

# Fundamentos básicos PHP

# Sirviendo HTML Directamente con PHP

PHP puede generar contenido HTML directamente usando la función `echo` o `print`. Esta es la forma más básica de crear páginas web dinámicas, donde el servidor procesa el código PHP y envía HTML puro al navegador.

Cuando el navegador solicita un archivo `.php`, el servidor web ejecuta el código PHP primero y luego envía el resultado HTML al cliente. Esto significa que el usuario nunca ve el código PHP, solo el HTML resultante.

- ❏ **Ventajas:** Control total sobre el HTML generado, posibilidad de crear contenido dinámico basado en datos del servidor, y separación clara entre lógica de servidor y presentación.

## Ejemplo básico:

```
<?php
echo "<h1>¡Hola Mundo!</h1>";
echo "<p>Esta página fue generada con PHP</p>";
?>
```



## Ejercicio Práctico

Crea un archivo llamado `saludo.php` y escribe código PHP que genere una página HTML completa con:

- Un título principal
- Un párrafo de bienvenida
- Una lista con tus hobbies favoritos

Usa solo funciones `echo` para generar todo el HTML.

# Código PHP con Texto Encadenado

## Concatenación de Strings

En PHP, utilizamos el operador punto (.) para unir cadenas de texto. Esta técnica es fundamental para crear contenido HTML dinámico que combine texto fijo con variables y datos dinámicos.

La concatenación te permite construir mensajes personalizados, generar HTML con contenido variable, y crear URLs dinámicas de forma eficiente.

### Operadores de concatenación:

- . - Une dos strings
- .= - Añade texto al final de una variable existente

```
<?php
$nombre = "Carlos";
$apellido = "García";
echo "Hola " . $nombre . " " . $apellido;

// Usando .=
$mensaje = "Hola ";
$mensaje .= $nombre;
$mensaje .= " " . $apellido;
echo $mensaje;
?>
```

```
string will gomenals>
exp>
----- string concatiration,
sulfortir sinerhitul);
php; ;
"dist_aopsant>
>> grins concation

nixis: ≡ (concatiation);
brmes< /y>
"coontation"
// yurdut>
```

### Concatenación en HTML:

```
<?php
$usuario = "Ana";
$edad = 25;
echo "<div class='perfil'>";
echo "<h2>" . $usuario . "</h2>";
echo "<p>Edad: " . $edad . " años</p>";
echo "</div>";
?>
```

📌 **Tip:** Para cadenas largas con muchas variables, considera usar comillas dobles: "Hola \$nombre \$apellido"

# Insertando Código PHP dentro del HTML

## Etiquetas PHP Básicas

Usa `<?php ... ?>` para insertar código PHP en cualquier parte de tu documento HTML. Esta es la forma estándar y más recomendada.

```
<?php echo "Contenido dinámico"; ?>
```

## Etiquetas Cortas

Las etiquetas `<? ... ?>` son más breves pero requieren configuración especial del servidor. No se recomienda su uso en producción.

```
<? echo $variable; ?>
```

## Echo Corto


Para mostrar variables rápidamente, usa `<?= ... ?>`. Es equivalente a `<?php echo ... ?>` y muy útil en plantillas.

```
<?= $nombre ?>
```

La mezcla de PHP y HTML permite crear páginas dinámicas donde el contenido puede cambiar según diferentes condiciones. Puedes insertar código PHP en cualquier lugar del documento HTML: dentro de atributos, contenido de etiquetas, o incluso para generar etiquetas completas dinámicamente.

## Ejemplo práctico de integración:

```
<!DOCTYPE html>
<html>
<head>
  <title><?php echo "Mi sitio - " . date('Y'); ?></title>
</head>
<body>
  <h1><?= "Bienvenido, hoy es " . date('d/m/Y') ?></h1>
  <p>Hora actual: <?php echo date('H:i:s'); ?></p>
</body>
</html>
```

 **Ejercicio:** Crea una página HTML que muestre la fecha y hora actual en diferentes formatos usando PHP. Incluye al menos 3 formatos diferentes de fecha y 2 de hora.

