PHP: El Cerebro Detrás del Telón

1. ¿Qué es PHP? La Analogía del Chef y el Restaurante 💪👁

Imagina que tu sitio web es un **restaurante**:

- **HTML** es el menú y la decoración del local (la estructura y presentación).
- **CSS** es el diseño de las mesas y los colores de las paredes (el estilo).
- **JavaScript** es el camarero que interactúa contigo en tu mesa (lo que sucede en tu navegador).
- **PHP** es el **Chef** que está en la cocina (el servidor).

PHP se encarga de recibir tu pedido (una solicitud web), va a la despensa (la base de datos **MySQL**), prepara el plato (procesa la información) y, finalmente, le entrega al camarero una página HTML ya terminada para que tú la veas.

Es un **lenguaje de *scripting* del lado del servidor** (Server-Side), lo que significa que todo su código se ejecuta *antes* de que la página llegue a tu navegador. Su nombre oficial es un acrónimo recursivo: **P**HP **H**ypertext **P**reprocessor. Está especialmente diseñado para el desarrollo web.

Usos Comunes de PHP:

- **Recolección de Datos de Formularios:** Guardar la información que un usuario introduce en un formulario de registro o contacto.
- **Generación de Contenido Dinámico:** Crear páginas personalizadas (como el *feed* de un usuario o el listado de productos de una tienda online).
- **Gestión de Bases de Datos:** Es el puente que conecta tu web con la información guardada en MySQL.
- **Gestión de Sesiones y Cookies:** Recordar quién eres entre una página y otra (mantener la sesión iniciada).

---

2. Sintaxis Básica: El Interruptor de PHP

Para que el servidor sepa que un trozo de código debe ser interpretado como PHP y no como HTML, necesitamos unas etiquetas especiales. Piensa en ellas como el **interruptor** que enciende y apaga el modo PHP dentro de un archivo.

El código PHP siempre comienza con `<?php` y termina con `?>`.

PHP

```
<!DOCTYPE html>
<html lang="es">
<head>
   <title>Ejemplo PHP</title>
```

```
</head>
<body>
   <h1>Bienvenido a mi sitio web</h1>

   <?php
      // Este código es PHP puro y se ejecuta en el servidor.
      $usuario = "Visitante Estelar";
      echo "<h2>¡Hola, " . $usuario . "!</h2>";
   ?>

   <p>Este es contenido HTML normal.</p>
</body>
</html>
```

- **`echo`:** Es la instrucción principal para **imprimir** o mostrar texto/HTML en la salida final de la página web.
- **El Punto y Coma (`;`):** Cada instrucción de PHP debe terminar con un punto y coma, como si fuera el punto final de una frase.

---

3. Variables: La Caja Fuerte de los Datos 🎁

Una **variable** en PHP es como una caja etiquetada donde guardas distintos tipos de información para usarla más tarde.

3.1. Declaración de Variables

La regla más importante de PHP: **todas las variables deben comenzar con el signo de dólar ($)**.

Sintaxis:

PHP

```
$nombre_de_la_variable = valor;
```

Reglas Clave:

1. **Siempre inician con `$`:** `$nombre` es válido.
2. **Sensible a Mayúsculas:** `$Nombre` y `$nombre` son variables diferentes.
3. **Tipado Dinámico:** PHP infiere el tipo de dato por el valor que le asignes.
4. **Concatenación (La Cola de Pegar):** Para unir una variable con una cadena de texto, utilizamos el operador **punto (`.`)**.

Ejemplo de Declaración y Uso:

```php
<?php
    // 1. Declaración de variables
    $nombre = "Carlos";          // Una cadena de texto (string)
    $puntuacion = 95;           // Un número entero (integer)
    $esta_conectado = true;     // Un valor booleano (boolean)

    // 2. Uso y Concatenación (unir texto)
    $saludo = "El usuario: " . $nombre . " tiene " . $puntuacion . " puntos.";

    echo "<h1>" . $saludo . "</h1>";
    // Muestra: El usuario: Carlos tiene 95 puntos.
?>
```

4. Tipos de Datos: Clasificando la Información

Los datos que guardas en tus variables se clasifican en tipos.

