Tema 2: PHP 8.4 Básico

Programación Web - Grado Superior

Índice

Características destacadas de PHP 8.4

- Property Hooks
- Asymmetric Visibility
- Nueva API DOM
- Nuevas funciones array_*()

Introducción a PHP 8.4

¿Qué es PHP?

PHP (acrónimo recursivo de "PHP: Hypertext Preprocessor") es un lenguaje de programación de código abierto especialmente diseñado para el desarrollo web.

PHP se ejecuta en el lado del servidor, generando HTML que se envía al cliente. El código PHP no es visible para el usuario final.

Características principales:

- Lenguaje de script del lado del servidor
- Orientado principalmente al desarrollo web
- Puede ser embebido en HTML
- Soporte para múltiples bases de datos
- Amplia comunidad y documentación

Evolución de PHP

- **1994:** Rasmus Lerdorf crea PHP/FI
- **2000:** PHP 4.0 - Introduce el motor Zend Engine
- **2004:** PHP 5.0 - Mejora el soporte para POO
- **2015:** PHP 7.0 - Mayor rendimiento y tipado escalar
- **2020:** PHP 8.0 - JIT, unión de tipos, atributos
- **2024:** PHP 8.4 - Property hooks, asymmetric visibility, DOM API

Uso de PHP en la actualidad

PHP impulsa aproximadamente el 77% de todos los sitios web cuyo lenguaje del lado del servidor es conocido, incluyendo plataformas populares como WordPress, Facebook, Wikipedia y muchos más.

Características de PHP 8.4 (I)

Property Hooks

Permiten crear propiedades computadas que pueden ser entendidas por los IDEs.

```
class Locale {
  public string $languageCode;

  public string $countryCode {
    set (string $code) {
      $this->countryCode = strtoupper($code);
    }
  }
}
```

Visibilidad Asimétrica

Controla independientemente el ámbito de escritura y lectura.

```
class PhpVersion {
  public private(set) string $version = '8.4';
}
```

Nueva API DOM y HTML5

Soporte compatible con estándares para documentos HTML5.

```
$dom = Dom\HTMLDocument::createFromString(
  '<article class="featured">PHP 8.4</article>'
);
$node = $dom->querySelector('article');
var_dump($node->classList->contains("featured"));
```

Atributo #[\Deprecated]

Mecanismo de deprecación disponible para funciones y métodos definidos por el usuario.

PHP 8.4 fue lanzado oficialmente el 21 de noviembre de 2024

Características de PHP 8.4 (II)

Nuevas funciones array_*()

Funciones para facilitar operaciones comunes con arrays.

```
$animal = array_find(
  ['dog', 'cat', 'cow'],
  fn(string $value): bool =>
    str_starts_with($value, 'c')
);
```

API de Objetos para BCMath

Nueva clase `BcMath\Number` para operaciones matemáticas de precisión arbitraria.

```
use BcMath\Number;

$a = new Number('1.234');
$b = new Number('5.678');
$result = $a->add($b);
echo $result; // 6.912
```

Subclases específicas de PDO

Mejoras en el manejo de bases de datos con clases específicas para cada controlador.

```
// Antes
$pdo = new PDO('mysql:host=localhost;dbname=test');

// Ahora (tipado específico)
$mysql = new PDO\MySQL('localhost', 'test');
$sqlite = new PDO\SQLite('database.sqlite');
```

Otras mejoras destacadas

- Instanciación de clases sin paréntesis
- Nuevos métodos `DateTime::createFromTimestamp()`
- Mejoras en el rendimiento del motor Zend
- Optimización de la gestión de memoria
- Mejoras en la JIT (Just-In-Time Compilation)

PHP 8.4 continúa la evolución del lenguaje con un enfoque en la seguridad, rendimiento y usabilidad

Entorno de Desarrollo (I)

Servidor Web

Necesario para procesar y servir archivos PHP:

- **Apache:** El más utilizado y compatible
- **Nginx:** Mayor rendimiento, configuración diferente
- **PHP-FPM:** Gestor de procesos FastCGI para PHP

Base de Datos

Para almacenamiento persistente:

- **MySQL/MariaDB:** La más común con PHP
- **PostgreSQL:** Mayor cumplimiento de estándares
- **SQLite:** Ligera, sin servidor separado

Intérprete PHP

PHP 8.4 requiere:

- ✓ Instalación del intérprete PHP 8.4
- ✓ Configuración en php.ini
- ✓ Extensiones necesarias activadas

Herramientas de Desarrollo

Para un desarrollo eficiente:

- **IDE/Editor:** VS Code, PhpStorm, Sublime Text
- **Depuración:** Xdebug, PHP Debug Bar
- **CLI:** Interfaz de línea de comandos de PHP

Un entorno de desarrollo completo permite aprovechar todas las características de PHP 8.4

Entorno de Desarrollo (II)

Paquetes Todo en Uno

Instalación rápida y sencilla:

- **XAMPP:** Apache + MariaDB + PHP + Perl
- **WAMP:** Windows + Apache + MySQL + PHP
- **MAMP:** Mac + Apache + MySQL + PHP
- **Laragon:** Entorno moderno para Windows

Contenedores

Entornos aislados y reproducibles:

- **Docker:** Contenedores ligeros y portables
- **Docker Compose:** Orquestación de servicios
- **Imágenes oficiales:** php:8.4-apache, php:8.4-fpm

Entornos Virtuales

Desarrollo consistente en equipo:

- **Vagrant:** Máquinas virtuales configurables
- **Laravel Homestead:** Box de Vagrant para PHP
- **Laravel Sail:** Docker para Laravel

Gestión de Dependencias

Administración de librerías:

- **Composer:** Gestor de dependencias para PHP
- **Packagist:** Repositorio principal de paquetes
- **composer.json:** Definición de dependencias

Las soluciones integradas facilitan la configuración del entorno de desarrollo para PHP 8.4

Sintaxis Básica I: Etiquetas PHP

Etiquetas PHP

El código PHP se delimita mediante etiquetas especiales que indican al servidor dónde comienza y termina el código PHP.

```
// Etiquetas estándar (recomendadas)
<?php
  echo "Hola, mundo!";
?>

// Etiquetas cortas (requieren short_open_tag=On)
<?
  echo "Hola, mundo!";
?>

// Etiquetas de impresión (echo)
<?= "Hola, mundo!" ?>
```

Separación de instrucciones

Las instrucciones en PHP se separan con punto y coma (;).

```
<?php
  echo "Primera línea";
  echo "Segunda línea";

  // Múltiples instrucciones en una línea
  echo "Hola"; echo "Mundo";

  // La última instrucción puede omitir el punto y coma
  echo "Final"
?>
```

Sensibilidad a mayúsculas

- **No sensible:** Nombres de funciones, clases y palabras clave
- **Sensible:** Variables, constantes definidas por el usuario

```
<?php
  // Ambas funcionan igual
  ECHO "Hola";
  echo "Hola";

  // Son variables diferentes
  $nombre = "Juan";
  $NOMBRE = "Pedro";
?>
```

Sintaxis Básica II: Comentarios

Tipos de Comentarios

Los comentarios son texto que no se ejecuta y sirve para documentar el código.

```
// Comentario de una línea

# También es un comentario de una línea (estilo shell)

/*
   Comentario de
   múltiples líneas
*/

/**
 * Comentario de documentación (DocBlock)
 * @param string $nombre
 * @return string
 */
```

Buenas prácticas

- Explicar el "por qué", no el "qué"
- Documentar funciones con DocBlocks
- Mantener comentarios actualizados

Estructura básica de un script PHP

```
<!DOCTYPE html>
<html>
<head>
  <title>Mi primera página PHP</title>
</head>
<body>
  <h1>Bienvenido</h1>

  <?php
    // Tu código PHP aquí
    echo "<p>La fecha actual es: " . date("d/m/Y") . "</p>";
  ?>

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

Alternancia entre PHP y HTML

```
<?php if ($mostrar_titulo): ?>
  <h1>Este título es condicional</h1>
<?php endif; ?>

<ul>
<?php foreach ($items as $item): ?>
  <li><?= htmlspecialchars($item) ?></li>
<?php endforeach; ?>
</ul>
```

Sintaxis Básica III: Estructura de Archivos

Estructura de un archivo PHP

Un archivo PHP puede contener texto, HTML, CSS, JavaScript y código PHP.

