Introducción a PHP

DWES UD2

1.- Introducción

Las páginas web estáticas, con extensión **.html**, se pueden ejecutar sin disponer de un servidor web.

Es el navegador el que interpreta el código de dichos archivos.

Lo hemos visto en el primer ejercicio en el que se probaron las etiquetas HTML.

Al hacer doble clic a un archivo .html el archivo se abre en el navegador directamente.

1.- Introducción

Para poder ejecutar archivos **.php** se necesita que el que ejecute estos archivos sea el servidor web.

A partir de ahora nuestros proyectos se deberán guardar en el repositorio de páginas web.

En **apache** el repositorio de páginas web es el directorio **htdocs**.

En principio en el servidor web sólo tendremos una aplicación web, **el proyecto en el que estemos trabajando en cada momento**. Para esto nos tenemos que asegurar que en el directorio **htdocs** no haya ningún archivo que no sea de nuestro proyecto.

1.- Introducción

Sin embargo, gracias a los **host virtuales** vistos en la unidad anterior y que se verán más detalladamente en el módulo **DAW** (desplieque de aplicaciones web) podremos tener diferentes proyectos a la vez en el mismo servidor web.

Para el desarrollo de las actividades del curso haremos uso de los hosts virtuales, por ejemplo se puede tener un host virtual llamado **localhost.actividades** en el que ir haciendo los ejercicios y otro llamado **localhost.proyecto** para realizar el proyecto del trimestre.

Evidentemente se pueden crear todos los que uno necesite.

2.- Integración de PHP en HTML

En este curso vamos a utilizar **PHP** como lenguaje de programación de páginas web dinámicas.

Como se comentó en la unidad anterior los archivos para crear páginas web dinámicas con PHP tendrán la extensión **.php** pero dentro contendrán código **HTML** combinado con código **PHP**.

Lo primero que vamos a ver es cómo podemos **integrar** el código PHP dentro de código HTML, pero también veremos cómo integrar código HTML dentro de PHP

2.- Integración de PHP en HTML

Ejercicio: Crea el archivo **prueba.php** en el virtual host **localhost.actividades**

```
<!doctype html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Primera prueba php</title>
	</head>

	<body>
		Este es un archivo php que se encuentra en el servidor.
	</body>
</html>
```

Entra al navegador y accede a la URL: **http://localhost.actividades/prueba.php**

2.- Integración de PHP en HTML

Es posible y bastante habitual que en un archivo .php sólo haya código PHP.

Esto ocurre cuando solo se busca procesar algo.

Por ejemplo al llamar a alguna función.

Ejercicio: Crea un archivo **info.php** en **htdocs**

```
<?php
    phpinfo();
?>
```

Abre el navegador y accede a la URL: **http://localhost/info.php**

2.- Integración de PHP en HTML

El mejor sitio para consultar la sintaxis y funciones de PHP es en la documentación oficial la cual siempre se encuentra actualizada y la mayor parte en español.

Podéis encontrar otros sitios web con documentación pero es posible que sea para versiones antiguas de PHP.

Lo bueno del sitio oficial es que te indica las versiones en las que funciona cada una de las funciones y si está desaconsejado o no su uso.

2.- Integración de PHP en HTML

PHP dentro de HTML

```
<article>
<?php

        //código con instrucciones php

?>
</article>
```

2.- Integración de PHP en HTML

HTML dentro de PHP

```
<?php

    echo ‘<h1>Bienvenido a mi página web</h1>’;

?>
```

3.- Características básicas de PHP

Comentarios

Comentarios de una línea:

```
// Función para loguearse
```

Comentarios de varias líneas:

```
/* Script desarrollado por:
    Álex Torres
    fecha: 10-9-17 */
```

El código comentado en PHP no aparecerá en documento HTML final que se genera.