Documentación: [PHP: date - Manual](#)

# Uso de Variables en PHP



## Declaración de Variables

En PHP, todas las variables comienzan con el símbolo \$. **No necesitas declarar el tipo de variable**; PHP lo determina automáticamente según el valor asignado.

```
$nombre = "María";  
$edad = 28;  
$activo = true;
```



## Tipos de Datos

PHP soporta varios tipos: **string (texto)**, **integer (números enteros)**, **float (decimales)**, **boolean (verdadero/falso)**, **array**, y **object**.

```
$precio = 19.99;  
$disponible = true;  
$descripcion = "Producto nuevo";
```



## Ámbito de Variables

Las variables pueden ser locales (dentro de funciones) o globales (accesibles desde cualquier lugar). También existen variables superglobales como `$_GET` y `$_POST`.

```
$global = "Visible en todo el script";  
function ejemplo() {  
    $local = "Solo aquí";  
}
```

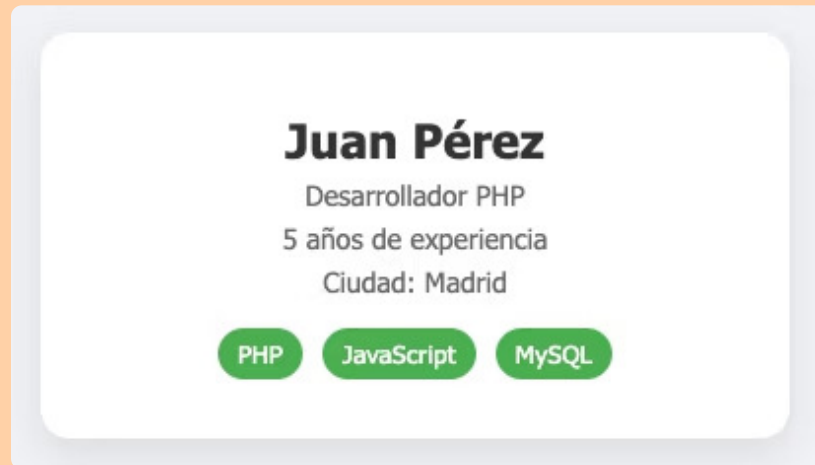
Las **variables** son **contenedores que almacenan datos** que pueden cambiar durante la ejecución del programa. En PHP, las **variables son dinámicas**, lo que significa que **pueden cambiar de tipo durante la ejecución**.

## Ejemplo práctico con variables:

```
<?php  
$titulo = "Mi Portfolio";  
$descripcion = "Desarrollador web especializado en PHP";  
$experiencia = 3;  
$disponible = true;  
  
echo "<h1>" . $titulo . "</h1>";  
echo "<p>" . $descripcion . "</p>";  
echo "<p>Experiencia: " . $experiencia . " años</p>";  
  
if ($disponible) {  
    echo "<p>Estado: Disponible para proyectos</p>";  
}  
?>
```

# Uso de Variables en PHP

- 📄 **Ejercicio:** Crea un perfil personal usando variables para almacenar tu información: nombre, profesión, años de experiencia, ciudad, y tres habilidades principales. Genera HTML dinámicamente. El resultado debe verse así:





# Uso de Arrays en PHP

Los arrays son estructuras de datos que permiten almacenar múltiples valores en una sola variable. PHP ofrece arrays indexados (con números) y asociativos (con claves personalizadas), ambos extremadamente útiles para organizar y manipular datos.