```
<!DOCTYPE html>
<html>
<head>
  <title>Mi página PHP</title>
  <style>
    body { font-family: Arial; }
  </style>
</head>
<body>
  <h1>Bienvenido</h1>

  <?php
    // Tu código PHP aquí
    $fecha = date("d/m/Y");
    echo "<p>Hoy es: $fecha</p>";
  ?>

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

Organización de archivos

Inclusión de archivos

```
// Incluir archivos externos
include 'encabezado.php';  // Continúa si falla

require 'config.php';    // Fatal error si falla

include_once 'funciones.php';  // Incluye solo una vez

require_once 'conexion.php';  // Requiere solo una vez
```

Estructura de proyecto típica

```
proyecto/
├── index.php           // Punto de entrada
├── config/
│   └── config.php      // Configuración
├── includes/
│   ├── header.php      // Encabezado común
│   └── footer.php      // Pie de página común
├── functions/
│   └── helpers.php     // Funciones auxiliares
└── assets/
    ├── css/            // Estilos
    └── js/             // JavaScript
```

Sintaxis Básica II

Ejemplos Prácticos

Ejemplo 1: Hola Mundo

```php
// Archivo: hola_mundo.php
<!DOCTYPE html>
<html>
<head>
    <title>Mi primera página PHP</title>
</head>
<body>
    <h1>
    <?php
        echo "¡Hola, Mundo!";
    ?>
    </h1>
</body>
</html>
```

Ejemplo 2: Uso de variables

```php
<?php
$nombre = "María";
$edad = 25;

echo "Hola, mi nombre es " . $nombre . " y tengo " . $eda

// Usando interpolación de strings
echo "<br>Hola, mi nombre es {$nombre} y tengo {$edad} a
?>
```

Buenas Prácticas

Estilo de Codificación

- Usar 4 espacios para indentación
- Siempre cerrar las instrucciones con punto y coma
- Usar llaves para bloques de código, incluso si son de una sola línea
- Nombrar variables y funciones en camelCase o snake_case

Seguridad Básica

- Validar y sanitizar todas las entradas de usuario
- Usar consultas preparadas para bases de datos
- No mostrar errores en producción
- Mantener PHP actualizado

Ejemplo 3: Uso de HTML y PHP

```php
<?php
$titulo = "Mi Página Web";
$mostrarMensaje = true;
?>

<!DOCTYPE html>
<html>
<head>
    <title><?= $titulo ?></title>
</head>
<body>
    <?php if ($mostrarMensaje): ?>
        <p>Este mensaje se muestra condicionalmente</p>
    <?php endif; ?>
</body>
</html>
```

Variables y Tipos de Datos I

Declaración de Variables

En PHP, las variables se identifican con un signo de dólar seguido por el nombre de la variable.

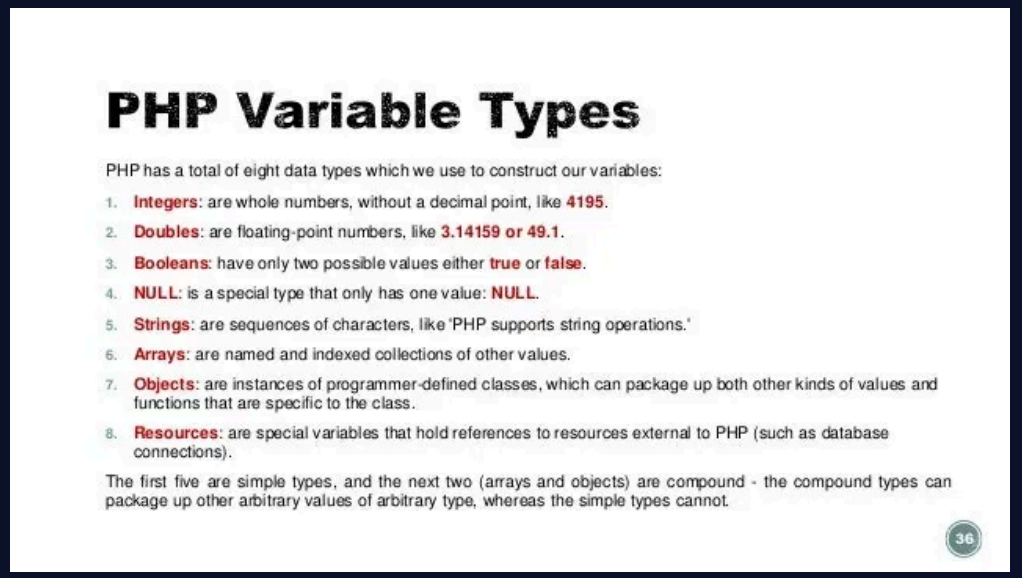
```php
// Declaración de variables
$nombre = "María";
$edad = 25;
$precio = 19.99;
$activo = true;

// Mostrar variables
echo "Nombre: " . $nombre . "<br>";
echo "Edad: " . $edad . " años<br>";
```

Reglas para nombres de variables

- Deben comenzar con una letra o guion bajo
- Solo pueden contener caracteres alfanuméricos y guiones bajos (A-z, 0-9, y _)
- Son sensibles a mayúsculas y minúsculas ($edad y $Edad son diferentes)

Tipos Escalares



Tipos básicos

int: Números enteros como 42, -7, 0

float: Números de punto flotante como 3.14, -2.5

string: Cadenas de texto como "Hola mundo"

bool: Valores booleanos (true o false)

```php
// Ejemplos de tipos escalares
$entero = 42;
$flotante = 3.14159;
$texto = "PHP 8.4";
$booleano = true;
```

Variables y Tipos de Datos II

Tipos Compuestos

Array

Colección ordenada de elementos que pueden ser de cualquier tipo.

```php
// Array indexado
$frutas = ["Manzana", "Banana", "Naranja"];
echo $frutas[0]; // Manzana

// Array asociativo
$persona = [
    "nombre" => "Juan",
    "edad" => 30
];
```

Object

Instancia de una clase con propiedades y métodos.

```php
// Definición de clase
class Persona {
    public $nombre;
    public $edad;

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

Tipos Especiales

NULL y Resource

NULL representa una variable sin valor. Resource son referencias a recursos externos.

