**SLIDE ONE**

Developed by **Rasmus Lerdorf in 1994**

Personal Home Page → **Php: Hypertext Preprocessor**

**--PHP** a widely-used open source server side scripting language use for web development.-

--Known for its ability to create **dynamic & interactive web pages.**

****What PHP can do:****

* generate dynamic page content
* can read/write/delete/open/close files on a server
* can collect form data
* can add/delete/modify data in a database
* send and receive cookies

➡ **How PHP is programmed:**

**in HTML (XHTML)**

<head>

</head>

<body>

<?php

?>

</body>

**➡ its own file**

<?php

echo (“body”);

Echo(“</body>);

?>

➡ **PHP include** → copies & content of the file at location of the include

<?php

include("fileName");

………………

?>

➡ **Scripting lang:** a type of programme language designed for automating tasks, often by interpreting code one line at a time, rather than requiring a full compilation process.

1. g. are series of commands that are able to be executed without need for compiling.

JavaScript, Python, Ruby

➡ **Client side scripting:** a type of web programming, where code runs in user’s browser after a page is loaded.

* used to create interactive & responsive websites

**examples**: JavaScript, HTML, CSS

**Features:**

* runs on client browser
* user can see/access client side code
* used to change content, validate form, create animations

➡ **Server side:** a type of web programming, where the code runs on a web server before the page is sent to the user browser.

* Used to create dynamic content.
* dynamic content, interact db, handle form submission.
* examples: PHP, Python, Ruby, Node.js

**Features:**

* runs on a server
* user cannot see server side code
* often used to fetch/store/process data in database

➡ **PHP Syntax**

<?php

echo "hello world";

?>

**echo to output text**

**comment: //, /\* \*/, #**

**end with ;**

➡ **How PHP works with Server**

* **User sends request** (browser sends a request to web server)
* **Web server receives a reques**t: If the file ends in .
* php, a server knows it needs help from PHP engine.
* **Server passes file to PHP interpreter**.
* **PHP code is executed:** the interpreter runs the code on a server. It can connect to Database, handle form data.
* **PHP generates HTML.**
* **HTML is sent to Browser.**
* **Browser displays the page.**

**Bug :** a capacity mistake or error in computer program that causes it to do something wrong at unexpected.

**SLIDE 2 AND 3**

**Variable**: a symbol that can store different values at different times.

**Variable Declaration**: begins with dollar sign followed by variable name. [$name]

**$name = "kofi";**

**case sensitive:** $A is not the same as $a;

**loosely typed** : figure the type based on the value;

**Array:** ["apple", "banana", "people"]

**String:** "hello"

**int** : 1234

Example

**$sum = 12 + 15;**

**echo $sum; // 27;**

## **ARRAY**

An ordered map. A map is a type that associates values to keys

| **array Creation** | **$fruits = ["apple", "banana"]** |
| --- | --- |
| **key1 -> value1,** | **echo $fruits[0]; // apple** |
| **key2 -> value2,** |  |

## **Control Structures**

Determine the flow of code within an application defining execution characteristics

**<body>**

**<h1 >PHP </h1>**

**<p>**

**$username = “kofi”;**

**Echo $username;**

**?>**

**</p>**

**</body>**

**</html>**

## **File inclusion**

* modular code
* code reuse
* load file into your PHP script using ( **require() or include()** )

**Require** will produce a fatal error

( E\_COMPILE\_ERROR() ) and stop the script

**while**

**Include** will only produce a warning

( E\_WARNING() ) and the script will continue.

**include()**

**<?php**

**echo “The main file”;**

**include(“reuse.php”);**

**echo”Hello World”;**

**?>**

**require()**

**<?php**

**echo “The main file”;**

**Require (“reuse.php”);**

**echo”Hello World”;**

**?>**

## **Advantages**

**Code reuse**,his reduces cost,

* increased reliability,
* improved consistency.

## **Modular** Page design: a web development and design approach that includes breaking webpage into reusable independent components.