Tipo de Dato	Analogía Sencilla	Descripción	Ejemplo de Código
String	**La Nota Adhesiva**	Secuencias de caracteres (texto). Se encierran en comillas simples (`'`) o dobles (`"`).	`$producto = "Teclado Mecánico";`
Integer	**El Número Redondo**	Números enteros, positivos o negativos, sin decimales.	`$edad = 35;`
Float	**El Número Decimal**	Números con punto flotante (decimales).	`$precio = 49.99;`
Boolean	**El Interruptor ON/OFF**	Solo puede ser `true` (verdadero) o `false` (falso).	`$es_admin = false;`
Array	**La Estantería Organizada**	Estructuras para guardar múltiples valores en una sola variable.	`$colores = ["Rojo", "Verde", "Azul"];`
Null	**La Caja Vacía**	Representa la ausencia intencional de valor.	`$fecha_de_baja = null;`

Ejemplo de Arrays (La Estantería)

Los **Arrays Asociativos** te permiten usar nombres descriptivos (claves) en lugar de índices numéricos para acceder a los datos.

```php
<?php
    // Array Asociativo: La llave (key) es el nombre de la propiedad.
    $usuario = [
        "nombre" => "Elena",
        "email" => "elena@web.com",
        "activo" => true
    ];
```

```
    // Acceder a los datos usando la clave:
    $saludo_email = "Email del usuario: " . $usuario["email"];

    echo "<h3>" . $saludo_email . "</h3>";
    // Muestra: Email del usuario: elena@web.com
?>
```

5. Operadores: Las Herramientas de Cálculo y Lógica 🛠️

Los operadores son símbolos que nos permiten manipular valores y tomar decisiones en el código.

5.1. Operadores Aritméticos (Cálculos)

Operador	Operación	Ejemplo
`+`	Suma	`$a + $b`
`-`	Resta	`$a - $b`
`*`	Multiplicación	`$a * $b`
`/`	División	`$a / $b`
`%`	Módulo (Resto de la división)	`10 % 3` (Resultado: 1)

¡Por supuesto! Aquí tienes el artículo completo de **PHP** con todos los puntos que hemos desarrollado, desde los fundamentos hasta la Programación Orientada a Objetos y la conexión a MySQL, listo para tu enciclopedia.

⚙️ PHP: El Cerebro Detrás del Telón

1. ¿Qué es PHP? La Analogía del Chef y el Restaurante 💪🧿

Imagina que tu sitio web es un **restaurante**:

- **HTML** es el menú y la decoración del local (la estructura y presentación).
- **CSS** es el diseño de las mesas y los colores de las paredes (el estilo).
- **JavaScript** es el camarero que interactúa contigo en tu mesa (lo que sucede en tu navegador).
- **PHP** es el **Chef** que está en la cocina (el servidor).

PHP se encarga de recibir tu pedido (una solicitud web), va a la despensa (la base de datos **MySQL**), prepara el plato (procesa la información) y, finalmente, le entrega al camarero una página HTML ya terminada para que tú la veas.

Es un **lenguaje de *scripting* del lado del servidor** (Server-Side), lo que significa que todo su código se ejecuta *antes* de que la página llegue a tu navegador. Su nombre oficial es un acrónimo recursivo: **P**HP **H**ypertext **P**reprocessor. Está especialmente diseñado para el desarrollo web.

Usos Comunes de PHP:

- **Recolección de Datos de Formularios:** Guardar la información que un usuario introduce en un formulario de registro o contacto.
- **Generación de Contenido Dinámico:** Crear páginas personalizadas (como el *feed* de un usuario o el listado de productos de una tienda online).
- **Gestión de Bases de Datos:** Es el puente que conecta tu web con la información guardada en MySQL.
- **Gestión de Sesiones y Cookies:** Recordar quién eres entre una página y otra (mantener la sesión iniciada).

---

2. Sintaxis Básica: El Interruptor de PHP

Para que el servidor sepa que un trozo de código debe ser interpretado como PHP y no como HTML, necesitamos unas etiquetas especiales. Piensa en ellas como el **interruptor** que enciende y apaga el modo PHP dentro de un archivo.