```php
$variable = null;
var_dump($variable); // NULL

$archivo = fopen("ejemplo.txt", "r");
// resource(3) of type (stream)
```

Novedades en PHP 8.4

- **Union Types mejorados:** Combinación de varios tipos
- **Typed Properties:** Declaración de tipo para propiedades

```php
// Union Types en PHP 8.4
function procesar(string|int|null $dato): string {
    return "Procesado: " . $dato;
}

// Typed Properties
class Usuario {
    public string $nombre;
    public int $edad;
}
```

Variables y Tipos de Datos III

Conversión de Tipos (Casting)

Conversión explícita

```php
// Conversión de string a int
$numero_texto = "42";
$numero = (int) $numero_texto;
var_dump($numero); // int(42)

// Conversión de float a int
$decimal = 3.14;
$entero = (int) $decimal;
var_dump($entero); // int(3)
```

Operadores de conversión

`(int)` - A entero

`(float)` - A flotante

`(string)` - A cadena

`(bool)` - A booleano

`(array)` - A array

`(object)` - A objeto

Verificación de Tipos

Funciones de verificación

```php
// Verificar tipo con gettype()
$valor = "Hola";
echo gettype($valor); // string

// Funciones específicas
var_dump(is_int(42));        // bool(true)
var_dump(is_string("PHP"));   // bool(true)
```

Conversión automática

```php
// Conversión automática en operaciones
$suma = "10" + 5;  // $suma = 15 (int)

// Comparación con == vs ===
var_dump("42" == 42);  // bool(true)
var_dump("42" === 42); // bool(false)
```

Mejores prácticas: Usar comparaciones estrictas (`===`, `!==`) y validar tipos antes de operaciones críticas

Variables y Tipos de Datos IV

Ámbito de Variables

Variables globales

```php
// Variable global
$nombre = "Juan";

function saludar() {
    global $nombre;  // Acceder a variable global
    echo "Hola, $nombre";
}

// Otra forma de acceder a variables globales
function despedir() {
    echo "Adiós, " . $GLOBALS['nombre'];
}
```

Variables estáticas

```php
function contador() {
    static $count = 0;  // Se inicializa solo una vez
    $count++;
    return $count;
}

echo contador();  // 1
echo contador();  // 2
echo contador();  // 3
```

Variables Predefinidas

Superglobales

Variables disponibles en todos los ámbitos del script.

`$_GET` - Variables pasadas por URL

`$_POST` - Variables enviadas por formulario

`$_REQUEST` - Combinación de GET, POST y COOKIE

`$_SERVER` - Información del servidor y entorno

`$_SESSION` - Variables de sesión

`$_COOKIE` - Cookies HTTP

`$_FILES` - Archivos subidos

`$_ENV` - Variables de entorno

Ejemplo de uso

```php
// Acceder a datos de formulario
if (isset($_POST['usuario'])) {
    $usuario = $_POST['usuario'];
    echo "Bienvenido, $usuario";
}

// Información del servidor
echo "Estás usando: " . $_SERVER['HTTP_USER_AGENT'];
echo "IP del servidor: " . $_SERVER['SERVER_ADDR'];
```

Mejoras en PHP 8.4

PHP 8.4 mejora el manejo de tipos en variables superglobales y ofrece nuevas funciones para validación de datos.

Constantes I

Definición de Constantes

Las constantes son identificadores que representan valores fijos que no cambian durante la ejecución del script.

Usando define()

```php
// Definir constantes con define()
define("PI", 3.14159);
define("NOMBRE_APP", "Mi Aplicación");
define("VERSION", "1.0");

// Usar constantes
echo "Valor de PI: " . PI . "<br>";
echo NOMBRE_APP . " versión " . VERSION;
```

Usando const

```php
// Definir constantes con const
const DATABASE_HOST = "localhost";
const DATABASE_USER = "root";
const DATABASE_PASS = "password";

// Usar constantes
echo "Conectando a: " . DATABASE_HOST;
```

Diferencias entre define() y const

- `define()` se evalúa en tiempo de ejecución
- `const` se evalúa en tiempo de compilación
- `const` no puede usarse en estructuras condicionales

Características de las Constantes

Propiedades

- No llevan el símbolo $ como las variables
- Son globales por defecto
- No se pueden redefinir o eliminar una vez declaradas
- Solo pueden contener valores escalares (boolean, integer, float y string) y arrays

Convenciones de Nombres

Por convención, los nombres de constantes se escriben en mayúsculas con guiones bajos para separar palabras.

```php
// Buenas prácticas para nombrar constantes
define("MAX_USUARIOS", 100);
define("TIEMPO_ESPERA_MAXIMO", 30);
const DIAS_SEMANA = 7;
```

Verificación de Constantes

```php
// Verificar si una constante está definida
if (defined("PI")) {
    echo "La constante PI está definida";
} else {
    echo "La constante PI no está definida";
}

// Obtener todas las constantes definidas
print_r(get_defined_constants());
```

Constantes II

Constantes Mágicas

Constantes mágicas principales

`__LINE__` - Línea actual

`__FILE__` - Ruta del archivo

`__DIR__` - Directorio

`__FUNCTION__` - Función

`__CLASS__` - Clase

`__METHOD__` - Método

Ejemplo de uso

```php
function mostrarInfo() {
    echo "Línea: " . __LINE__ . "<br>";
    echo "Archivo: " . __FILE__ . "<br>";
    echo "Función: " . __FUNCTION__;
}

mostrarInfo();
```

Constantes Predefinidas

Constantes de PHP

`PHP_VERSION` - Versión

`PHP_OS` - Sistema operativo

`PHP_INT_MAX` - Entero máximo

`PHP_EOL` - Fin de línea

```php
echo "PHP: " . PHP_VERSION . "<br>";
echo "OS: " . PHP_OS;
```

Constantes de Clase en PHP 8.4

```php
class Configuracion {
    // Constantes de clase con tipos en PHP 8.4
    public const string ENTORNO = "produccion";
    public const int TIEMPO_SESION = 3600;
    public const array DOMINIOS = [
        "ejemplo.com"
    ];
}
```

Mejores prácticas: Usar constantes para valores que no cambian y definirlas al inicio del script

Constantes III: Enumeraciones

Enumeraciones (PHP 8.1+)

Las enumeraciones (enums) son un tipo especial que representa un conjunto de valores posibles.

Definición básica

```php
// Definir una enumeración
enum Estado {
    case Pendiente;
    case Procesando;
    case Completado;
    case Cancelado;
}

// Usar la enumeración
function actualizarEstado(Estado $estado) {
    echo "Estado actualizado a: " . $estado->name;
}

actualizarEstado(Estado::Procesando);
```

Enumeraciones con Valores

Enums con valores

```php
// Enum con valores asociados
enum StatusCode: int {
    case OK = 200;
    case NOT_FOUND = 404;
    case SERVER_ERROR = 500;

    public function mensaje(): string {
        return match($this) {
            self::OK => "Todo correcto",
            self::NOT_FOUND => "Recurso no encontrado",
            self::SERVER_ERROR => "Error del servidor"
        };
    }
}

echo StatusCode::NOT_FOUND->value;  // 404
echo StatusCode::NOT_FOUND->mensaje();  // "Recurso no encontrado"
```

Mejoras en PHP 8.4

PHP 8.4 añade nuevas funcionalidades a las enumeraciones:

- Mejor integración con tipos de retorno
- Rendimiento optimizado
- Soporte mejorado para serialización

Las enumeraciones en PHP moderno proporcionan una forma segura y expresiva de definir conjuntos de valores constantes

Operadores I

Operadores Aritméticos

Operador	Nombre	Ejemplo	Resultado
+	Suma	$a + $b	Suma de $a y $b
-	Resta	$a - $b	Diferencia de $a y $b
*	Multiplicación	$a * $b	Producto de $a y $b
/	División	$a / $b	Cociente de $a y $b
%	Módulo	$a % $b	Resto de $a / $b
**	Exponenciación	$a ** $b	$a elevado a $b

Ejemplos