*Recuerda que los comentarios en HTML sí que aparecerán: <!-- -->

3.- Características básicas de PHP

Comentarios con phpDoc

Es recomendable utilizar los comentarios estilo phpDoc de manera que el IDE los pueda reconocer y que posteriormente se pueda generar la documentación de manera automática

```
/**
* Bloque de comentario phpDoc
*
*
*/
```

3.- Características básicas de PHP

Comentarios con phpDoc

Con phpDoc es recomendable comentar todos los elementos estructurales:

- Archivo
- Código importado: require(_once) e include(_once)
- Clases
- Interfaces
- Rasgos (traits)
- Funciones y métodos
- Propiedades de las clases
- Constantes

Antes de continuar echemos un vistazo a la cómo se documenta con phpDoc

Ejercicio

Introducción

Realizar el ejercicio 1 del boletín.

3.- Características básicas de PHP

Inclusión de ficheros externos

Cuando se desarrolla un programa es habitual que crezca y resulte difícil encontrar código.

También es más que probable que haya trozos del código que se necesiten en diferentes sitios, en nuestro caso en diferentes páginas web de la aplicación.

En PHP existen unas instrucciones que permiten añadir todo el contenido de otro archivo en un punto dado del programa.

include

include_once

require

require_once

3.- Características básicas de PHP

Inclusión de ficheros externos

Todas estas **funciones** buscan el archivo indicado en caso de encontrarlo añaden el contenido en el punto donde se encuentra la llamada a la función.

```
include('ruta_archivo_php');
include_once('ruta_archivo_php);
```

```
require('ruta_archivo_php');
require_once("ruta_archivo_php");
```

Las funciones **include** si no encuentran el archivo da un aviso pero continúa. sin embargo las instrucciones **require** si no encuentran el archivo detienen la ejecución mostrando un error.

Las versiones con **_once** sirven para asegurar que el archivo solo se añade una vez ya que si se añade más de una vez en diferentes puntos puede dar errores, por ejemplo, si el archivo añadido contiene una función.

3.- Características básicas de PHP

Inclusión de ficheros externos

Es habitual que si un archivo está pensado para que siempre que se use se haga mediante su inclusión con **include** o **require** se le ponga la extensión **.inc.php**.

Por ejemplo:

formulario.inc.php

Ejercicio

Inclusión de ficheros externos

Realizar el ejercicio 2 del boletín.

3.- Características básicas de PHP

Variables y tipos de datos

Las variables en PHP siempre empiezan por el símbolo **$**

A continuación del **$** tiene que haber una **letra** o el **símbolo _**

Luego ya puede contener números

$edad	$cantidad pelotas
$_piso	$_3tipos
$variable2	$primer_apellido
$nobreCalle	$i

3.- Características básicas de PHP

Variables y tipos de datos

PHP es un lenguaje de programación **débilmente tipado**.

No se especifica el tipo de dato que se va a almacenar en cada variable.

En las variables se puede cambiar el tipo de dato que se contiene según interese.

Para asignar valor a las variables se utiliza el operador **=**

```
$variable = 5;

$variable = “Federico”;

$variable = TRUE;
```

3.- Características básicas de PHP

Variables y tipos de datos

Los tipos de datos que se pueden almacenar en variables en PHP son:

- booleano (boolean): **TRUE/FALSE**, **0** = false, **cualquier_otro_número** = true.
- entero (integer): número sin decimales.
- real (float): número con decimales.
- cadena (string): conjunto de caracteres delimitados por comillas.
- null: tipo especial que indica que la variable no tiene valor.

3.- Características básicas de PHP

Variables y tipos de datos

```
$booleano = FALSE;

$edad = 58;

$kilos = 5.3;

$nombreCompleto = “Ana Gómez Parra”;

$otro = null;
```

3.- Características básicas de PHP

Variables y tipos de datos

Si en una operación intervienen operaciones de distintos tipos el módulo de php intentará convertirlas a un tipo común (casting).

```
$cantidad = 3;

$precio = 1.6;

$total = $cantidad * $precio;
```

3.- Características básicas de PHP

Variables y tipos de datos

El casting se puede realizar de manera forzada:

```
$cantidad = 3;
$precio = 1.6;
$total = $cantidad * (int)$precio;
//¿Cuánto valdrá $total?
```

En la documentación puedes ver los tipos de conversiones permitidos.

