PHP – Cheatsheet Rápida (DWES)

SINTAXIS Y ESTRUCTURA

- Bloque PHP en archivo `.php`:

```
<?php
// código aquí
?>
```

- No hace falta cerrar `?>` en archivos solo PHP (buena práctica).
- Comentarios:

```
// una línea
# otra forma una línea
/* varias
   líneas */
```

- Fin de instrucción: `;`
- Mostrar errores (solo en desarrollo):

```
error_reporting(E_ALL);
ini_set('display_errors', '1');
```

TIPOS DE DATOS Y CASTING

- Escalares:
  - `int` (enteros)
  - `float` (decimales)
  - `string` (cadenas)
  - `bool` (true/false)
- Compuestos:
  - `array`
  - `object`
- Especiales:
  - `resource`
  - `null`
- Casting:

```
```

```
(int)$x;
(float)$x;
(string)$x;
(bool)$x;
```

- Comprobaciones:

```
is_int($x); is_float($x); is_string($x);
is_bool($x); is_array($x); is_null($x);
```

- Debug rápido:

```
var_dump($x);
print_r($x);
```

VARIABLES, CONSTANTES Y ÁMBITO

- Variables:

```
$nombre = "Marcos";
$edad = 20;
```

- Case-sensitive: `$x` ≠ `$X`
- Variables variables:

```
$foo = 'bar';
$$foo = 123;     // crea $bar = 123
```

- Constantes:

```
define('PI', 3.14);
const VERSION = '1.0.0';
```

- Constantes mágicas:

```
__FILE__;   // ruta del archivo actual
__DIR__;    // carpeta del archivo
__LINE__;   // número de línea
```

```
__FUNCTION__;
__CLASS__;
```

- Ámbito/Scope:

```
$x = 10;

function prueba() {
    global $x;      // usa la global
    $x = $x + 5;
}
```

- `static` dentro de funciones:

```
function contador() {
    static $c = 0;
    $c++;
    return $c;
}
```

OPERADORES

- Aritméticos: `+ - * / % **`
- Asignación: `=`, `+=`, `-=`, `*=`, `/=`, `.=` (concatena)
- Comparación:
  - `==` igual valor
  - `===` igual valor y tipo
  - `!=` distinto
  - `!==` distinto valor o tipo
  - `<`, `>`, `<=`, `>=`
  - `<=>` spaceship (devuelve -1, 0, 1)
- Lógicos:
  - `&&` (AND)
  - `||` (OR)
  - `!` (NOT)
  - `xor`
- Ternario:

```
$resultado = $condicion ? "sí" : "no";
```

- Null coalesce:

```php
$user = $nombreUsuario ?? 'anónimo';
```

- Null coalesce con asignación:

```php
$config['modo'] ??= 'produccion';
```

CONTROL DE FLUJO

```php
if ($a > 0) {
    ...
} elseif ($a == 0) {
    ...
} else {
    ...
}

switch ($opcion) {
    case 1:
        ...
        break;
    case 2:
        ...
        break;
    default:
        ...
}

while ($i < 10) {
    $i++;
}

do {
    $i++;
} while ($i < 10);

for ($i = 0; $i < 10; $i++) {
    ...
}

foreach ($lista as $valor) {
    ...
}

foreach ($lista as $clave => $valor) {
    ...
}
```

```php
// saltos
break;      // sale del bucle/switch
continue;   // salta a la siguiente iteración
return $x;  // devuelve valor desde función
exit;       // termina el script
```

ARRAYS

Creación

```php
// indexado
$nums = [10, 20, 30];
$nums[] = 40;

// asociativo
$user = [
    'nombre' => 'Ana',
    'edad'   => 25,
];

// multidimensional
$clase = [
    ['nombre' => 'Ana', 'nota' => 9],
    ['nombre' => 'Luis', 'nota' => 7],
];

echo $nums[0];            // 10
echo $user['nombre'];     // Ana
echo $clase[1]['nota'];   // 7
```

Funciones útiles

```php
count($a);                      // tamaño
in_array(3, $a);                // existe valor
array_key_exists('edad', $user); // existe clave

array_keys($user);              // ['nombre', 'edad']
array_values($user);            // ['Ana', 25]

sort($a);   // ordena por valor, reindexa
rsort($a);  // orden inverso
asort($a);  // ordena, mantiene índices
ksort($a);  // ordena por clave

$todo = array_merge($a, $b);
$suma = array_sum($nums);
$media = array_sum($nums) / count($nums);
```

```php
$dobles   = array_map(fn($x) => $x * 2, $nums);
$positivos = array_filter($nums, fn($x) => $x > 0);

$trozos  = array_chunk($nums, 2);    // trozos de 2
$flip    = array_flip($user);        // intercambia claves/valores
```