```php
$a = 10;
$b = 3;

echo $a + $b;  // 13
echo $a - $b;  // 7
echo $a * $b;  // 30
echo $a / $b;  // 3.3333...
echo $a % $b;  // 1
echo $a ** $b; // 1000
```

Operadores de Asignación

Asignación básica

```php
$a = 5;     // Asigna 5 a $a
$b = $a;    // Asigna el valor de $a a $b
$c = $d = 10; // Asignación múltiple
```

Operadores combinados

Operador	Equivalente	Descripción
+=	$a = $a + $b	Suma y asignación
-=	$a = $a - $b	Resta y asignación
*=	$a = $a * $b	Multiplicación y asignación
/=	$a = $a / $b	División y asignación
%=	$a = $a % $b	Módulo y asignación

Ejemplos

```php
$x = 10;

$x += 5;  // $x = 15
$x -= 3;  // $x = 12
$x *= 2;  // $x = 24
$x /= 4;  // $x = 6
$x %= 4;  // $x = 2
```

Operadores II

Operadores de Comparación

Operador	Nombre	Ejemplo	Resultado
==	Igual	$a == $b	TRUE si $a es igual a $b
===	Idéntico	$a === $b	TRUE si $a es igual a $b y del mismo tipo
!=	Diferente	$a != $b	TRUE si $a no es igual a $b
!==	No idéntico	$a !== $b	TRUE si $a no es igual a $b o no son del mismo tipo
<	Menor que	$a < $b	TRUE si $a es menor que $b
>	Mayor que	$a > $b	TRUE si $a es mayor que $b

Ejemplos

```php
$a = 5;
$b = "5";
$c = 10;

var_dump($a == $b);   // bool(true) - Igual en valor
var_dump($a === $b);  // bool(false) - Diferente tipo
var_dump($a != $c);   // bool(true) - Diferente valor
var_dump($a < $c);    // bool(true) - $a es menor que $c
```

Operadores Lógicos

Operador	Nombre	Ejemplo	Resultado
&&	Y (AND)	$a && $b	TRUE si $a y $b son TRUE
||	O (OR)	$a || $b	TRUE si $a o $b es TRUE
!	NO (NOT)	!$a	TRUE si $a es FALSE
and	Y (AND)	$a and $b	Como &&, pero menor precedencia
or	O (OR)	$a or $b	Como ||, pero menor precedencia
xor	XOR	$a xor $b	TRUE si $a o $b es TRUE, pero no ambos

Ejemplos

```php
$a = true;
$b = false;

var_dump($a && $b);  // bool(false)
var_dump($a || $b);  // bool(true)
var_dump(!$a);       // bool(false)
var_dump($a xor $b); // bool(true)

// Ejemplo de precedencia
$result = true && false; // $result = false
$result = true and false; // $result = true
```

Operadores III

Operadores Especiales

Operador Ternario

```php
// Sintaxis: condición ? valor_si_verdadero : valor_si_falso

$edad = 20;
$mensaje = ($edad >= 18) ? "Mayor de edad" : "Menor";
echo $mensaje; // "Mayor de edad"
```

Operador Null Coalescing (??)

```php
// Antes de PHP 7
$username = isset($_GET['user']) ? $_GET['user'] : 'nadie';

// Con el operador ??
$username = $_GET['user'] ?? 'nadie';

// Encadenamiento
$username = $_GET['user'] ?? $_POST['user'] ?? 'nadie';
```

Novedades en PHP 8.4

Null Coalescing Assignment (??=)

```php
// Antes de PHP 7.4
if (!isset($array['key'])) {
    $array['key'] = "valor";
}

// Con el operador ??=
$array['key'] ??= "valor";
```

Match Expression (PHP 8.0+)

```php
$status = 200;

// Usando match
$mensaje = match($status) {
    200, 201 => 'Éxito',
    400 => 'Solicitud incorrecta',
    404 => 'No encontrado',
    default => 'Error desconocido'
};
```

Match expression en PHP 8.4 ofrece una sintaxis más concisa y segura que switch, con comparaciones estrictas por defecto

Entrada y Salida I

Funciones de Salida

echo

```php
// Formas de usar echo
echo "Hola Mundo";
echo "Hola", " ", "Mundo"; // Múltiples argumentos
echo "El resultado es: " . 5 * 3;

// Sintaxis corta
?>Texto HTML con  incrustada
```

print y printf

```php
// Uso de print
print "Hola Mundo"; // Devuelve 1

// printf - Imprime la cadena formateada
printf("Nombre: %s, Edad: %d", "Juan", 25);

// sprintf - Devuelve la cadena formateada
$texto = sprintf("Precio: %.2f €", 19.95);
```

Funciones de Depuración

var_dump

```php
$array = ["manzana", "banana", "naranja"];
var_dump($array);
/* Salida:
array(3) {
  [0]=> string(7) "manzana"
  [1]=> string(6) "banana"
  [2]=> string(7) "naranja"
}
*/
```

print_r

```php
$array = ["manzana", "banana", "naranja"];
print_r($array);
/* Salida:
Array
(
    [0] => manzana
    [1] => banana
    [2] => naranja
)
*/
```

Consejo: Usar var_dump() durante el desarrollo para depurar y echo/print para la salida final al usuario

Entrada y Salida II

Variables Superglobales

$_GET y $_POST

```php
// Acceder a datos enviados por GET (URL)
// ejemplo.php?nombre=Juan&edad=25
$nombre = $_GET['nombre'] ?? ''; // "Juan"
$edad = $_GET['edad'] ?? 0;   // 25

// Acceder a datos enviados por POST
$usuario = $_POST['usuario'] ?? '';
```

$_SERVER

```php
// $_SERVER contiene información del servidor
echo "Método: " . $_SERVER['REQUEST_METHOD'];
echo "URL: " . $_SERVER['REQUEST_URI'];
```

Validación de Entrada

Sanitización Básica

```php
// Validar si un campo existe y no está vacío
if (isset($_POST['email']) && !empty($_POST['email'])) {
    // Sanitizar email
    $email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
}
```

Filtros en PHP

```php
// Filtrar y validar en una sola operación
$opciones = [
    'options' => ['min_range' => 1, 'max_range' => 100]
];

$id = filter_input(
    INPUT_GET,
    'id',
    FILTER_VALIDATE_INT,
    $opciones
);
```

Siempre valida y sanitiza la entrada del usuario para prevenir vulnerabilidades como XSS e inyección SQL

Estructuras de Control I

Estructura if

if simple

```php
if ($edad >= 18) {
    echo "Eres mayor de edad";
}
```

if-else

```php
if ($puntuacion >= 60) {
    echo "Aprobado";
} else {
    echo "Reprobado";
}
```

if-elseif-else

```php
$nota = 85;

if ($nota >= 90) {
    echo "Sobresaliente";
} elseif ($nota >= 80) {
    echo "Notable";
} elseif ($nota >= 70) {
    echo "Bien";
} else {
    echo "Insuficiente";
}
```

Estructura switch

switch básico

```php
$dia = 3;

switch ($dia) {
    case 1:
        echo "Lunes";
        break;
    case 2:
        echo "Martes";
        break;
    case 3:
        echo "Miércoles";
        break;
    default:
        echo "Otro día";
}
```

Casos agrupados

```php
$dia = "Sábado";

switch ($dia) {
    case "Lunes":
    case "Martes":
    case "Miércoles":
    case "Jueves":
    case "Viernes":
        echo "Día laborable";
        break;
    default:
        echo "Fin de semana";
}
```

Usa switch cuando compares una variable con múltiples valores y if-elseif para condiciones más complejas

Estructuras de Control II

Match Expression (PHP 8.0+)

Sintaxis básica