3.- Características básicas de PHP

Constante

Las constantes son identificadores que almacenan un valor que no puede variar durante la ejecución del programa. Se usa la palabra reservada **define**.

```
<?php
        define (“PI”, 3.141592);
        define (“NOMBRE”, “Luisa”, true);

        $radio = 5;
        $superficie = PI * $radio * $radio;
?>
```

Si se añade **true** al final de la sentencia **define** significa que el nombre será **CI**;

Se permiten los tipos **integer**, **float**, **string**, **boolean** y **null**.

3.- Características básicas de PHP

Fechas y horas

No existe un tipo de datos para trabajar con fechas y horas.

Se almacena como un número entero (fecha UNIX - segundos desde 1/1/70 00:00:00)

Es importante establecer la zona horaria correctamente para poder mostrar los datos bien.

```
<?php
        date_default_timezone_set('Europe/Madrid');
?>
```

3.- Características básicas de PHP

Fechas y horas

Función **time** devuelve la fecha y hora actual en formato UNIX.

```
<?php
        $start = time();
        /*
                Muchas instrucciones php
        */
        echo “Esta página se ha generado en ”. time()-$start .” segundos”;
?>
```

Si se quiere mostrar información detallada de la fecha se necesita usar una función que permita obtener datos concretos como día, mes, hora… es la función **date**.

3.- Características básicas de PHP

Fechas y horas

La función **date** puede funcionar directamente con la fecha y hora actual o con una fecha que se le indique de manera directa.

```
<?php
        //Las dos instrucciones almacenarán los mismo

        $hoy = date("l, d M Y");

        $hoy = date("l, d M Y", time());
?>
```

El resto de opciones de formateo de la fecha se pueden consultar en la documentación.

3.- Características básicas de PHP

Expresiones y operadores

Como en otros lenguajes de programación, en PHP también se utilizan expresiones, por ejemplo sumar, restar, asignar.

En las expresiones usualmente aparecerán operadores, que son similares a los de otros lenguajes de programación.

- Asignación: **=**
- Aritméticas: **+ - * / % ++ --**
- Comparación:**> < >= <= == != !== ===**
- Comparaciones booleanas: **&& || !**

En [la documentación] puedes ver todos los operandos.

3.- Características básicas de PHP

Ámbito de las variables

Se pueden usar las variables en cualquier punto del programa, si no existe se creará la primera vez que se utilice, reservándole espacio en memoria.

Dependiendo del primer lugar donde aparezca la variable por primera vez se decidirá dónde podrá ser usada (visibilidad de la variable).

Si la variable aparece dentro de una función por primera vez, esa variable será local a la función.

Una vez acaba la función esa variable desaparece y su valor se pierde.

3.- Características básicas de PHP

Ámbito de las variables

```
$a = 1;
$b = 2;

function prueba() {
        $c = $a;

        global $b;
        $c = $b;
}
```

3.- Características básicas de PHP

Ámbito de las variables

```
function contador() {
        static $a=0;

        $a++;
}
```

3.- Características básicas de PHP

Generación de código HTML

Ya hemos visto anteriormente cómo introducir código HTML desde instrucciones PHP.

Hemos utilizado la instrucción **echo**. Existen también las instrucciones **print** y **printf**.

```
<?php
        $modulo = “DWES”;
        echo “<p> Módulo”;
        print $modulo;
        print “</p>”;
?>
```

Realmente **echo** y **print** no son funciones por eso no hace falta poner los ( );

3.- Características básicas de PHP

Generación de código HTML

Mediante el operador **.** se pueden concatenar/unir los argumentos en las instrucciones **echo** y **print**.

```
<?php
        $modulo = “DWES”;
        echo “<p> Módulo” . $modulo . “</p>;
?>
```

3.- Características básicas de PHP

Generación de código HTML

La función **printf** es mucho más completa que **echo** y **print** pero su uso se suele a limitar a los casos en los que interesa **formatear** los datos que se van a mostrar (cantidad de decimales, longitud…)

Documentación (apartado **format**).

```
<?php
        $ciclo = “DAW”;
        $modulo = “DWES”;
         printf(“%s es un módulo de %d curso de %s”, $modulo, 2, $ciclo);
?>
```

3.- Características básicas de PHP

Bloques de instrucciones

En PHP se pueden utilizar las llaves **{ }** para agrupar un grupo de sentencias.