## Arrays Indexados

Los elementos se acceden mediante números, empezando desde 0.

```
$colores = array("rojo", "azul",  
"verde");  
// O sintaxis corta:  
$frutas = ["manzana", "naranja",  
"plátano"];  
  
echo $frutas[0]; // "manzana"
```

## Arrays Asociativos

Usa claves personalizadas para acceder a los valores, ideal para datos estructurados.

```
$persona = array(  
"nombre" => "Carlos",  
"edad" => 30,  
"ciudad" => "Madrid"  
);  
  
echo $persona["nombre"]; // "Carlos"
```

## Arrays Multidimensionales

Arrays dentro de arrays, perfectos para datos complejos como tablas o listas de registros.

```
$estudiantes = array(  
array("Ana", 22, "Informática"),  
array("Luis", 24, "Diseño"),  
array("Sara", 21, "Marketing")  
);
```

## Funciones útiles para arrays:

- `count()` - Cuenta elementos
- `array_push()` - Añade elementos al final
- `array_merge()` - Combina arrays
- `in_array()` - Busca un valor
- `array_keys()` - Obtiene todas las claves

```
<?php  
$productos = ["Laptop", "Mouse", "Teclado"];  
echo "<h3>Productos (" . count($productos) . ")</h3>";  
echo "<ul>";  
foreach ($productos as $producto) {  
    echo "<li>" . $producto . "</li>";  
}  
echo "</ul>";  
?>
```



- 📝 **Ejercicio:** Crea un array asociativo con información de 5 países (nombre, capital, población). Genera una tabla HTML que muestre toda la información usando un bucle foreach.

# Uso de Condicionales en PHP

Las estructuras condicionales permiten que tu código tome decisiones basadas en diferentes condiciones. Son esenciales para crear páginas web dinámicas que respondan de manera diferente según los datos del usuario o el estado de la aplicación.

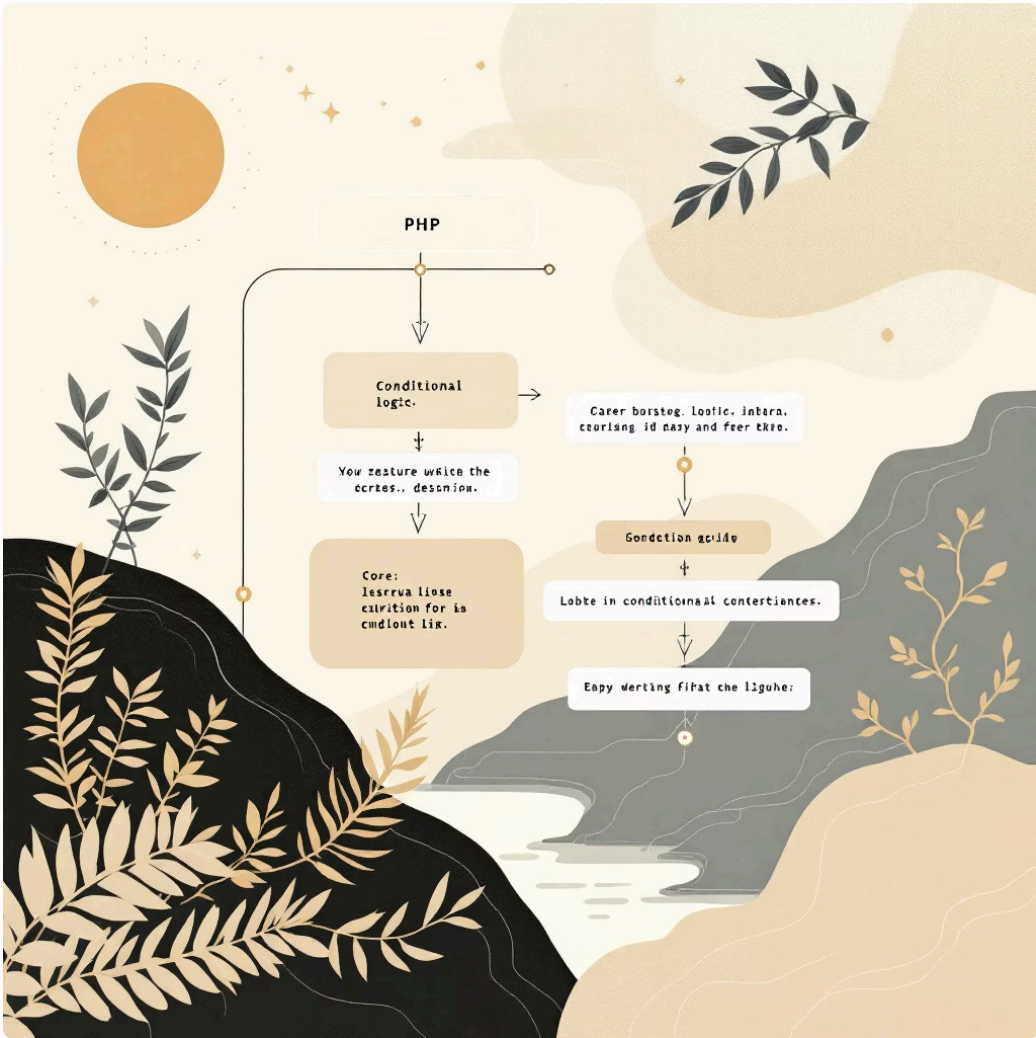
1	2	3
<b>IF Simple</b>	<b>IF-ELSE</b>	<b>ELSEIF</b>
Ejecuta código solo si la condición es verdadera.	Ejecuta una acción u otra según la condición.	Evalúa múltiples condiciones en secuencia.
<pre>if (\$edad &gt;= 18) {     echo "Eres mayor de edad"; }</pre>	<pre>if (\$usuario_logueado) {     echo "Bienvenido de vuelta"; } else {     echo "Por favor, inicia sesión"; }</pre>	<pre>if (\$nota &gt;= 9) {     echo "Sobresaliente"; } elseif (\$nota &gt;= 7) {     echo "Notable"; } else {     echo "Necesita mejorar"; }</pre>

