For absolute beginners

Lecturer: Amer Jaganjac a.jaganjac@fontys.nl

Content

1. PHP Level 1: Understand the fundamentals of PHP, set up a local development environment, and write simple scripts.
2. PHP Level 2: Learn to handle user input through forms, validate data, and perform basic CRUD operations.
3. PHP Level 3: Introduce OOP concepts, PHP sessions, and deployment to a live server.

Level 1 - Topics

Introduction to PHP
Setting Up a Development Environment
Basic PHP Syntax
Control Structures
Loops
Basic Functions and Arrays

What is PHP

PHP is a popular server-side scripting language designed for web development.

Stands for “PHP: Hypertext Preprocessor”.

Embeds directly into HTML.

Powers websites like Facebook and WordPress.

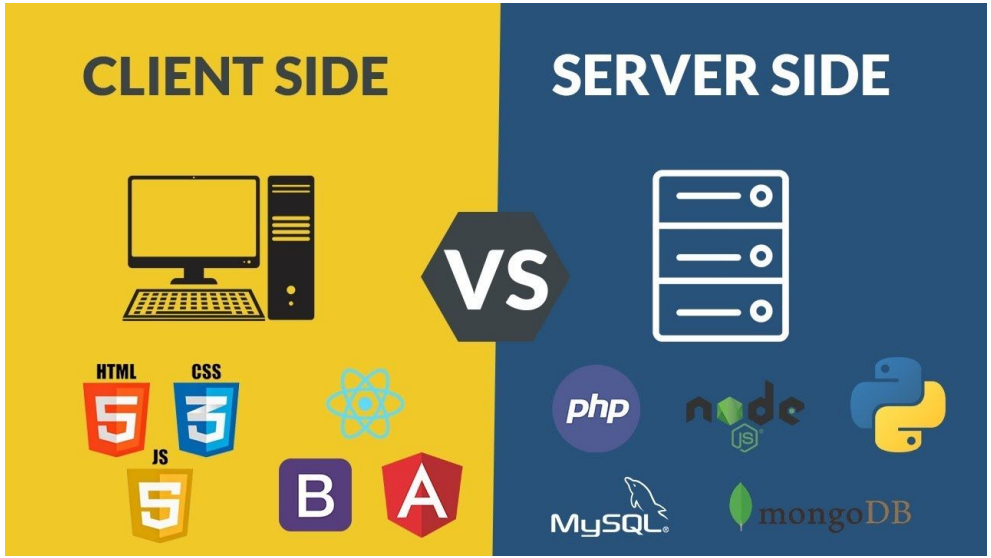
At time of writing this presentation, version 8.3 was the newest supported version.

https://www.php.net/

Client-side vs. Server-side Scripting

Client-side (e.g., JavaScript): Runs on the user's browser.

Server-side (e.g., PHP): Runs on the web server.



Setting Up a Development Environment

Install **XAMPP**/WAMP/MAMP.

Tip: When installing XAMPP ensure that the Apache and MySQL options are selected, as these are necessary for running PHP and MySQL.

Set up **VS Code** or another IDE.

https://www.apachefriends.org/ (XAMPP recommended because it is cross platform)

Create a project folder and structure

Right-click and create a new folder. Name it something relevant to your project, like my_php_project.

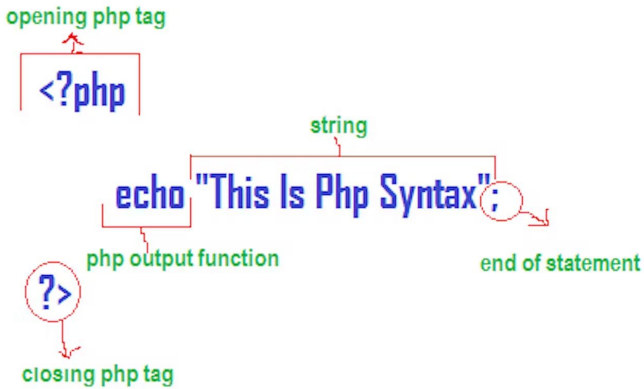
Subfolders:

- css/: For storing CSS files.
- js/: For storing JavaScript files.
- images/: For storing images used in the project.
- includes/: For reusable PHP files, such as headers, footers, and database connections.
- Example: header.php, footer.php, db_connect.php
- views/: For storing different pages or templates of your application.
- Example: home.php, contact.php

Also files: index.php: The main entry point for your project. This is the file that will be loaded when you access your project in the browser, and config.php: Configuration file for setting constants, database credentials, etc.

First php script

```
<?php
  echo "Hello, World!";
?>
```



Save as index.php and run on localhost.

PHP Variables and Data Types

Variables: Start with $, e.g., $name.

Data Types: String, Integer, Float, Boolean.

