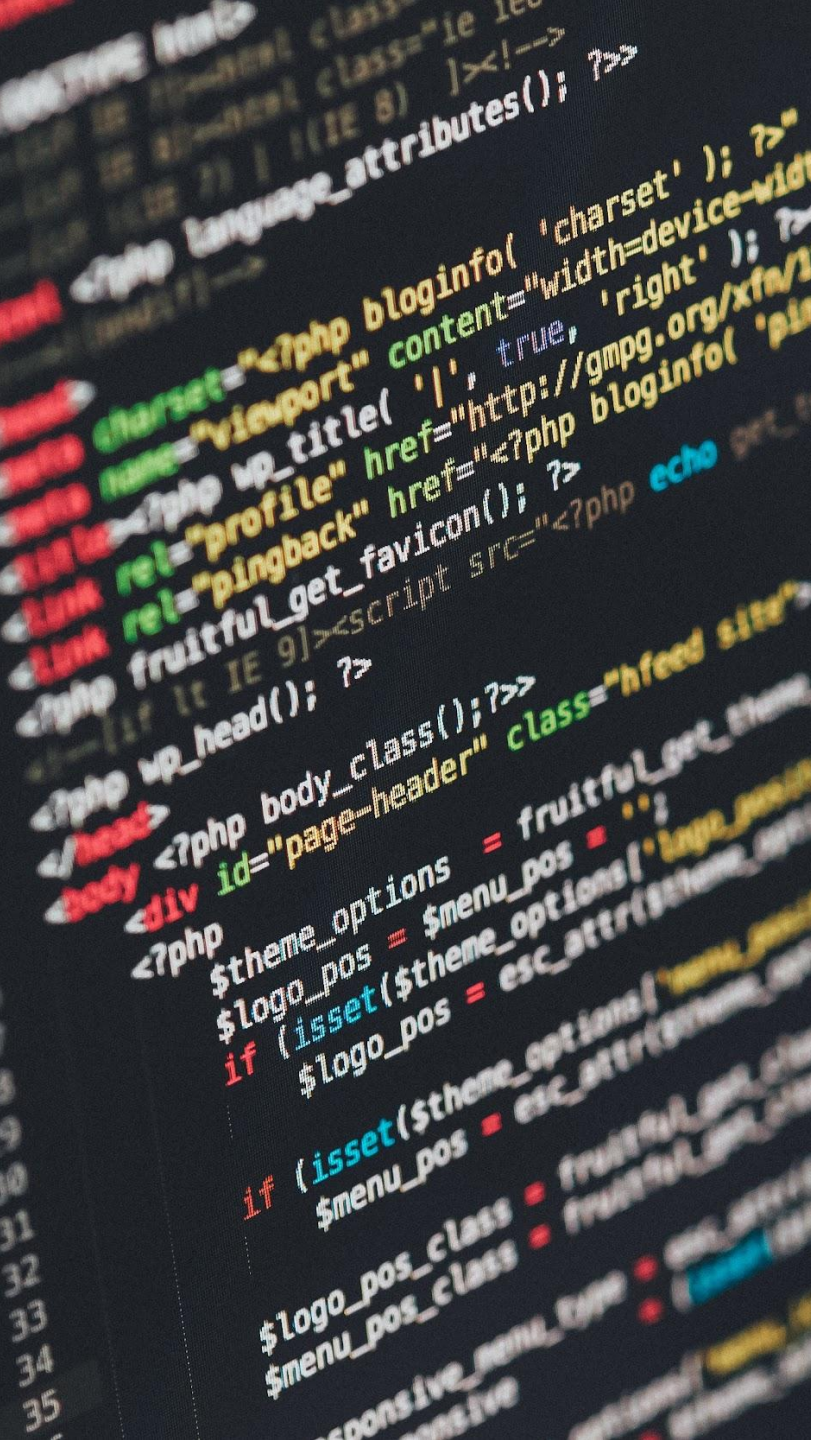

INTRODUCCIÓN A PHP

DISEÑO DE SISTEMAS SOFTWARE

Contenido

1. Introducción
2. Variables y tipos de datos
3. Operadores
4. Estructuras de control
5. Funciones
6. Programación orientada a objetos
7. Espacios de nombres

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)
  → Se puede ejecutar en línea de comandos

```
$ php fichero.php
```

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

```php
<?php
    echo “¡Hola mundo!”;
?>
```

Salida

```
¡Hola mundo!
```

Introducción a PHP

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
- El nombre es sensible 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`

Tipos de datos: consideraciones

Comillas simples

```
'¡Hola, mundo!'
```

- Caracteres de escape sólo 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()` o `[]`

```php
$variable = array();  // array vacío
$variable = [];       // array vacío
```