Mediante las estructuras de control se puede decidir si un bloque de instrucciones se ejecuta o no, o si se ha de repetir la ejecución de dicho bloque

3.- Características básicas de PHP

Estructuras de control

PHP aún siendo un lenguaje de programación de **scripts** (guiones) tiene como cualquier otro lenguaje de programación de alto nivel sentencias que permiten alterar el flujo predefinido de ejecución (instrucción por instrucción de arriba hacia abajo).

Estas sentencias ya las conoces de otros lenguajes de programación como Java.

if **switch** **while** **do/while** **for**

3.- Características básicas de PHP

```
<?php
	if ($a < $b)
		echo "a es menor que b";
	else if ($a > $b)
		echo "a es mayor que b";
	else
		echo "a es igual a b";
?>
```

```
<?php
	if ($a < $b) {
		echo "a es menor que b";
	} else if ($a > $b) {
		echo "a es mayor que b";
	} else {
		echo "a es igual a b";
	}
?>
```

3.- Características básicas de PHP

```
<?php
	if ($a < $b)
	{
		echo "a es menor que b";
	}
	else if ($a > $b)
	{
		echo "a es mayor que b";
	}
	else
	{
		echo "a es igual a b";
	}
?>
```

3.- Características básicas de PHP

```
<?php
        switch ($a) {
                case 0:     echo "a vale 0";
                            break;
                case 1:     echo "a vale 1";
                            break;
                default:    echo "a no vale 0 ni 1";
        }
?>
```

3.- Características básicas de PHP

```
<?php
        switch ($a) {
                case 0:     echo "a vale 0";
                 case 1:    echo "a vale 1";
                 case 2:    echo "a vale 2";
                             break;
                 default:   echo "a no vale 0 ni 1";
        }
?>
```

¿Qué hace el script anterior?

3.- Características básicas de PHP

```
<?php
	$a = 1;
	while ($a < 8) {
		$a += 3;
	}
	echo $a;
?>
```

¿Qué hace el script anterior?

3.- Características básicas de PHP

```
<?php
	$a = 5;
	do {
		$a -= 3;
	} while ($a > 10);
	echo $a;

?>
```

¿Qué hace el script anterior?

3.- Características básicas de PHP

```
<?php
	for ($a = 0; $a<10; $a++) {
		echo $a;
		echo "<br>";
	}
?>
```

¿Qué hace el script anterior?

3.- Características básicas de PHP

```
<?php
	for ($a = 5; $a<10; $a+=3) {
		echo $a;
		echo "<br>";
	}
?>
```

¿Qué hace el script anterior?

Ejercicio

Inclusión de ficheros externos

Realizar los ejercicios 3 y 4 del boletín.

3.- Características básicas de PHP

Cadenas de texto

Para declarar/usar cadenas de texto se puede hacer uso de comillas simples o comillas dobles. La diferencia es que si se usan comillas dobles podemos incluir variables dentro y el módulo de PHP las convertirá en su valor.

```
<?php
        $edad = 24
        echo “Edad:  $edad”;
        echo “<br>”;
        echo “Juan tiene ${edad}años”;
?>
```

3.- Características básicas de PHP

Programming tip: Cadenas de texto

Dado el funcionamiento y cómo se codifica el lenguaje HTML, en el que se usan de manera muy habitual las comillas dobles (argumento-valor en las etiquetas):

```
<a href=”fotos.php”><img src=”camara.png” alt=”Galería fotográfica”></a>
```

En PHP para declarar y usar cadenas se recomienda el uso de comillas simples.

```
<?php
    echo ‘<a href=”fotos.php”><img src=”camara.png” alt=”Galería fotográfica”></a>’;
?>
```

Si se quiere imprimir una comilla simple se tendrá que usar la secuencia de escape: **\’**

3.- Características básicas de PHP

Cadenas de texto

Las cadenas de texto pueden usar dos operadores exclusivos para ellas:

Concatenar: **.**

Concatenar y asignar: **.=**

```
<?php
    $a = “Módulo”;
    $b = $a . “DWES”;
    $a .= “DWES”;
?>
```

¿Qué contienen $a y $b?