These modules can be modern, faster navigation  
but navigate etc.

# **Key concept of Modular Page Design:**

* readability (reuse occurs multiple page)
* separation (cleaner structure)
* Maintainability (update in one place affects all)
* Scalability (maximal and minimal scanning code)
* consistency (ensures uniform look)

## **Example of modules**

* headers
* Navigation bar
* sidebar
* footer
* login form
* Product card

## **Benefit of MPD (Modular Page Design)**

* faster development
* cleaner codebase
* easier debugging
* easier HTML
* better team automation
* individual push

## **Function**

* executed by a call to a function
* will not execute immediately when a page loads
* when calling a function you pass any number of arguments as function returns value.

# **Declaring a function**

when declaring, start with the function statement then a name for your function having inside the parentheses id list of arguments separated by commas

### **Syntax:**

function functionName() {  
// code to be executed;  
}

**SLIDES 4 AND 5**

****Forms****

-To get input from the user and submit it to the web server for processing.

-On HTML that consist of graphical user interface items such as input boxes, check boxes, radio buttons.

- Submit info via POST method as key-value pairs in body of request.

**list of things needed to create a form.**

* <form> </form >tags.
* POST or GET - form submission method.
* Submission URL.
* input fields: input box, text area, button etc.

**form Input Types.**

* Text
* Radio button
* select /Drop down.
* Password
* check box
* features.

#### **HTML & PHP on a same file. Page 8**

**GET vs POST Method.**

* method through which browser sends infor to the server...
* enables communication between the client and server.
* browser => application running on computer system

**GET METHOD.**

-is required to submit HTML form data. The data is collected by a predefined $\_GET variable for processing.

-Variablesnames and values will be displayed in the URL, using GET method is visible to everyone in the browser address.

-Has limit of 256 characters.

-not secure to send sensitive info.

#### **Post method**

-uses predefined superglobal variable $\_POST

-does not have a restriction on the content of data sent

-nobody can see data being submitted

### **Form Data Validation**

process of ascertaining if data entered by a user in various form elements is acceptable for processing.

**Importance of form validation**

* checking if data meets specific requirements.
* essential for securing applications.
* proving data accuracy.
* providing meaningful feedback to users.
* avoid possible exceptions and runtime errors.

**Rules for form validation:**

* required fields
* length check
* email validation
* pattern matching
* number validation
* URL validation

**Types of form Validation:**

**Client side validation**: happens in user's web browser before form submission, provides immediate feedback to users. Commonly done with JavaScript.

**Server side validation**: after submission of form, server side validation happens; important for security features, data is checked on server.

**How to validate form in PHP:**

* Create a form.
* Collect form data.
* Perform form validation.
* Provide feedback.

**Server side Validation.**

2 Common approaches:

**Approach 1 : Single file**.

1. The form and validation code are both **in one file(form.php)**
2. use $\_SERVER["PHP\_SELF"] as form action.

**Approach 2: Separate file**

* One html file contains the form (for.html)
* One php processes/ validate input(process.php)

### **Understanding $\_SERVER["PHP\_SELF"]**

HTML form look like:

* **<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">**

* **method="post"**: sends data in HTTP request body
* **$\_SERVER["PHP\_SELF"]**: returns filename of current script
* - enables same page submission and validation
* **- $\_SERVER["PHP\_SELF"]** is a superglobal variable that returns filename of the currently executing script
* **- htmlspecialchars()**: converts special characters to HTML entities. It will replace HTML characters like < and > with & t; and &gt;. Prevents attackers from exploiting the code by injecting HTML or JavaScript code in form.
* - without **htmlspecialchars() PHP\_SELF** can be exploited:
* - **Cross Site Scripting (XSS)** is a type of computer security vulnerability typically found in web applications.

**Data validation in PHP**

* Use html specialchars()
* Remove spaces --> trim()
* Remove backslashes --> stripslashes ()

# **- Escaping User Inputs:** a technique used in data protection to ensure that any user-supplied data is treated as plain-text and not executable code or other harmful inputs.

