[$id]` inside `->execute()` function.

PDO prepared statement expects data in array. So, even if there is one value, we need to pass it as array.

Count Statement

```php
<?php

require_once "db.php";

$stmt = $pdo->query("SELECT count(*) FROM `users`");
$stmt->execute();
$data = $stmt->fetchColumn();

var_dump($data);
```