3.- Características básicas de PHP

Cadenas de texto

Para el uso de cadenas existe un gran conjunto de funciones, todas ellas se pueden consultar en la documentación oficial, según las vayamos a utilizar durante el curso se explicarán.

A continuación se muestran dos ejemplos:

```
<?php
        $nombre = “Antonio”;

        echo strlen($nombre);

        $nombreMayus = strtoupper($nombre);
?>
```

3.- Características básicas de PHP

Arrays

Los arrays permiten almacenar varios valores del mismo tipo de dato.

Cada miembro del array se almacena en una posición.

La posición (índice/clave) que puede ser un **valor** o un **string** (array asociativo).

Existen varias maneras de declarar un array.

3.- Características básicas de PHP

Arrays

```
$colores = array("rojo", "verde", "azul");

$colores = ["rojo", "verde", "azul"];     // a partir de php 5.4


$ciclos = array("DAW" => "Desarrollo web", "DAM" => "Desarrollo multiplataforma");


$colores = [1 => "rojo", 5 => "verde", "0" => "azul"];
```

3.- Características básicas de PHP

Arrays

Hay que tener mucho cuidado con los índices de los arrays. Si no se indica el índice, este será numérico y **comenzará en cero**.

Si se declara el array asociativo ya no se podrá acceder a él con las posiciones numéricas.

Se pueden mezclar claves numéricas y asociativas. Si la clave es asociativa y es un número, se podría acceder al elemento con el número.

3.- Características básicas de PHP

Arrays

Habitualmente si se quiere acceder a un array mediante su índice numérico y también mediante una clave asociativa se suele realizar de la siguiente manera.

De hecho hay funciones del sistema que cuando devuelven un array lo hacen de esta manera.

$ciclos = array(0 => "Desarrollo web", "DAW" => "Desarrollo web", 1 => "Desarrollo multiplataforma", "DAM" => "Desarrollo multiplataforma");

3.- Características básicas de PHP

Arrays

Para acceder a las posiciones de los arrays se utiliza la notación de corchetes **[ ]**.

```
$colores = ["rojo", "verde", "azul"];
echo $colores[0];
```

3.- Características básicas de PHP

Arrays

PHP es muy flexible en la creación de arrays no siendo necesario indicar el tamaño del array.

Tampoco es necesario indicar la clave, PHP asignará la siguiente automáticamente.

```
$colores[0] = "rojo";
$colores[1] = "azul";
$colores[ ] = "verde";

$numeros[ ] = "uno";
$numeros[ ] = "dos";
```

3.- Características básicas de PHP

Arrays

Recuerda que mediante la función **print_r( )** se puede mostrar todo el contenido del array incluidas sus claves. Aunque esta es una función solamente recomendable para la depuración de los scripts.

Para recorrer un array se puede utilizar un bucle **for** como es habitual en todos los lenguajes de programación, en este caso es necesario conocer la longitud del array, para esto se dispone de la función **count($array)** que devolverá un número entero.

3.- Características básicas de PHP

Arrays

Existe otra forma más aconsejable para recorrer un array, es mediante el uso de la función **foreach**. Existen dos maneras de de usarla:

```
$numeros = ["uno", "dos", "tres"];

foreach($numeros as $valor) {
        echo $valor ."<br>";

foreach($numeros as $clave => $valor) {
        echo "[". $clave ."]". $valor ."<br>";
```

3.- Características básicas de PHP

Arrays multidimensionales

En PHP también existen los **array multidimensionales**. El funcionamiento es exáctamente el mismo que los array normales.