## Operadores de Comparación:

- == - Igual (valor)
- === - Idéntico (valor y tipo)
- != o <> - Diferente
- !== - No idéntico
- <, >, <=, >= - Comparaciones numéricas


## Operadores Lógicos:

- && o and - Y lógico
- || o or - O lógico
- ! - Negación



## Switch Statement:

```
<?php  
$dia = date('w');  
switch ($dia) {  
    case 0:  
        echo "Domingo";  
        break;  
    case 1:  
        echo "Lunes";  
        break;  
    case 6:  
        echo "Sábado";  
        break;  
    default:  
        echo "Día laborable";  
}  
?>
```

 **Ejercicio:** Crea un sistema de calificaciones que tome una puntuación numérica (0-100) y muestre la calificación correspondiente: Suspenso (0-49), Aprobado (50-69), Notable (70-89), Sobresaliente (90-100). Incluye validación para números fuera del rango.




# Uso de Bucles en PHP

Los bucles permiten ejecutar código repetidamente, lo que es fundamental para procesar listas de datos, generar contenido HTML dinámico, y automatizar tareas repetitivas. PHP ofrece varios tipos de bucles para diferentes situaciones.

01	02	03
<h2>Bucle FOR</h2> <p>Ideal cuando <b>conoces exactamente cuántas veces necesitas repetir una acción</b>. Perfecto para contadores y rangos específicos.</p> <pre>for (\$i = 1; \$i &lt;= 5; \$i++) {     echo "&lt;p&gt;Iteración número: " . \$i . " &lt;/p&gt;"; }</pre>	<h2>Bucle WHILE</h2> <p>Continúa ejecutándose <b>mientras la condición sea verdadera</b>. Útil cuando <b>no sabes cuántas iteraciones necesitarás</b>.</p> <pre>\$contador = 1; while (\$contador &lt;= 3) {     echo "&lt;p&gt;Contador: " . \$contador . " &lt;/p&gt;";     \$contador++; }</pre>	<h2>Bucle FOREACH</h2> <p>Específicamente diseñado para <b>recorrer arrays</b>. Es la forma más <b>elegante y eficiente de procesar listas de datos</b>.</p> <pre>\$nombres = ["Ana", "Luis", "Carmen"]; foreach (\$nombres as \$nombre) {     echo "&lt;p&gt;Hola " . \$nombre . "&lt;/p&gt;"; }</pre>