STRINGS (CADENAS)

```php
$nombre = "Marcos";
$saludo = "Hola ".$nombre;
$saludo2 = "Hola $nombre";   // interpolado
```

Funciones clave

```php
strlen($s);                      // longitud
trim($s);                        // quita espacios extremos
ltrim($s); rtrim($s);

strtolower($s); strtoupper($s);

$pos  = strpos($s, "php");       // posición o false
$parte = substr($s, 0, 5);       // subcadena

$nuevo = str_replace("a", "o", $s);

$trozos = explode(",", "a,b,c"); // ['a','b','c']
$cadena = implode("-", $trozos); // 'a-b-c'

$txt = sprintf("Hola %s (%d)", $nombre, $edad);

// PHP 8
str_contains($s, "abc");
str_starts_with($s, "abc");
str_ends_with($s, "abc");
```

FUNCIONES

```php
function saludar($nombre = "anónimo"): string {
    return "Hola $nombre";
}

echo saludar("Marcos");
echo saludar();        // usa el valor por defecto
```

Tipos y parámetros

```php
function suma(int $a, int $b): int {
    return $a + $b;
}

function total(...$nums): int {
    return array_sum($nums);
}

function inc(&$x): void {   // referencia
    $x++;
}
```

Anónimas y callbacks

```php
$doble = function(int $x): int {
    return $x * 2;
};

echo $doble(3);   // 6

$nums = [1,2,3];
$dobles = array_map(fn($x) => $x * 2, $nums);
```

INCLUDE / REQUIRE

```php
include 'cabecera.php';        // warning si no existe
require 'config.php';          // fatal error si no existe

include_once 'util.php';       // solo una vez
require_once 'conexion.php';
```

Se usan para dividir la aplicación en varios archivos (cabecera, pie, config, etc.).

POO – CLASES Y OBJETOS

Clase básica

```php
class Persona {
    public string $nombre;
    private int $edad;
```

```
    public function __construct(string $n, int $e) {
        $this->nombre = $n;
        $this->edad   = $e;
    }

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

    public function getEdad(): int {
        return $this->edad;
    }
}

$p = new Persona("Ana", 20);
echo $p->saludar();
echo $p->getEdad();
```

Herencia

```
class Empleado extends Persona {
    private float $salario;

    public function __construct(string $n, int $e, float $s) {
        parent::__construct($n, $e);
        $this->salario = $s;
    }

    public function getSalario(): float {
        return $this->salario;
    }

    public function saludar(): string {
        return parent::saludar()." y gano ".$this->salario;
    }
}
```

Visibilidad y estáticos

```
class Config {
    public const APP_NAME = 'MiApp';

    private static string $modo = 'dev';

    public static function getModo(): string {
        return self::$modo;
    }

    public static function setModo(string $m): void {
```

```
        self::$modo = $m;
    }
}

echo Config::APP_NAME;
Config::setModo('prod');
echo Config::getModo();
```

Interfaces y traits

```
interface Loggable {
    public function log(string $mensaje): void;
}

trait TiempoCreacion {
    private int $creadoEn;

    public function inicializarTiempo(): void {
        $this->creadoEn = time();
    }

    public function getCreadoEn(): int {
        return $this->creadoEn;
    }
}

class FileLogger implements Loggable {
    use TiempoCreacion;

    public function __construct() {
        $this->inicializarTiempo();
    }

    public function log(string $mensaje): void {
        // escribir a archivo...
    }
}
```

Métodos mágicos útiles

```
class Usuario {
    private array $datos = [];

    public function __get($prop) {
        return $this->datos[$prop] ?? null;
    }

    public function __set($prop, $valor) {
        $this->datos[$prop] = $valor;
```

```
    }

    public function __toString(): string {
        return "Usuario";
    }
}
```

NAMESPACES

```
// archivo: src/Modelo/Usuario.php
namespace App\Modelo;

class Usuario {
    // ...
}
```

Uso en otro archivo:

```
use App\Modelo\Usuario;

$u = new Usuario();
```

O:

```
$u = new \App\Modelo\Usuario();   // con barra inicial (global)
```

SUPERGLOBALES

- `$_GET` → datos enviados en la URL
- `$_POST` → datos enviados en el cuerpo de la petición
- `$_REQUEST` → mezcla (GET+POST+COOKIE)
- `$_SERVER` → info del servidor/petición:

  ```
  $_SERVER['REQUEST_METHOD']; // GET, POST...
  $_SERVER['PHP_SELF'];       // script actual
  $_SERVER['HTTP_HOST'];      // host
  ```