El código PHP siempre comienza con `<?php` y termina con `?>`.

PHP
```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Ejemplo PHP</title>
</head>
<body>
    <h1>Bienvenido a mi sitio web</h1>

    <?php
        // Este código es PHP puro y se ejecuta en el servidor.
        $usuario = "Visitante Estelar";
        echo "<h2>¡Hola, " . $usuario . "!</h2>";
    ?>

    <p>Este es contenido HTML normal.</p>
</body>
</html>
```

- **`echo`:** Es la instrucción principal para **imprimir** o mostrar texto/HTML en la salida final de la página web.
- **El Punto y Coma (`;`):** Cada instrucción de PHP debe terminar con un punto y coma, como si fuera el punto final de una frase.

---

3. Variables: La Caja Fuerte de los Datos 🎁

Una **variable** en PHP es como una caja etiquetada donde guardas distintos tipos de información para usarla más tarde.

3.1. Declaración de Variables

La regla más importante de PHP: **todas las variables deben comenzar con el signo de dólar ($)**.

Sintaxis:

PHP

```
$nombre_de_la_variable = valor;
```

Reglas Clave:

1. **Siempre inician con $:** `$nombre` es válido.
2. **Sensible a Mayúsculas:** `$Nombre` y `$nombre` son variables diferentes.
3. **Tipado Dinámico:** PHP infiere el tipo de dato por el valor que le asignes.
4. **Concatenación (La Cola de Pegar):** Para unir una variable con una cadena de texto, utilizamos el operador **punto (`.`)**.

Ejemplo de Declaración y Uso:

PHP

```
<?php
    // 1. Declaración de variables
    $nombre = "Carlos";           // Una cadena de texto (string)
    $puntuacion = 95;           // Un número entero (integer)
    $esta_conectado = true;     // Un valor booleano (boolean)

    // 2. Uso y Concatenación (unir texto)
    $saludo = "El usuario: " . $nombre . " tiene " . $puntuacion . " puntos.";

    echo "<h1>" . $saludo . "</h1>";
    // Muestra: El usuario: Carlos tiene 95 puntos.
?>
```

---

4. Tipos de Datos: Clasificando la Información

Los datos que guardas en tus variables se clasifican en tipos.

Tipo de Dato	Analogía Sencilla	Descripción	Ejemplo de Código
String	**La Nota Adhesiva**	Secuencias de caracteres (texto). Se encierran en comillas simples (`'`) o dobles (`"`).	`$producto = "Teclado Mecánico";`
Integer	**El Número Redondo**	Números enteros, positivos o negativos, sin decimales.	`$edad = 35;`
Float	**El Número Decimal**	Números con punto flotante (decimales).	`$precio = 49.99;`
Boolean	**El Interruptor ON/OFF**	Solo puede ser `true` (verdadero) o `false` (falso).	`$es_admin = false;`
Array	**La Estantería Organizada**	Estructuras para guardar múltiples valores en una sola variable.	`$colores = ["Rojo", "Verde", "Azul"];`

Tipo de Dato	Analogía Sencilla	Descripción	Ejemplo de Código
Null	**La Caja Vacía**	Representa la ausencia intencional de valor.	`$fecha_de_baja = null;`

Ejemplo de Arrays (La Estantería)

Los **Arrays Asociativos** te permiten usar nombres descriptivos (claves) en lugar de índices numéricos para acceder a los datos.

PHP

```
<?php
    // Array Asociativo: La llave (key) es el nombre de la propiedad.
    $usuario = [
        "nombre" => "Elena",
        "email" => "elena@web.com",
        "activo" => true
    ];

    // Acceder a los datos usando la clave:
    $saludo_email = "Email del usuario: " . $usuario["email"];

    echo "<h3>" . $saludo_email . "</h3>";
    // Muestra: Email del usuario: elena@web.com
?>
```

---

5. Operadores: Las Herramientas de Cálculo y Lógica 🛠️

Los operadores son símbolos que nos permiten manipular valores y tomar decisiones en el código.