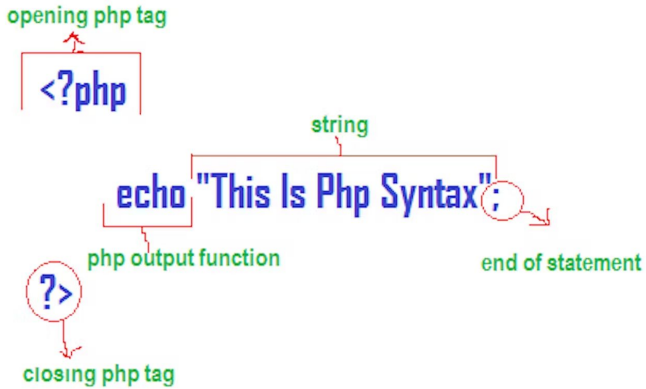
```
$name = "John";

$age = 25;

$is_student = true;
```

Control Structures (if, else, elseif)

```
$age = 20;
if ($age >= 18) {
  echo "You are an adult.";
} else {
  echo "You are a minor.";
}
```



Loops in PHP

```
for ($i = 1; $i <= 5; $i++) {
  echo $i;
}
```

- Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

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

function

DEFINITION

KEYWORD NAME ARGUMENT 1 ARGUMENT 2

```
function name( $arg1, $arg2 )
{
  //code;
}
```

CALL

EXPRESSION 1 EXPRESSION 2

```
name( $param1, $param2 );
```

A function is a block of statements wrapped inside the curly brackets { } indicating beggining and end of function code respectively.

Function arguments are specified under function definition, inside the round brackets after the function name.

Arguments act just like variables. When the function is called, the values get passed to function's arguments.

Parameters is also just another name for variables that are passed when the function is called.

Assignment 1

Set up your project, server environment and link the files.

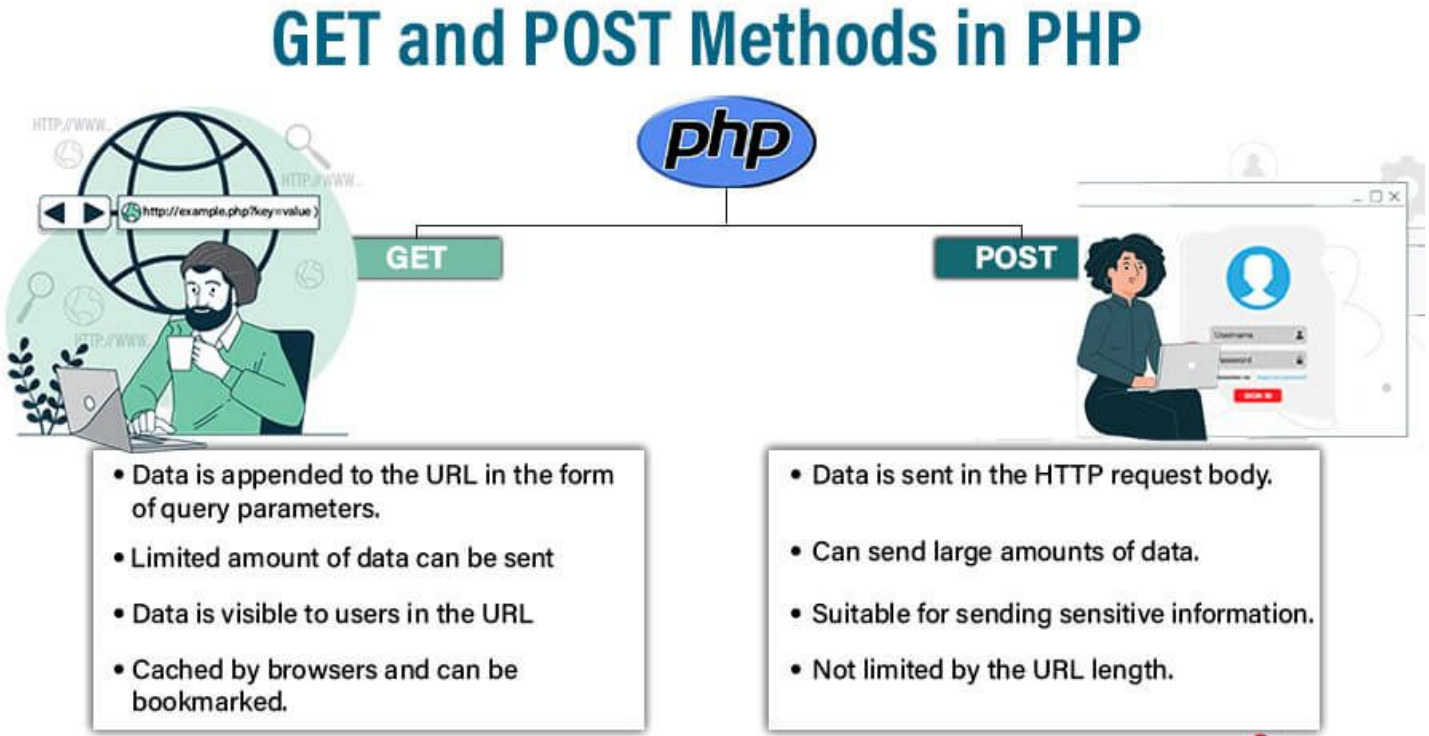
Follow the instructions from Assignment 1 pdf document.