```php
$codigo = 404;

$mensaje = match($codigo) {
    200 => 'OK',
    404 => 'No encontrado',
    500 => 'Error del servidor',
    default => 'Código desconocido'
};

echo $mensaje; // 'No encontrado'
```

Diferencias con switch

- Match usa comparación estricta (===)
- Match es una expresión que devuelve un valor
- Match no necesita break
- Match lanza error si no hay coincidencia (sin default)

Múltiples valores

```php
$dia = 'Sábado';

$tipo = match($dia) {
    'Lunes', 'Martes', 'Miércoles',
    'Jueves', 'Viernes' => 'Laborable',
    'Sábado', 'Domingo' => 'Fin de semana',
    default => 'Día inválido'
};

echo $tipo; // 'Fin de semana'
```

Expresiones en Match

Evaluación de condiciones

```php
$edad = 25;

$grupo = match(true) {
    $edad < 13 => 'Niño',
    $edad < 18 => 'Adolescente',
    $edad < 65 => 'Adulto',
    default => 'Adulto mayor'
};

echo $grupo; // 'Adulto'
```

Uso con funciones

```php
function validar($valor) {
    return match(true) {
        is_numeric($valor) => "Es un número",
        is_string($valor) && strlen($valor) > 0 => "Es texto",
        is_array($valor) => "Es un array",
        default => "Tipo desconocido"
    };
}

echo validar("Hola"); // "Es texto"
echo validar(42);     // "Es un número"
```

Mejoras en PHP 8.4

PHP 8.4 mejora el rendimiento de match y optimiza su uso en expresiones complejas.

```php
// Ejemplo de match con expresiones en PHP 8.4
$resultado = match($valor ?? '') {
    '' => 'Valor vacío',
    default => fn($x) => "Valor: {$x}"
}($valor);
```

Match es una alternativa moderna y más segura a switch, con comparaciones estrictas y sintaxis más concisa

Bucles e Iteración I

Bucle while

Sintaxis básica

```php
$i = 1;
while ($i <= 5) {
    echo $i . " ";
    $i++;
}
// Salida: 1 2 3 4 5
```

Bucle do-while

```php
// Ejecuta al menos una vez
$i = 1;
do {
    echo $i . " ";
    $i++;
} while ($i <= 5);

// Con condición falsa inicialmente
$i = 10;
do {
    echo $i;  // Se ejecuta una vez
} while ($i < 10);
```

Bucle for

Sintaxis básica

```php
for ($i = 1; $i <= 5; $i++) {
    echo $i . " ";
}
// Salida: 1 2 3 4 5
```

Múltiples expresiones

```php
// Inicialización múltiple
for ($i = 0, $j = 10; $i < 10; $i++, $j--) {
    echo "i: " . $i . " j: " . $j . "
";
}
```

Control de bucles

```php
// break - Sale del bucle
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        break;
    }
    echo $i;  // Salida: 1234
}

// continue - Salta a la siguiente iteración
for ($i = 1; $i <= 5; $i++) {
    if ($i == 3) {
        continue;
    }
    echo $i;  // Salida: 1245
}
```

Usa for cuando conoces el número de iteraciones y while/do-while cuando la condición de salida no depende de un contador

Bucles e Iteración II

Bucle foreach

Sintaxis básica

```php
// Array indexado
$frutas = ["manzana", "banana", "naranja"];

foreach ($frutas as $fruta) {
    echo $fruta . "
";
}
// Salida: manzana, banana, naranja
```

Con clave y valor

```php
// Array asociativo
$persona = [
    "nombre" => "Juan",
    "edad" => 30,
    "ciudad" => "Madrid"
];

foreach ($persona as $clave => $valor) {
    echo $clave . ": " . $valor . "
";
}
// Salida: nombre: Juan, edad: 30, ciudad: Madrid
```

Modificando valores

```php
// Modificar valores por referencia
$numeros = [1, 2, 3, 4, 5];

foreach ($numeros as &$numero) {
    $numero *= 2;  // Duplica cada número
}
unset($numero);  // Buena práctica: liberar la referencia

print_r($numeros);  // Array([0]=>2, [1]=>4, [2]=>6, [3]=>8, [4]=>10)
```

Iteración Avanzada

Bucles anidados

```php
// Tabla de multiplicar
for ($i = 1; $i <= 3; $i++) {
    for ($j = 1; $j <= 3; $j++) {
        echo $i . " x " . $j . " = " . ($i * $j) . "
";
    }
    echo "
";
}
```

Iteración de objetos

```php
class Persona {
    public $nombre = "Ana";
    public $edad = 25;
    private $dni = "12345678X";
}

$p = new Persona();

// Itera solo propiedades públicas
foreach ($p as $clave => $valor) {
    echo $clave . ": " . $valor . "
";
}
// Salida: nombre: Ana, edad: 25
```

Iteradores en PHP 8.4

```php
// Uso de iteradores
$directorio = new DirectoryIterator("./archivos");

foreach ($directorio as $archivo) {
    if (!$archivo->isDot()) {
        echo $archivo->getFilename() . "
";
    }
}
```

foreach es la forma más elegante de recorrer arrays y objetos en PHP, especialmente para estructuras de datos complejas

Funciones Básicas I

Definición de Funciones

Sintaxis básica

```php
// Función sin parámetros
function saludar() {
    echo "¡Hola, mundo!";
}

// Llamada a la función
saludar();  // Salida: ¡Hola, mundo!
```

Funciones con parámetros

```php
// Función con parámetros
function saludarPersona($nombre) {
    echo "¡Hola, " . $nombre . "!";
}

saludarPersona("María");  // Salida: ¡Hola, María!

// Función con múltiples parámetros
function sumar($a, $b) {
    return $a + $b;
}

$resultado = sumar(5, 3);  // $resultado = 8
```

Valores por defecto

```php
// Parámetros con valores predeterminados
function configurar($color = "azul", $tamaño = "mediano") {
    echo "Color: " . $color . ", Tamaño: " . $tamaño;
}

configurar();                // Color: azul, Tamaño: mediano
configurar("rojo");          // Color: rojo, Tamaño: mediano
configurar("verde", "grande");  // Color: verde, Tamaño: grande
```

Retorno de Valores

Retorno simple

```php
// Función con retorno
function cuadrado($numero) {
    return $numero * $numero;
}

$resultado = cuadrado(4);  // $resultado = 16

// Función sin retorno explícito
function mostrarMensaje($mensaje) {
    echo $mensaje;
}

$valor = mostrarMensaje("Hola");  // $valor = null
```

Retorno múltiple

```php
// Retornar múltiples valores con array
function obtenerDimensiones() {
    return [10, 20, 30];  // [ancho, alto, profundidad]
}

$dimensiones = obtenerDimensiones();
echo $dimensiones[0];  // 10

// Usando list() para asignar valores
list($ancho, $alto, $profundidad) = obtenerDimensiones();
echo $ancho;  // 10
```

Declaración de tipos

```php
// Tipos de parámetros y retorno
function multiplicar(int $a, int $b): int {
    return $a * $b;
}

// Tipos unión (PHP 8.0+)
function procesar(string|int $dato): string {
    return "Procesado: " . $dato;
}
```

Las funciones son bloques de código reutilizables que mejoran la organización y mantenimiento del código

Funciones Básicas II

Ámbito de Variables

Variables locales

```php
// Variables dentro de funciones
function prueba() {
    $local = "Variable local";
    echo $local;
}

prueba();  // Salida: Variable local
// echo $local;  // Error: $local no existe fuera de la función
```

Variables globales