5.1. Operadores Aritméticos (Cálculos)

Operador	Operación	Ejemplo
`+`	Suma	`$a + $b`
`-`	Resta	`$a - $b`
`*`	Multiplicación	`$a * $b`
`/`	División	`$a / $b`
`%`	Módulo (Resto de la división)	`10 % 3` (Resultado: 1)

5.2. Operadores de Comparación (La Pregunta Lógica)

Se utilizan para comparar dos valores y siempre devuelven un valor **Booleano** (`true` o `false`).

Operador	Descripción
`==`	Igualdad (Débil)
`===`	**Igualdad Estricta (Valor y Tipo)**
`!=`	Desigualdad (Débil)
`!==`	**Desigualdad Estricta (Valor o Tipo)**
`>`	Mayor que
`<`	Menor que

Recomendación de Experto: ¡Usa Siempre `===` y `!==`!

La **Igualdad Estricta** evita que PHP intente convertir tipos de datos, lo que previene errores sutiles.

---

6. 🚦 Control de Flujo: Tomando Decisiones Lógicas (`if`, `else`, `elseif`)

El **Control de Flujo** permite que tu código decida qué bloques de instrucciones ejecutar basándose en condiciones lógicas.

6.1. La Sentencia `if...else` (Si pasa esto, si no, haz esto otro...)

```
<?php
   $edad_usuario = 17;

   if ($edad_usuario >= 18) {
      echo "<p>Acceso Concedido.</p>";
   } else {
      echo "<p>Acceso Denegado: Eres menor de edad.</p>";
   }
?>
```

6.2. La Sentencia `if...elseif...else` (Múltiples Opciones)

```
<?php
   $hora = date("H"); // Obtiene la hora actual del servidor

   if ($hora < 12) {
      echo "<h2>Buenos días.</h2>";
   } elseif ($hora < 19) {
      echo "<h2>Buenas tardes.</h2>"; // Se ejecuta si $hora es >= 12 y < 19
   } else {
      echo "<h2>Buenas noches.</h2>"; // Se ejecuta si $hora es >= 19
   }
?>
```

7. 🔁 Bucles (Loops): La Repetición Eficiente

Los **bucles** permiten ejecutar un bloque de código repetidamente hasta que se cumpla una condición de terminación.

7.1. Bucle `for`: El Contador Definido

Se utiliza cuando sabes de antemano cuántas veces necesitas repetir la acción.

```
<?php
   // Imprimir los números del 1 al 5.
```

```
    for ($i = 1; $i <= 5; $i++) {
        echo "<p>Contador: " . $i . "</p>";
    }
?>
```

7.2. Bucle `foreach`: Recorriendo la Estantería (Arrays)

Este es el bucle más común en PHP. Está **hecho para recorrer *Arrays*** sin preocuparte por los índices numéricos.

```
<?php
    $productos = ["Taza", "Gorra", "Camiseta"];

    echo "<h3>Lista de Productos Disponibles:</h3>";

    // foreach (array_a_recorrer as valor_actual)
    foreach ($productos as $producto) {
        echo "<li>" . $producto . "</li>";
    }

    // Para Arrays Asociativos, puedes acceder también a la clave:
    $usuario = ["nombre" => "Ana", "rol" => "editor"];

    foreach ($usuario as $clave => $valor) {
        echo "<p>Propiedad **" . $clave . "** tiene el valor: " . $valor . "</p>";
    }
?>
```

8. 🏗️ Funciones: Reutilizando Código

Una **Función** es un bloque de código que se agrupa, se nombra y se puede llamar repetidamente desde cualquier parte de tu programa.

8.1. Declaración de Funciones

Se usa la palabra clave `function` seguida de un nombre y los **parámetros** (la información de entrada). La palabra clave `return` devuelve un valor al lugar donde se llamó.

```
<?php
    // Declaración: La función que calcula el área.
    function calcularAreaRectangulo($ancho, $alto) {
        $area = $ancho * $alto;
        return $area;
```

```
    }

    // Llamada o Ejecución de la función
    $area1 = calcularAreaRectangulo(10, 5); // $area1 vale 50

    echo "<p>El área del primer rectángulo es: " . $area1 . "</p>";
?>
```

8.2. Funciones Integradas (Built-in) Útiles

PHP tiene miles de funciones pre-construidas.