Level 2 - Topics

Working with HTML Forms
Introduction to Databases (MySQL)
CRUD Operations in PHP

HTML Forms in PHP

- Forms are the main way to collect user input.
- GET vs. POST methods.
- Form elements: <input>, <textarea>, <select>, <button>.



Form example

```php
<?php
// Check if form data is received
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Collect form data and sanitize
    $name = htmlspecialchars($_POST['name']);
    $email = htmlspecialchars($_POST['email']);

    // Display the submitted information
    echo "<h2>Submitted Information:</h2>";
    echo "Name: " . $name . "<br>";
    echo "Email: " . $email . "<br>";
} else {
    echo "No data received.";
}
?>
```

process.php

```html
<!-- HTML Form -->
<form method="post" action="process.php">
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="name" required><br><br>

    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email" required><br><br>

    <input type="submit" value="Submit">
</form>
```

index.html

Introduction to Databases and MySQL

- Setting up a MySQL database using phpMyAdmin:
  - go to https://www.phpmyadmin.net/ and download phpMyAdmin. Not needed if you have during the installation of XAMPP, ensured that the Apache and MySQL options are selected.
- Open the XAMPP Control Panel.
  - Click on the "Start" buttons next to Apache and MySQL.
  - You should see green indicators confirming that both services are running.
  - Open your web browser and go to http://localhost/phpmyadmin. By default, use root as the username and leave the password field blank (unless you set a password during the installation of XAMPP).