```php
$global = "Variable global";

function usarGlobal() {
    global $global;  // Acceder a variable global
    echo $global;

    // Alternativa usando $GLOBALS
    echo $GLOBALS['global'];
}

usarGlobal();  // Salida: Variable global
```

Variables estáticas

```php
function contador() {
    static $count = 0;  // Se inicializa solo una vez
    $count++;
    echo $count . " ";
}

contador();  // Salida: 1
contador();  // Salida: 2
contador();  // Salida: 3
```

Funciones Avanzadas

Funciones anónimas

```php
// Función anónima asignada a variable
$saludo = function($nombre) {
    return "Hola, " . $nombre;
};

echo $saludo("Juan");  // Salida: Hola, Juan

// Como argumento de otra función
$numeros = [1, 2, 3, 4];
$cuadrados = array_map(function($n) {
    return $n * $n;
}, $numeros);  // [1, 4, 9, 16]
```

Funciones flecha (PHP 7.4+)

```php
// Sintaxis más concisa para funciones simples
$cuadrados = array_map(fn($n) => $n * $n, $numeros);

// Captura automática de variables del ámbito
$factor = 2;
$multiplicados = array_map(
    fn($n) => $n * $factor,  // Usa $factor sin 'use'
    $numeros
);
```

Novedades en PHP 8.4

```php
// Atributo #[\Deprecated] para funciones
#[\Deprecated(reason: "Usar nuevaFuncion() en su lugar")]
function funcionAntigua() {
    return "Método antiguo";
}

// Mejoras en funciones anónimas
$filtro = fn(int|float $valor): bool =>
    $valor > 0 && $valor < 100;
```

Las funciones anónimas y de flecha son especialmente útiles para operaciones de procesamiento de datos y callbacks

Arrays Modernos I

Arrays Indexados

```php
// Definición de array indexado
$frutas = ["manzana", "banana", "naranja"];

// Sintaxis antigua
$frutas = array("manzana", "banana", "naranja");

// Acceso por índice
echo $frutas[0];  // manzana
echo $frutas[1];  // banana
```

Arrays Asociativos

```php
// Definición de array asociativo
$persona = [
    "nombre" => "Juan",
    "edad" => 30,
    "ciudad" => "Madrid"
];

// Acceso por clave
echo $persona["nombre"];  // Juan
echo $persona["edad"];    // 30
```

Los arrays en PHP son extremadamente versátiles y pueden usarse como arrays, listas, tablas hash, diccionarios y colecciones

Arrays Multidimensionales

Arrays de Arrays

```php
// Array multidimensional (matriz)
$matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
];

// Acceso a elementos
echo $matriz[0][0];  // 1
echo $matriz[1][1];  // 5
echo $matriz[2][2];  // 9
```

Arrays Asociativos Anidados

```php
// Array asociativo con arrays anidados
$usuarios = [
    "usuario1" => [
        "nombre" => "Ana",
        "roles" => ["admin", "editor"]
    ],
    "usuario2" => [
        "nombre" => "Luis",
        "roles" => ["usuario"]
    ]
];

// Acceso a datos anidados
echo $usuarios["usuario1"]["nombre"];  // Ana
echo $usuarios["usuario2"]["roles"][0];  // usuario
```

Los arrays multidimensionales son ideales para representar datos jerárquicos o estructuras complejas

Arrays Modernos II: Operaciones Básicas

Añadir y Eliminar Elementos

Añadir elementos

```php
// Añadir al final
$frutas = ["manzana", "banana"];
$frutas[] = "naranja";  // [manzana, banana, naranja]

// Añadir con clave específica
$persona = ["nombre" => "Juan"];
$persona["edad"] = 30;  // [nombre=>Juan, edad=>30]
```

Eliminar elementos

```php
// Eliminar un elemento
$frutas = ["manzana", "banana", "naranja"];
unset($frutas[1]);  // [0=>manzana, 2=>naranja]

// Eliminar por clave
$persona = ["nombre" => "Juan", "edad" => 30];
unset($persona["edad"]);  // [nombre=>Juan]
```

Recorrer Arrays

Bucle foreach

```php
// Recorrer valores
$frutas = ["manzana", "banana", "naranja"];
foreach ($frutas as $fruta) {
    echo $fruta . "
";
}

// Recorrer claves y valores
$persona = ["nombre" => "Juan", "edad" => 30];
foreach ($persona as $clave => $valor) {
    echo $clave . ": " . $valor . "
";
}
```

Funciones de iteración

```php
// Obtener claves
$claves = array_keys($persona);  // [nombre, edad]

// Obtener valores
$valores = array_values($persona);  // [Juan, 30]

// Verificar existencia de clave
if (array_key_exists("nombre", $persona)) {
    echo "La clave 'nombre' existe";
}
```

PHP proporciona numerosas funciones para manipular arrays de forma eficiente

Arrays Modernos III: Funciones Útiles

Búsqueda y Filtrado

Buscar en arrays

```php
// Buscar valor
$frutas = ["manzana", "banana", "naranja"];
if (in_array("banana", $frutas)) {
    echo "Banana encontrada";
}

// Buscar clave
$posicion = array_search("banana", $frutas);
echo $posicion;  // 1
```

Filtrar arrays

```php
// Filtrar con array_filter
$numeros = [1, 2, 3, 4, 5, 6];
$pares = array_filter($numeros, function($n) {
    return $n % 2 == 0;
});
// $pares = [2, 4, 6] (mantiene las claves originales)
```

Transformación

Mapear arrays

```php
// Transformar con array_map
$numeros = [1, 2, 3, 4, 5];
$cuadrados = array_map(function($n) {
    return $n * $n;
}, $numeros);
// $cuadrados = [1, 4, 9, 16, 25]
```

Reducir arrays

```php
// Reducir con array_reduce
$numeros = [1, 2, 3, 4, 5];
$suma = array_reduce($numeros, function($acumulador, $n) {
    return $acumulador + $n;
}, 0);
echo $suma;  // 15
```

Las funciones de orden superior como array_filter, array_map y array_reduce permiten manipular arrays de forma funcional

Arrays Modernos IV: Mejoras en PHP 8.4

Nuevas Características

Spread operator en arrays

```php
// Combinar arrays con spread operator
$frutas1 = ["manzana", "banana"];
$frutas2 = ["naranja", "uva"];
$todasFrutas = [...$frutas1, ...$frutas2];
// $todasFrutas = ["manzana", "banana", "naranja", "uva"]
```

Desestructuración de arrays

```php
// Desestructuración básica
$datos = ["Juan", 30, "Madrid"];
[$nombre, $edad, $ciudad] = $datos;

// Desestructuración con claves
$persona = ["nombre" => "Ana", "edad" => 25];
["nombre" => $n, "edad" => $e] = $persona;
```

Rendimiento y Optimización

Mejoras de rendimiento

- Optimización de funciones de array internas
- Mejor manejo de memoria para arrays grandes
- Operaciones más rápidas para arrays asociativos

Nuevas funciones de array

```php
// array_is_list(): Verifica si un array es una lista
$lista = ["a", "b", "c"];
$noLista = [1 => "a", 0 => "b"];

var_dump(array_is_list($lista));    // true
var_dump(array_is_list($noLista));  // false
```

Arrow functions con arrays

```php
// Uso de arrow functions (PHP 7.4+)
$numeros = [1, 2, 3, 4, 5];
$dobles = array_map(fn($n) => $n * 2, $numeros);
// $dobles = [2, 4, 6, 8, 10]
```

PHP 8.4 continúa mejorando el manejo de arrays con nuevas funcionalidades y optimizaciones de rendimiento

Strings y Manipulación I

Declaración de Strings

Comillas simples y dobles

