
UA . MASTER MOVILES

MÁSTER UNIVERSITARIO EN DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

# PROGRAMACIÓN HIPERMEDIA PARA DISPOSITIVOS MÓVILES

## Introducción a PHP

# CONTENIDO

1. Introducción
2. Variables y tipos de datos
3. Operadores
4. Estructuras de control
5. Funciones
6. Clases y objetos

# INTRODUCCIÓN

- Lenguaje de *scripting*
- No necesita compilación: guardar y listo!
- La extensión de los ficheros tiene que ser "`.php`"
- Permite mezclar código HTML y PHP en el mismo documento.
- El código PHP se procesa en el servidor.

  → El cliente nunca podrá ver el código PHP.
- Necesita servidor Web que soporte PHP (Apache)

# INTRODUCCIÓN

- Inicio y fin de secciones de código PHP:

```
<?php  …  ?>
<? … ?>
```

- Usa el “punto y coma” como separador de instrucciones:

```
echo “¡Hola mundo!”;
```

- Comentarios:

```
// Comentario de una sola línea
# Comentario de una sola línea
/* Esto es un
comentario de varias líneas */
```

# HOLA MUNDO EN PHP

## Código PHP / HTML

```
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <?php
        echo "¡Hola mundo!";
    ?>
  </body>
</html>
```

## Salida

¡Hola mundo!

# VARIABLES, TIPOS DE DATOS Y OPERADORES

# VARIABLES

- Las variables van precedidas por el símbolo “$”:

```
$variable = 15;
```

  - No se especifica el tipo.
  - No es necesario declarar las variables.
  - El nombre tiene que empezar por letra o subrayado.
  - Sensibles a mayúsculas y minúsculas.
  - No se admiten caracteres como: - @ . ¡ +
  - No tienen un tipo fijo.

# TIPOS DE DATOS

| | | |
|---|---|---|
| Escalares | `boolean` | Valores `true` o `false`. |
| | `int` | Enteros positivos o negativos. |
| | `float` | Números decimales. |
| | `string` | Cadenas de texto. |
| Compuestos | `array` | Lista de elementos. |
| | `object` | Contenedor de objetos de datos. |

# TIPOS DE DATOS - CONSIDERACIONES

- Booleanos:
  - Son falsos: `null, 0, 0.0, “”`, arrays vacíos, objetos vacíos
- Enteros:
  - No hay `unsigned`.
- Cadenas, se pueden representar de dos formas:
  - Comillas simples: `'¡Hola mundo!'`
    - Caracteres de escape solo para comillas simples: `\'`
    - Las comillas dobles (“) son un carácter más.
    - Las variables NO se interpretan.
  - Comillas dobles: `“¡Hola mundo!”`
    - Caracteres de escape: `\n, \r, \t, …`
    - Carácter de escape para comillas dobles `\”`
    - Sí que interpreta variables: `“Hola $nombre”`

# ARRAYS

- Se crean mediante "`array()`"

```
$variable = array();  // array vacío
```

- Se accede con "`[índice]`", empezando en cero:

```
echo $variable[0];
```

- Para asignar valor también usamos "`[indice]`":

```
$variable[0] = 'nuevo valor';
```

- Para añadir un elemento al final del array usamos "`[]`":

```
$variable[] = $foo;
```

# ARRAYS

- Hay dos tipos de arrays:
  - Arrays indexados (posiciones numéricas):

```
$miArray = array('Pedro', 'Juan', 'María');
echo $miArray[1];       // Imprime Juan
$miArray[] = 'Laura';   // Añadimos Laura
```

  - Arrays asociativos (tipo tabla *hash*):

```
$notas = array( 'Juan' => 6, 'Luis' => 9);
echo $notas['Juan'];
$notas['Laura'] = 8.5;
```

# OPERADORES

- Aritméticos: `+, -, /, *, %`
- Incremento y decremento: `++, --`
- Lógicos: `&&, and, ||, or, xor, !`
- Concatenación de cadenas: `.` (punto)
- Comparación: `==, ===, !=, <>, !==, <, <=, >, >=`
- Asignación: `=, +=, -=, *=, /=, %=, .=`

```
$var1 = 5;
$var2 = 2;
$var2++;
$result = $var1 * $var2;
$result += 5;
```

# ESTRUCTURAS DE CONTROL

# ESTRUCTURAS DE CONTROL

- Expresiones "`if – elseif – else`":

```
if( $expresion1 ) {
     echo 'La expresión 1 es válida';
}
elseif( $expresion2 ) {
     echo 'La expresión 2 es válida';
}
else {
     echo 'Ninguna expresión es válida';
}
```