* **Uses** htmlspecialchars() or filter\_var()

# **PHP required fields**

* **Use** PHP empty() to test if field is left blank
* **Display** a custom message if field is missing

# **PHP Validate Name**

if (!preg\_match("/^[a-zA-Z ]\*$/", $name)) {  
$nameErr = "Only letters and white space allowed";  
}

# **PHP validate Email**

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {  
$emailErr = "Invalid email format";  
}

# **PHP validate URL**

if (!filter\_var($website, FILTER\_VALIDATE\_URL)) {  
$websiteErr = "Invalid URL";  
}

# **File Modes in PHP**

* **"w"**: write only (creates file if not exists)
* **"r":** read only
* **"a":** write only append (write to end of file)
* **"w+**": reading and writing (creates):
* **"r+**": reading and writing
* **"a+**": reading and writing (append)
* **"x"**: create file for write only, fails if file already exists

**SLIDES 6**

# **File handling**

a process of interacting with files on the server such as reading, writing, creating new files or deleting existing ones

# **Types of file operations in PHP**

* **Reading files** (getting information from file)
* **Writing to files** (putting information in file)
* File metadata (getting information about file)
* File uploading

**File Handling functions**

* fopen(): open a file
* fclose(): close file
* fread(): read from a file
* fwrite(): write to a file
* file\_exists(): Check if a file exists
* unlink(): delete a file

**// Reading files**

$file = fopen("eg.txt","r"); // open in read mode  
$content = fread($file, filesize("eg.txt"));  
fclose($file);  
echo $content; // show content in a file

**// Writing files**

$file = fopen("eg.txt","w");  
fwrite($file, "Hello world");  
fclose($file);  
echo "file written successfully";

# **Deleting files**

file\_exists ("eg.txt");  
unlink ("eg.txt");  
echo "file deleted successfully";

# **File uploads**

fxns for file uploads

finfo and $\_FILES

**uploading file involves 2 main component**

1. client-side HTML form
2. server-side PHP script  
   [php.ini]

# **Creating HTML form for file uploading**

# -to allow users to upload files, you need a form that uses the post method and multipart/form-data encoding to send the uploaded file to the server.

-Create upload html file in your root directory.

**SLIDES 7**

# **Database with PHP**

-**PDO** : PHP Data Objects, allows developers  
to work with multiple database systems, through consistents  
API and offers advanced features such as prepare  
statements, error handling & transaction support:

**Why PDO over other methods**

* Database Independence
* Prepared Statements
* Advanced Features
* Consistents API

# **How to Send SQL Queries from PHP**

1. Connect to the database
2. Prepare the SQL query
3. Execute the SQL query
4. Check to make sure it worked

# **Database Connection**

<?php

$conn = mysql\_connect("localhost","root",

" ","database-name");

if (!$conn) {

die("Connection failed:".mysql\_connect\_error());

}

echo "Connection successfully";

?>

**SLIDES 8**

# **Upload image into database & display using PHP.**

# **Process to implement a file upload**

* Exactly in web app
* HTML form to upload image
* Upload a image to Server using PHP
* Store file name in a database using PHP's MySQL
* Retrieve images from a database & display in web page.

# **Files needed**

[db.php](https://db.php/" \t "https://chat.deepseek.com/a/chat/s/_blank) - upload.php

[index.php](https://index.php/" \t "https://chat.deepseek.com/a/chat/s/_blank) - preview.php

# Create database table

# Database configuration ([db.php](https://db.php/" \t "https://chat.deepseek.com/a/chat/s/_blank))

# Upload from HTML ([index.php](https://index.php/" \t "https://chat.deepseek.com/a/chat/s/_blank))

# Upload file to server before in DB ([upload.php](https://upload.php/" \t "https://chat.deepseek.com/a/chat/s/_blank))

# Preview /Display images from DB