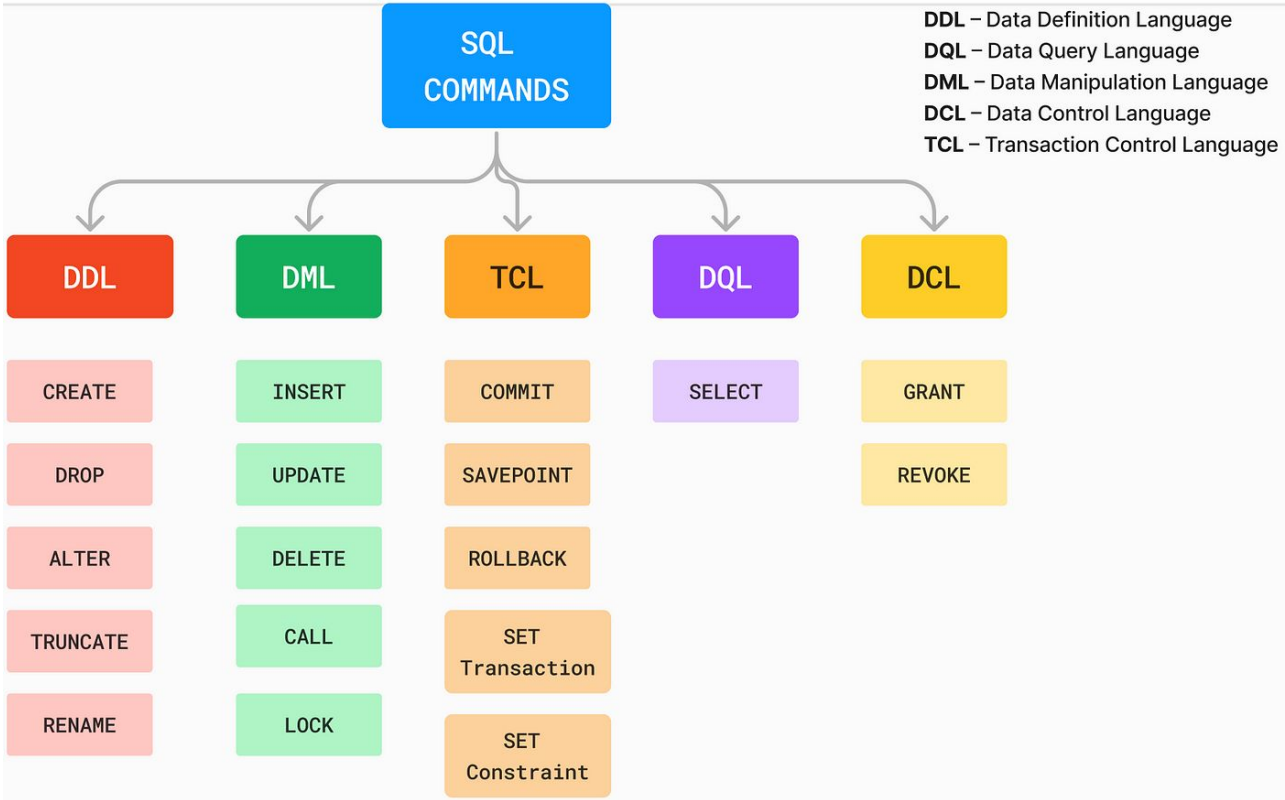
Create Your First Database

- In phpMyAdmin, click on the “Databases” tab at the top.
- Enter a name for your database in the “Create database” field (e.g., test_db).
- Click the “Create” button.
- Your database is now created and will be listed on the left panel.
- Click on your newly created database (test_db).
- Click on the “Create table” field and enter a name for your table (e.g., users).
- Specify the number of columns you want (e.g., 4), then click “Go”.
- For each column, enter the following details:
  - Column Name: e.g., id, name, email, age.
  - Type: e.g., INT for id, VARCHAR for name and email, INT for age.
  - Length/Values: e.g., 11 for id and age, 100 for name and email.
  - Index: Set the id column as the PRIMARY key and check the A_I (Auto Increment) box.

Check Your Database and Table

- Click on your table (users) on the left panel.
- Click on the “Structure” tab to view the table schema.
- Click on the “Insert” tab.
- Enter some sample data for each column (leave the id field empty as it will auto-increment).
- Click the “Go” button to insert the data.
- Click on the “Browse” tab to see the data you just inserted.
- You should see the rows and columns with the data you added.

SQL is the language of databases



Assignment 2

Work with databases and set up connections, create tables and display data.

Follow the instructions in Assignment 2 pdf.

Level 3 - Topics

Introduction to Object-Oriented Programming (OOP)
Working with Sessions and Cookies
Error Handling and Debugging
PHP and External APIs
Deployment of PHP Applications

Introduction to Object-Oriented Programming (OOP)

What is OOP?

Benefits: Reusability, Modularity, and Scalability.

Key concepts: Class, Object, Property, Method.

Class

```php
class Car {
  public $color;
  public $model;

  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }

  public function display() {
    return "This car is a " . $this->color . " " . $this->model;
  }
}

$car1 = new Car("red", "Toyota");
echo $car1->display();
```

OOP

```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();
?>
```

Inheritance and Polymorphism

```
class ElectricCar extends Car {
  public $batteryCapacity;
  public function display() {
    return parent::display() . " with a battery capacity of " . $this->batteryCapacity . " kWh.";
  }
}
$car2 = new ElectricCar("blue", "Tesla");
$car2->batteryCapacity = 100;
echo $car2->display();
```

Working with Sessions and Cookies

What are sessions and cookies?

Starting a session: `session_start()`.

Storing and accessing session variables.

Setting and retrieving cookies.

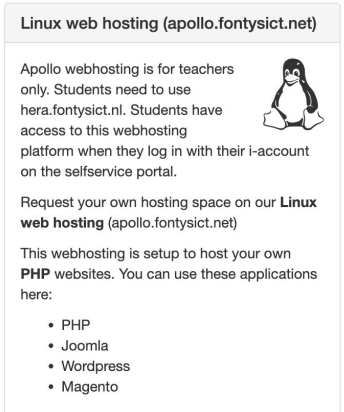
```php
// Starting a session
session_start();
$_SESSION['username'] = "JohnDoe";

// Accessing session data
echo "Welcome, " . $_SESSION['username'];

// Setting a cookie
setcookie("user", "JohnDoe", time() + (86400 * 30), "/"); // 1 day
```

Preparing for Deployment

- Clean up code and remove sensitive data.
- Test thoroughly on a local server.
- Choose a hosting provider: Shared hosting vs. Cloud services. Or Apollo FontysICT through Self service portal…
- Use version control (Git) for tracking changes.



What to learn next?

We recommend to learn Laravel: https://laravel.com/

“Laravel is a web application framework with expressive, elegant syntax. We’ve already laid the foundation — freeing you to create without sweating the small things.”

https://laravel.com/docs/11.x

https://www.youtube.com/watch?v=rIfdg_Ot-LI <- short introduction

Assignment 3

Final assignment: As a final project, you'll build a simple blog application. Start by setting up user authentication so that only logged-in users can create, edit, or delete posts. Then, implement CRUD operations for blog posts and display them in a paginated view to handle large amounts of data efficiently.

Follow instructions from Assignment 3