- Ejemplo:

```
<?php
$var = 15;
if( $var == 15 ) {
     echo 'La variable es igual a "15"';
}
?>
```

# ESTRUCTURAS DE CONTROL

- Expresión "switch":

```
switch( $expresion )
{
    case $value1:
         echo 'Expresión igual a: '. $value1';
         break;
    case $value2:
         echo 'Expresión igual a: '. $value2;
         break;
    default:
         echo 'No hay coincidencias';
}
```

# ESTRUCTURAS DE CONTROL

- Repetir bloques de código con "`while`":

```
while( $expresion ) {
    echo 'La expresión es cierta';
}
```

- Repetir bloques de código con "`do - while`":

```
do {
    echo 'La expresión es cierta';
} while( $expresion );
```

# ESTRUCTURAS DE CONTROL

- Repetir bloques de código con "`for`":

```
for( $indice = 0; $indice < $MAX; $indice++ ) {
    echo 'Ejemplo de bucle for';
}
```

- Repetir sobre los valores de un array con "`foreach`":

```
$miArray = array(1,2,3,4,5,6);
foreach( $miArray as $valor )
    echo $valor;

$notas = array( 'ana' => 4, 'juan' => 7 );
foreach( $notas as $clave => $valor )
    echo "$clave tiene una nota de $valor";
```

# FUNCIONES

# FUNCIONES

- Se declaran mediante la palabra reservada "`function`".
- El nombre tiene que empezar por letra o subrayado, nunca por número.
- Permiten recursividad.
- Para devolver valores utilizamos "`return`"
- Ejemplo:

```
function suma( $num1, $num2 ) {
    return $num1 + $num2;
}
echo suma( 15, 5 );    // Imprime 20
```

# FUNCIONES

- Parámetros con valor por defecto:

```
function f1( $x = "Juan" ) {
    echo "Hola Sr. $x";
}
f1( "Luis" );     // Imprime: Hola Sr. Luis
f1();             // Imprime: Hola Sr. Juan
```

- Parámetros por referencia con "&":

```
function f2( &$x ) {
    $x++;
}
$variable = 1;
f2( $variable );
echo $variable;   // Imprime: 2
```

# FUNCIONES - ÁMBITO DE LAS VARIABLES

## Local

```
$x = 1;

function foo() {
    $x = 2;
}

foo();
echo $x;   // = 1
```

## Global

```
$x = 1;

function foo() {
    global $x;
    $x = 2;
}

foo();
echo $x;   // = 2
```

# FUNCIONES PREDEFINIDAS ÚTILES

| Función | Descripción |
|---|---|
| `echo var/string` | Imprimir el contenido de una variable o cadena. |
| `print var/string` | Igual que echo |
| `var_dump(var)`<br>`print_r(var)` | Información de una variable |
| `isset(var)` | Determinar si una variable existe y tiene valor |
| `strlen( string )` | Obtiene la longitud de una cadena. |
| `count( $lista )` | Número de elementos de un array. |
| `in_array(var,arr)` | Devuelve cierto si encuentra 'var' en el array. |

# CLASES Y OBJETOS

# CLASES

- Se definen mediante la palabra reservada "`class`":
- Podemos indicar el ámbito de las variables y funciones con "`public`", "`private`" o "`protected`".
- Para crear instancias de una clase usamos "`new`".

```
class Persona {
        private $nombre;
        public function setNombre( $nombre ) {
                $this->nombre = $nombre;
        }
        public function getNombre() {
                return $this->nombre;
        }
}

$juan = new Persona();
$juan->setNombre( 'Juan' );
echo $juan->getNombre();
```

# CLASES – CONSTRUCTOR Y DESTRUCTOR

El constructor/destructor se definen con `__construct/__destruct`

```
class Persona {
  private $nombre, $edad;
  function __construct( $nombre, $edad = 0 ) {
     $this->nombre = $nombre;
     $this->edad = $edad;
  }
  public function envejecer() {
     $this->edad++;
  }
  public function toString() {
     echo $this->nombre .': '. $this->edad .' años';
  }
}

$juan = new Persona("Juan", 24);
$juan->envejecer();
$juan->toString();   // Imprime "Juan: 25 años"
```

# CLASES - HERENCIA

- Para que una clase herede de otra clase usamos "`extends`".
- Por ejemplo:

```
class Empleado extends Persona
{
    private $empresa;

    function __construct( $nombre, $edad, $empresa ) {
        parent::__construct( $nombre, $edad );
        $this->empresa = $empresa;
    }
}

$personas = array(
                new Persona( "Juan", 22 ),
                new Empleado( "Luis", 30, "UA" ) );
```

# ¿PREGUNTAS?