## Foreach con Arrays Asociativos:

```
<?php  
$estudiantes = array(  
    "Ana" => 85,  
    "Luis" => 92,  
    "Carmen" => 78,  
    "David" => 88  
);  
  
echo "<h3>Calificaciones de Estudiantes</h3>";  
echo "<ul>";  
foreach ($estudiantes as $nombre => $nota) {  
    echo "<li>" . $nombre . ": " . $nota . " puntos</li>";  
}  
echo "</ul>";  
?>
```

 **Ejercicio Práctico:**

Crea una tabla HTML de multiplicar usando bucles anidados. La tabla debe mostrar las tablas del 1 al 10, con formato HTML apropiado (thead, tbody, etc.). Usa diferentes tipos de bucles para practicar.

## Control de Bucles:

- break** - Termina el bucle inmediatamente
- continue** - Salta a la siguiente iteración

# Ejercicio bucles

- 📄 **Ejercicio:** Crea una tabla HTML de multiplicar usando bucles anidados. La tabla debe mostrar las tablas del 1 al 10, con formato HTML apropiado (thead, tbody, etc.). Usa diferentes tipos de bucles para practicar.

# Uso de Funciones en PHP

Las funciones son bloques de código reutilizable que realizan tareas específicas. Permiten organizar el código de manera eficiente, evitar repetición, y crear programas más mantenibles y legibles.

f(x)

Definir Funciones

Una función se define una vez y puede ser llamada múltiples veces desde cualquier parte del código.

```
function saludar($nombre) {  
    return "¡Hola " . $nombre . "!";  
}  
  
echo saludar("María");
```

Parámetros y Argumentos

Las funciones pueden recibir datos de entrada (parámetros) y devolver resultados procesados.

```
function calcular($a, $b, $operacion = '+') {  
    switch($operacion) {  
        case '+': return $a + $b;  
        case '-': return $a - $b;  
        case '*': return $a * $b;  
        default: return 0;  
    }  
}
```

Valores de Retorno

Usa return para devolver un resultado. Sin return, la función devuelve NULL.