```php
// Comillas simples - literal
$simple = 'Hola mundo';

// Comillas dobles - interpreta variables
$nombre = "Juan";
$saludo = "Hola $nombre";  // "Hola Juan"

// Escapar caracteres
$texto1 = 'No se interpreta $nombre';
$texto2 = "Línea 1\nLínea 2";  // Salto de línea
$texto3 = "Comillas \"dobles\" dentro";
```

Sintaxis heredoc y nowdoc

```php
// Heredoc - interpreta variables (como "")
$edad = 30;
$texto = <<// Nowdoc - literal (como '')
$texto = <<<'EOT'
Hola $nombre,
Tienes $edad años.
EOT;
```

Concatenación

```php
// Operador de concatenación (.)
$nombre = "Juan";
$apellido = "Pérez";
$nombreCompleto = $nombre . " " . $apellido;  // "Juan Pérez"

// Operador de asignación concatenada (.=)
$texto = "Hola";
$texto .= " mundo";  // "Hola mundo"
```

Funciones Básicas

Longitud y posición

```php
// Longitud de un string
$texto = "Hola mundo";
echo strlen($texto);  // 10

// Encontrar posición
$pos = strpos($texto, "mundo");  // 5
$pos2 = strpos($texto, "xyz");   // false

// Búsqueda segura
if (strpos($texto, "mundo") !== false) {
    echo "Encontrado";
}
```

Extracción de substrings

```php
// Extraer parte de un string
$texto = "Hola mundo";
$sub1 = substr($texto, 0, 4);  // "Hola"
$sub2 = substr($texto, 5);     // "mundo"
$sub3 = substr($texto, -5);    // "mundo" (desde el final)

// Extraer hasta cierto carácter
$email = "usuario@ejemplo.com";
$usuario = substr($email, 0, strpos($email, "@"));
// "usuario"
```

Transformación

```php
// Cambiar mayúsculas/minúsculas
$texto = "Hola Mundo";
echo strtolower($texto);  // "hola mundo"
echo strtoupper($texto);  // "HOLA MUNDO"
echo ucfirst("hola");     // "Hola"
echo ucwords("hola mundo"); // "Hola Mundo"
```

PHP ofrece más de 100 funciones para manipular strings, lo que lo hace especialmente potente para el procesamiento de texto

Strings y Manipulación II

Búsqueda y Reemplazo

Reemplazo básico

```php
// Reemplazar texto
$texto = "Hola mundo";
$nuevo = str_replace("mundo", "PHP", $texto);
echo $nuevo;  // "Hola PHP"

// Reemplazar múltiples valores
$texto = "manzana, banana, cereza";
$buscar = ["manzana", "cereza"];
$reemplazar = ["pera", "uva"];
$nuevo = str_replace($buscar, $reemplazar, $texto);
// "pera, banana, uva"
```

Expresiones regulares

```php
// Buscar con patrón
$texto = "Contacto: info@ejemplo.com";
$patron = "/[\w\.-]+@[\w\.-]+\.\w+/";
preg_match($patron, $texto, $coincidencias);
echo $coincidencias[0];  // "info@ejemplo.com"

// Reemplazar con patrón
$texto = "Fecha: 2023-10-15";
$patron = "/(\d{4})-(\d{2})-(\d{2})/";
$reemplazo = "$3/$2/$1";
$nuevo = preg_replace($patron, $reemplazo, $texto);
// "Fecha: 15/10/2023"
```

Formateo y División

Formateo de strings

```php
// sprintf - formato similar a C
$nombre = "Juan";
$edad = 30;
$altura = 1.75;

$texto = sprintf(
    "Nombre: %s, Edad: %d, Altura: %.2f m",
    $nombre, $edad, $altura
);
// "Nombre: Juan, Edad: 30, Altura: 1.75 m"
```

División de strings

```php
// Dividir por delimitador
$csv = "manzana,banana,cereza";
$frutas = explode(",", $csv);
print_r($frutas);  // Array([0]=>manzana, [1]=>banana, [2]=>cereza)

// Unir array en string
$array = ["PHP", "HTML", "CSS"];
$lista = implode(", ", $array);
echo $lista;  // "PHP, HTML, CSS"
```

Novedades en PHP 8.4

```php
// str_pad_left y str_pad_right
$num = "42";
$padded = str_pad_left($num, 5, "0");
echo $padded;  // "00042"

// str_contains, str_starts_with, str_ends_with (PHP 8.0+)
$texto = "Hola mundo PHP";
var_dump(str_contains($texto, "mundo"));    // true
var_dump(str_starts_with($texto, "Hola"));  // true
var_dump(str_ends_with($texto, "PHP"));    // true
```

PHP 8.4 mejora el rendimiento de las funciones de string y añade nuevas funciones para simplificar operaciones comunes

Manejo de Errores I

Tipos de Errores

Niveles de error

- **E_ERROR:** Errores fatales que detienen la ejecución
- **E_WARNING:** Advertencias que no detienen la ejecución
- **E_NOTICE:** Avisos sobre posibles problemas
- **E_PARSE:** Errores de sintaxis en el código
- **E_DEPRECATED:** Funcionalidades obsoletas

Ejemplos comunes

```php
// E_WARNING: División por cero
$resultado = 10 / 0;

// E_NOTICE: Variable indefinida
echo $variable_no_definida;

// E_ERROR: Llamada a función inexistente
funcion_que_no_existe();
```

Configuración de Errores

Funciones de configuración

```php
// Establecer nivel de reporte de errores
error_reporting(E_ALL);  // Todos los errores
error_reporting(E_ALL & ~E_NOTICE);  // Sin notices

// Mostrar errores (desarrollo)
ini_set('display_errors', 1);

// Ocultar errores (producción)
ini_set('display_errors', 0);
```

Registro de errores

```php
// Activar registro de errores
ini_set('log_errors', 1);

// Establecer archivo de registro
ini_set('error_log', '/ruta/a/error.log');

// Registrar mensaje personalizado
error_log("Error en la base de datos");
```

Un buen manejo de errores es fundamental para desarrollar aplicaciones PHP robustas y seguras

Manejo de Errores II

Try-Catch-Finally

Estructura básica

```php
try {
    // Código que puede generar una excepción
    $resultado = 10 / 0;
} catch (Exception $e) {
    // Manejo de la excepción
    echo "Error: " . $e->getMessage();
} finally {
    // Código que siempre se ejecuta
    echo "Operación finalizada";
}
```

Múltiples catch

```php
try {
    // Código que puede generar excepciones
    if (!file_exists("archivo.txt")) {
        throw new Exception("Archivo no encontrado");
    }
} catch (InvalidArgumentException $e) {
    // Manejo específico
    echo "Argumento inválido: " . $e->getMessage();
} catch (Exception $e) {
    // Manejo general
    echo "Error general: " . $e->getMessage();
}
```

Excepciones Personalizadas

Crear excepciones propias

```php
// Definir excepción personalizada
class DatabaseException extends Exception {
    public function __construct($mensaje, $codigo = 0) {
        parent::__construct($mensaje, $codigo);
    }

    public function getErrorInfo() {
        return "Error de base de datos: " .
               $this->getMessage();
    }
}
```

Usar excepciones personalizadas