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

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

- Para asignar valor también usamos "`[índice]`"

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

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

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

Arrays

Hay dos tipos de arrays:

- Arrays indexados (posiciones numéricas)

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

- Arrays asociativos (tipo tabla *hash*)

```php
$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;
```

Operador nullsafe

```php
// before nullsafe operator
$country =  null;

if ($session !== null) {
  $user = $session->user;

  if ($user !== null) {
    $address = $user->getAddress();

    if ($address !== null) {
      $country = $address->country;
    }
  }
}

// using nullsafe operator
$country = $session?->user?->getAddress()?->country;
```

Introducción a PHP

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

```
$var = 15;
if( $var == 15 ) {
    echo 'La variable es igual a “15”';
}
```

Estructuras de control

- Expresión `switch`

```php
switch( $expresion )
{
    case $valor1:
        echo 'Expresión igual a: '. $valor1;
        break;
    case $valor2:
        echo 'Expresión igual a: '. $valor2;
        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";
```

Introducción a PHP

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`

```php
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

- Parámetros con nombre

```
htmlspecialchars($string, default, default, false);
// vs
htmlspecialchars($string, double_encode: false);
```

Se puede especificar a qué parámetros se está dando valor, útil cuando hay varios parámetros con valor por defecto.

Funciones: ámbito de las variables

Local

```php
$x = 1;

function foo() {
    $x = 2;
}

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

Global

```php
$x = 1;

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

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

Funciones predefinidas útiles

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

Introducción a PHP

PROGRAMACIÓN ORIENTADA A OBJETOS

Clases

- Se definen mediante la palabra reservada `class`
- Podemos indicar la visibilidad de los atributos y métodos con `public`, `private` o `protected`

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

Objetos

- Para crear instancias de una clase usamos `new`
- Para acceder a los atributos o métodos públicos de un objeto usamos el operador `->`

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

Atributos y constantes

- Para acceder a los atributos dentro de los métodos de una clase debemos usar siempre `$this->atributo`
- Para acceder a atributos estáticos y constantes usamos `self::constante` y `self::$atributo`

```php
class Circulo {
    private $radio;
    const PI = 3.1416;

    // ...

    public function area() {
        return self::PI * $this->radio**2;
    }
}
```

Constructor y destructor

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

```php
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"
```

Herencia

Para que una clase herede de otra clase usamos `extends`

```
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" )
);
```

Interfaces

- Se definen mediante la palabra reservada `interface`
- Se usan mediante la palabra reservada `implements`

```php
interface IPrintable {
    public function print();
}

class Documento implements IPrintable {
    private $contenido;
    function __construct($contenido) {
        $this->contenido = $contenido;
    }
    public function print() {
        echo $this->contenido;
    }
}
```

Introducción a PHP

Espacios de nombres

Espacios de nombres

Los espacios de nombres se declaran con la instrucción

`namespace mi\espacio\de\nombres`

```php
<?php
    namespace es\dominio\miaplicacion\Util;
    function saluda($nombre) {
        echo "Hola $nombre\n";
    }
    class Logger {
        public function warning($mensaje) {
            echo "[WARNING]" . $mensaje . "\n";
        }
    }
?>
```

Espacios de nombres

- Para usar funciones o clases definidas en otro espacio de nombres es necesario incluir el archivo con las definiciones con `require()` o `require_once()`
- Se debe incluir el espacio de nombres completo como prefijo al nombre de las funciones o clases

```php
<?php
    require_once("util.php");
    es\dominio\miaplicacion\Util\saluda("Juan");
?>
```

Espacios de nombres

Se pueden asignar alias a los paquetes con `use`

```php
<?php
    use es\dominio\miaplicacion\Util as U;
    require_once("util.php");
    U\saluda("Juan");
    $logger = new U\Logger();
    $logger->warning("Esto es un warning");
?>
```

Espacios de nombres

Si no se especifica un alias con `use` es equivalente a usar como alias el último elemento del espacio de nombres

```php
<?php
    use es\dominio\miaplicacion\Util;
    require_once("util.php");
    Util\saluda("Juan");
    $logger = new Util\Logger();
    $logger->warning("Esto es un warning");
?>
```

Espacios de nombres

Se pueden importar clases o funciones específicas de un paquete, en este caso se usan sin prefijo

- Las clases se importan con `use`
- Las funciones se importan con `use function`

```php
<?php
    use es\dominio\miaplicacion\Util\Logger;
    use function es\dominio\miaplicacion\Util\saluda;
    require_once("util.php");
    saluda("Juan");
    $logger = new Logger();
    $logger->warning("Esto es un warning");
?>
```

¿PREGUNTAS?