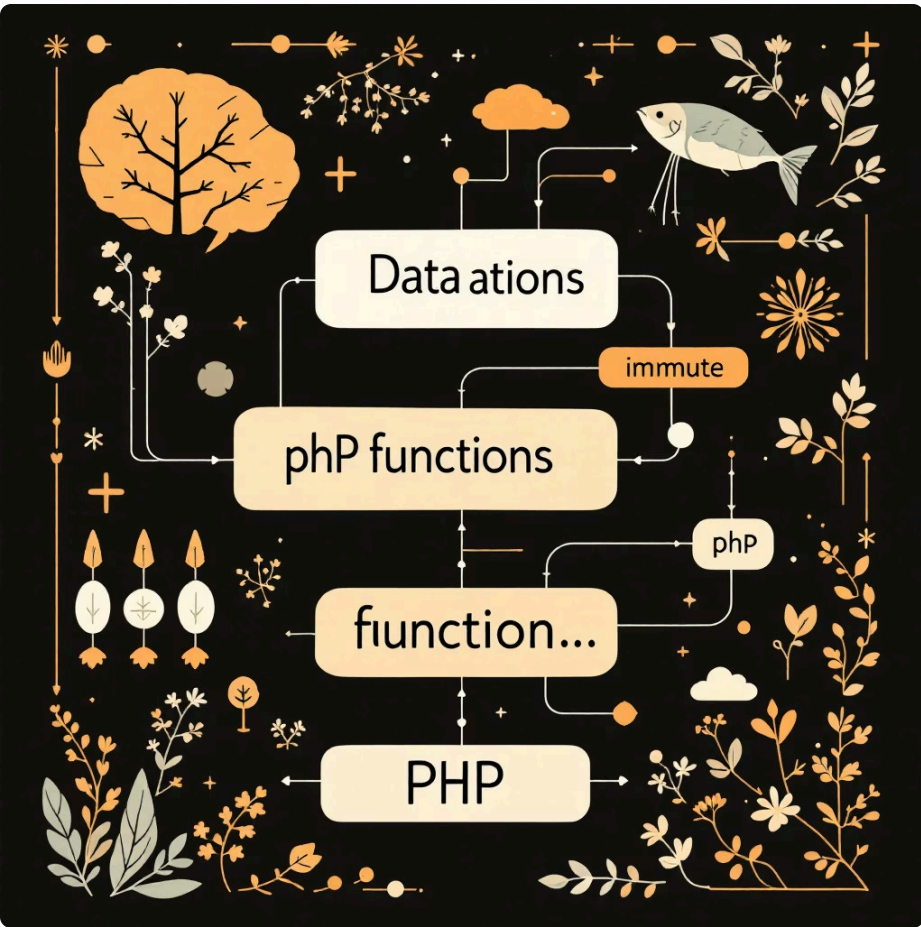
```
function generarHTML($titulo,  
$contenido) {  
    return "<div><h2>" . $titulo .  
        "</h2><p>" . $contenido . "</p>  
</div>";  
}
```

## Funciones para generar HTML:

```
<?php  
function crearTarjeta($nombre, $cargo, $email, $telefono) {  
    $html = "<div class='tarjeta'>";  
    $html .= "<h3>" . $nombre . "</h3>";  
    $html .= "<p><strong>" . $cargo . "</strong></p>";  
    $html .= "<p>Email: " . $email . "</p>";  
    $html .= "<p>Teléfono: " . $telefono . "</p>";  
    $html .= "</div>";  
    return $html;  
}  
  
// Uso de la función  
echo crearTarjeta("Ana García", "Desarrolladora",  
    "ana@email.com", "123-456-789");  
echo crearTarjeta("Luis Martín", "Diseñador",  
    "luis@email.com", "987-654-321");  
?>
```

## Ámbito de Variables en Funciones:

Las variables dentro de funciones son locales. Para usar variables globales, usa la palabra clave global.



## Funciones con Parámetros Opcionales:

```
function crearBoton($texto, $clase = "btn", $activo = true) {  
    $disabled = !$activo ? " disabled" : "";  
    return "<button class='" . $clase . "' .  
        $disabled . ">" . $texto . "</button>";  
}  
  
// Diferentes formas de llamar la función  
echo crearBoton("Enviar");  
echo crearBoton("Cancelar", "btn-secondary");  
echo crearBoton("Bloqueado", "btn", false);
```

# Ejercicio bucles + funciones

📄 **Ejercicio:** Crea una variable que contenga un array con datos de estudiantes, con el siguiente formato:

```
$personas = [  
  [  
    "nombre" => "Ana",  
    "apellidos" => "Garcia",  
    "puesto" => "Desarrollador Frontend",  
    "experiencia" => "5 años",  
    "skills" => ["HTML", "CSS", "JAVASCRIPT"]  
  ],  
  [  
    "nombre" => "Carlos",  
    "apellidos" => "Martínez",  
    "puesto" => "Desarrollador Backend",  
    "experiencia" => "3 años",  
    "skills" => ["PHP", "MySQL", "Laravel"]  
  ]  
];
```

Crea una función que reciba por parámetro el nombre, apellidos, puesto, experiencia y skills (este último será un array). Esta función debe devolver el código HTML de la tarjeta de un estudiante.

Recorre tu array de personas, por cada elemento llama a la función que acabas de crear y pinta el HTML resultado