Función	Propósito	Ejemplo de Uso
`strlen()`	Contar el número de caracteres de un `string`.	`strlen("Hola");` // Resultado: 4
`str_replace()`	Reemplazar una parte de un texto por otro.	`str_replace("mal", "bien", $texto);`
`rand()`	Generar un número entero aleatorio.	`rand(1, 100);`
`count()`	Contar cuántos elementos tiene un array.	`count($productos);`

9. 🧱 Programación Orientada a Objetos (POO): El Kit de Construcción

La POO organiza el código en **entidades lógicas** llamadas **Objetos**.

9.1. Clases y Objetos

Una **Clase** es el molde (**`Usuario`**). Un **Objeto** es una instancia real creada a partir de ese molde (**`$usuario1`**). Los objetos tienen **Propiedades** (datos) y **Métodos** (acciones).

```
<?php
    class Usuario {
        public $nombre;

        // El constructor se llama al crear el objeto.
        public function __construct($nombre_inicial) {
            $this->nombre = $nombre_inicial;
        }

        public function saludar() {
            return "Hola, soy **" . $this->nombre . "**";
        }
    }
```

```
    // Creamos un objeto (instancia)
    $usuario1 = new Usuario("Marta");

    // Llamar a los métodos (acciones):
    echo "<p>" . $usuario1->saludar() . "</p>";
?>
```

10. 💾 Conexión a MySQL: Accediendo a la Despensa

Para la conexión a la base de datos se recomienda usar la librería **MySQLi** (MySQL Improved) o PDO. Aquí usamos MySQLi para una introducción sencilla.

Ejemplo de Conexión y Consulta (CRUD: Read)

```
<?php
    // 1. Datos de Conexión (Asegúrate de que XAMPP esté corriendo)
    $servidor = "localhost";
    $usuario = "root";
    $password = ""; // Contraseña vacía por defecto en XAMPP
    $bd = "mi_enciclopedia_web";

    // 2. Crear la Conexión
    $conexion = new mysqli($servidor, $usuario, $password, $bd);

    // 3. Revisar si hay errores
    if ($conexion->connect_error) {
        die("❌ Error al conectar con la base de datos: " . $conexion->connect_error);
    }
    echo "✅ Conexión exitosa a la base de datos.<br>";

    // 4. Ejecutar una Consulta (SQL)
    $sql = "SELECT titulo FROM articulos LIMIT 3";
    $resultado = $conexion->query($sql);

    // 5. Procesar los Resultados
    if ($resultado->num_rows > 0) {
        while($fila = $resultado->fetch_assoc()) {
            echo "<li>" . $fila["titulo"] . "</li>";
        }
    } else {
        echo "<p>No se encontraron artículos.</p>";
    }

    // 6. Cerrar la Conexión
```

```
    $conexion->close();
?>
```

11. ♡ Validación y Limpieza de Datos: La Puerta de Seguridad

¡Nunca confíes en los datos que vienen del navegador! La **validación** verifica que los datos sean correctos, y la **limpieza** (sanitización) los hace inofensivos.

11.1. Limpieza (Sanitización)

- **`htmlspecialchars()`:** Esencial para prevenir ataques de inyección de código (XSS) al convertir caracteres peligrosos (`<`, `>`) en texto inofensivo.

11.2. Funciones de Validación Útiles

Función	Propósito
`isset()`	Comprueba si una variable **existe**.
`empty()`	Comprueba si una variable está vacía (cadena vacía, 0, `false`).
`filter_var()`	Valida formatos específicos como emails o URLs.

Ejemplo Completo de Validación de Entrada:

```
<?php
    $email_recibido = "ejemplo_invalido.com";
    $nombre_recibido = "Usuario con guion";

    // 1. Limpieza de datos (Sanitización)
    $nombre_limpio = htmlspecialchars($nombre_recibido);

    // 2. Validación de formato de Email
    if (filter_var($email_recibido, FILTER_VALIDATE_EMAIL)) {
        echo "<p>El email es válido.</p>";
    } else {
        echo "<p>El email **" . $email_recibido . "** NO tiene un formato válido.</p>";
    }
?>
```