```php
try {
    // Simular error de conexión
    throw new DatabaseException(
        "No se pudo conectar al servidor"
    );
} catch (DatabaseException $e) {
    echo $e->getErrorInfo();
    // Registrar el error
    error_log($e->getMessage());
}
```

Las excepciones permiten manejar errores de forma estructurada y recuperarse de situaciones problemáticas

Manejo de Errores III: Novedades en PHP 8.4

Mejoras en Excepciones

Nuevas clases de excepción

- **TypeError:** Para errores de tipo
- **ValueError:** Para valores inválidos
- **UnhandledMatchError:** Para match sin coincidencias
- **FiberError:** Para errores en fibras

Mejoras en el manejo

```php
// Captura de excepciones con unión de tipos
try {
    // Código que puede generar excepciones
} catch (TypeError|ValueError $e) {
    // Manejo de errores de tipo o valor
    echo "Error de tipo o valor: " . $e->getMessage();
}
```

Buenas Prácticas

Desarrollo vs. Producción

```php
// Configuración para desarrollo
if ($entorno === 'desarrollo') {
    ini_set('display_errors', 1);
    error_reporting(E_ALL);
} else {
    // Configuración para producción
    ini_set('display_errors', 0);
    error_reporting(E_ALL & ~E_DEPRECATED & ~E_STRICT);
    ini_set('log_errors', 1);
}
```

Recomendaciones generales

- Usar try-catch para código que puede fallar
- Registrar errores en archivos de log
- Mostrar mensajes amigables al usuario
- Crear excepciones específicas para tu aplicación
- Validar datos antes de procesarlos
- Usar declaraciones de tipo para prevenir errores

PHP 8.4 mejora el sistema de manejo de errores, facilitando la creación de aplicaciones más robustas y seguras

Ejercicios Prácticos I

Ejercicio 1: Calculadora

Descripción

Crear una calculadora simple que realice operaciones básicas (suma, resta, multiplicación, división) utilizando funciones.

```php
// calculadora.php
function calcular($num1, $num2, $operacion) {
    return match($operacion) {
        'suma' => $num1 + $num2,
        'resta' => $num1 - $num2,
        'multiplicacion' => $num1 * $num2,
        'division' => $num2 != 0 ? $num1 / $num2 : "Error: División por cero",
        default => "Operación no válida"
    };
}

// Ejemplo de uso
$resultado = calcular(10, 5, 'suma');
echo "Resultado: $resultado";  // Resultado: 15
```

Reto adicional

Ampliar la calculadora para incluir operaciones como potencia, raíz cuadrada y módulo. Implementar manejo de errores con excepciones.

Ejercicio 2: Validador de Formularios

Descripción

Crear un validador de formularios que compruebe campos como email, nombre, teléfono y contraseña.

```php
// validador.php
function validarEmail($email) {
    return filter_var($email, FILTER_VALIDATE_EMAIL) !== false;
}

function validarNombre($nombre) {
    return strlen($nombre) >= 2 && preg_match('/^[a-zA-Z\s]+$/', $nombre);
}

function validarTelefono($telefono) {
    return preg_match('/^[0-9]{9}$/', $telefono);
}

function validarClave($clave) {
    // Al menos 8 caracteres, una mayúscula, una minúscula y un número
    return strlen($clave) >= 8 &&
           preg_match('/[A-Z]/', $clave) &&
           preg_match('/[a-z]/', $clave) &&
           preg_match('/[0-9]/', $clave);
}
```

Reto adicional

Crear una clase Validador que encapsule todas las funciones y devuelva mensajes de error específicos para cada campo.

Estos ejercicios te ayudarán a practicar los conceptos básicos de PHP 8.4 y a desarrollar habilidades prácticas de programación

Ejercicios Prácticos II

Ejercicio 3: Manipulación de Arrays

Descripción

Crear funciones para procesar una lista de productos con operaciones como filtrado, ordenación y transformación.

```php
// productos.php
$productos = [
    ["id" => 1, "nombre" => "Laptop", "precio" => 899.99, "stock" => 10],
    ["id" => 2, "nombre" => "Teléfono", "precio" => 499.50, "stock" => 15],
    ["id" => 3, "nombre" => "Tablet", "precio" => 349.99, "stock" => 5]
];

// Filtrar productos con precio > 400
$caros = array_filter($productos, fn($p) => $p["precio"] > 400);

// Ordenar por precio (ascendente)
usort($productos, fn($a, $b) => $a["precio"] <=> $b["precio"]);

// Calcular valor total del inventario
$valorTotal = array_reduce($productos, fn($total, $p) =>
    $total + ($p["precio"] * $p["stock"]), 0
);
```

Reto adicional

Implementar una función de búsqueda que permita filtrar productos por nombre usando coincidencias parciales.

Ejercicio 4: Procesador de Texto

Descripción

Crear un procesador de texto que analice un párrafo y extraiga estadísticas como conteo de palabras, frecuencia de palabras y longitud promedio.

```php
// procesador_texto.php
function analizarTexto($texto) {
    // Limpiar y dividir el texto en palabras
    $texto = strtolower($texto);
    $texto = preg_replace('/[^\w\s]/', '', $texto);
    $palabras = preg_split('/\s+/', $texto, -1, PREG_SPLIT_NO_EMPTY);

    // Contar palabras
    $totalPalabras = count($palabras);

    // Frecuencia de palabras
    $frecuencia = array_count_values($palabras);
    arsort($frecuencia);  // Ordenar por frecuencia

    // Longitud promedio
    $longitudTotal = array_reduce($palabras, fn($total, $p) =>
        $total + strlen($p), 0
    );
    $longitudPromedio = $totalPalabras > 0 ?
        $longitudTotal / $totalPalabras : 0;

    return [
        'total_palabras' => $totalPalabras,
        'frecuencia' => $frecuencia,
        'longitud_promedio' => $longitudPromedio
    ];
}
```

Reto adicional

Ampliar el procesador para identificar n-gramas (secuencias de n palabras) y detectar frases comunes en el texto.

Estos ejercicios te permitirán aplicar los conocimientos de arrays y strings en situaciones prácticas de programación

Recursos y Referencias

Documentación Oficial

Manual de PHP

Documentación completa y oficial de PHP con ejemplos y explicaciones detalladas.
https://www.php.net/manual/es/

Notas de la versión PHP 8.4

Información oficial sobre las nuevas características, mejoras y cambios en PHP 8.4.
https://www.php.net/releases/8.4/

Funciones de PHP por categoría

Listado organizado de todas las funciones disponibles en PHP agrupadas por categoría.
https://www.php.net/manual/es/funcref.php

Tutoriales y Cursos

PHP: The Right Way

Guía de mejores prácticas para PHP moderno, actualizada regularmente.
https://phptherightway.com/

Laracasts

Videotutoriales de alta calidad sobre PHP y frameworks relacionados.
https://laracasts.com/

Herramientas y Entornos

XAMPP

Entorno de desarrollo local que incluye PHP, MySQL, Apache y más.
https://www.apachefriends.org/

Docker PHP

Imágenes oficiales de Docker para PHP, ideales para desarrollo y producción.
https://hub.docker.com/_/php

Visual Studio Code + PHP

Editor de código con extensiones para PHP como PHP Intelephense y PHP Debug.
https://code.visualstudio.com/

Comunidad y Soporte

Stack Overflow - PHP

Preguntas y respuestas sobre PHP de la comunidad de desarrolladores.
https://stackoverflow.com/questions/tagged/php

GitHub - PHP

Repositorio oficial de PHP y proyectos relacionados.
https://github.com/php

Estos recursos te ayudarán a profundizar tus conocimientos de PHP 8.4 y mantenerte actualizado con las mejores prácticas