- `$_FILES` → info de ficheros subidos
- `$_COOKIE` → cookies del cliente
- `$_SESSION` → datos de sesión

- `$_ENV` → variables de entorno

FORMULARIOS (GET / POST) Y VALIDACIÓN BÁSICA

Formulario HTML (POST)

```
<form action="procesa.php" method="post">
  <label>Usuario:
    <input type="text" name="usuario">
  </label>
  <label>Contraseña:
    <input type="password" name="pass">
  </label>
  <button>Entrar</button>
</form>
```

Procesar en PHP

```
<?php
// procesa.php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $user = trim($_POST['usuario'] ?? '');
    $pass = trim($_POST['pass'] ?? '');

    $errores = [];

    if ($user === '') {
        $errores[] = "El usuario es obligatorio";
    }

    if ($pass === '') {
        $errores[] = "La contraseña es obligatoria";
    }

    if ($errores) {
        // mostrar errores
        foreach ($errores as $e) {
            echo "<p>$e</p>";
        }
    } else {
        // seguir con el login, etc.
    }
}
```

GET vs POST

- **GET**

    - Datos en la URL (`?user=ana`)
    - Para búsquedas, filtros, etc.
    - No para contraseñas
- **POST**
    - Datos en el cuerpo
    - Para login, formularios largos, etc.

COOKIES

```php
// Crear/actualizar cookie (antes de sacar HTML)
setcookie("usuario", "Ana", time() + 3600, "/"); // 1h

// Leer cookie
$usuario = $_COOKIE['usuario'] ?? 'invitado';

// Borrar cookie
setcookie("usuario", "", time() - 3600, "/");
```

SESIONES

Iniciar sesión

```php
// al principio de cada script que use sesión
session_start();
```

Login básico

```php
// login.php
session_start();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $user = trim($_POST['user'] ?? '');
    $pass = trim($_POST['pass'] ?? '');

    // Aquí se comprobaría en la BD
    if ($user === 'admin' && $pass === '1234') {
        $_SESSION['user'] = $user;
        header("Location: principal.php");
        exit;
    } else {
        $error = "Credenciales incorrectas";
    }
}
```

```php
// principal.php
session_start();
if (!isset($_SESSION['user'])) {
    header("Location: login.php");
    exit;
}

echo "Hola " . $_SESSION['user'];
```

```php
// logout.php
session_start();
session_unset();   // vacía la sesión
session_destroy(); // elimina la sesión
header("Location: login.php");
exit;
```

SUBIDA DE FICHEROS

Formulario HTML

```html
<form method="post" enctype="multipart/form-data">
  <input type="file" name="fichero">
  <button>Subir</button>
</form>
```

Procesar en PHP

```php
<?php
session_start();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_FILES['fichero'])) {
        echo "No se ha enviado ningún archivo";
        exit;
    }

    $f = $_FILES['fichero'];

    if ($f['error'] === UPLOAD_ERR_OK) {
        $nombreTmp = $f['tmp_name'];
        $nombreOriginal = basename($f['name']);

        $destino = __DIR__ . '/uploads/' . $nombreOriginal;
```

```
        if (move_uploaded_file($nombreTmp, $destino)) {
            echo "Fichero subido correctamente";
        } else {
            echo "Error al mover el fichero";
        }
    } else {
        echo "Error en la subida (código: {$f['error']})";
    }
}
```

CABECERAS HTTP Y REDIRECCIONES

```
// Enviar JSON
header("Content-Type: application/json; charset=utf-8");
echo json_encode(['ok' => true]);

// Redirigir a otra página
header("Location: login.php");
exit;

// Forzar descarga de archivo
header("Content-Type: text/plain");
header("Content-Disposition: attachment; filename=\"notas.txt\"");
readfile("notas.txt");
```

FECHAS, HORAS Y JSON

```
// Fecha y hora actual
echo date('Y-m-d H:i:s');

// Zona horaria
date_default_timezone_set('Europe/Madrid');

// Timestamp actual
$ahora = time();

// JSON
$datos = ['nombre' => 'Ana', 'edad' => 25];
$json  = json_encode($datos);       // array => JSON
$back  = json_decode($json, true);  // JSON => array asociativo
```