Para el acceso a los datos simplemente hay que usar la clave adecuada para cada dimensión.

```
$array =  [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

echo $array[1][0];
```

3.- Características básicas de PHP

Arrays multidimensionales

Hay que tener en cuenta que no se declara el tamaño de las dimensiones de los arrays y que se podría mezclar la clave con índice o asociativo.

```
$array = [["Ana", "Luis", "Marta"],
          "colores" => ["amarillo", "verde"],
          ["saludo" => "hola", "despedida" => "adiós", "¿Qué tal estás?"]];
```

Ejercicio

Inclusión de ficheros externos

Realizar el ejercicio 5 del boletín.

Corrección de ejercicios

Presentación proyecto del trimestre

3.- Características básicas de PHP

Funciones para tipos de datos

Mediante funciones se puede comprobar el tipo de dato que está almacenando en un momento dado una variable. Se puede hacer de dos maneras, consultando el tipo de dato o preguntando si es de un tipo de dato específico:

```
string gettype ($variable)
```

Se obtendrá en el string un valor entre los siguientes: array, boolean, double, integer, object, string, null, resource o unknown type.

```
boolean is_array($variable)
boolean is_numeric()
boolean is_bool()
boolean is_integer()
```

3.- Características básicas de PHP

Funciones para tipos de datos

¿Qué son? Breve descripción

Funciones del sistema

Funciones propias

3.- Características básicas de PHP

Comprobar si una variable existe

En ocasiones es interesante saber si la variable existe o no, por ejemplo cuando se recibe un formulario y se quiere comprobar que han llegado todas las variables. Para ello existe una función que devuelve un valor booleano: **isset**

También existe una función que permite destruir variables: **unset**

```
<?php
        $a = 25;
        $existe = isset($a);
        unset($a);
        $existe = isset($a);
?>
```

¿Qué valor tiene $existe en cada caso?

Ejercicio repaso

Se utiliza para definir constantes	date_default_timezone_set
Muestra una cadena con formato	define
Indica si una variable está definida y su valor no es null	settype
Establece el tipo de una variable	date
Obtiene una cadena de texto a partir de una fecha/hora	printf
Indica si una variable es de tipo cadena de texto (string)	getdate
Obtiene un array con información sobre la fecha y hora actual	isset
Establece la zona horaria	is_string

3.- Características básicas de PHP

Funciones

Las funciones son bloques de código que encontrándose en otro lugar se pueden ejecutar realizando lo que se conoce **llamada a la función**.

Ya se ha utilizado en clase una llamada a una función:

```
<?php
        phpinfo( );
?>
```

Y se ha visto cómo funciona la documentación para consultar funciones predefinidas por el lenguaje PHP.

3.- Características básicas de PHP

Funciones

Además de las funciones predefinidas por PHP también se pueden crear funciones propias.

NO es necesario definir las funciones antes de usarlas, simplemente han de estar en el **mismo script** que en el que se realiza la llamada ya sea directamente o en un archivo externo mediante una instrucción **include** o **require**.

Veamos un ejemplo de una función propia:

3.- Características básicas de PHP

Funciones

```
<?php
	$precio = 10;
	precio_con_iva();

	function precio_con_iva() {
		global $precio;
		$precio_iva = $precio * 1.21;
		echo "El precio con IVA es ". $precio_iva;
	}
?>
```

*El uso de **global** ya se explicó en clase y **NO es aconsejable este uso**.

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

Se puede pasar valores a las funciones mediante el uso de **argumentos**.

Los argumentos son una lista de variables separadas por comas.

No se indica el tipo de dato de la variable.

Si es necesario se puede hacer que la función **devuelva un valor**.

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

```
<?php
	function precio_con_iva($precio) {
		$precio_iva = $precio * 1.21;
		echo "El precio con IVA es ". $precio_iva;
	}

	$precio = 10;
	precio_con_iva($precio);
?>
```

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

```
<?php
	function precio_con_iva($precio) {
		$precio_iva = $precio * 1.21;
		return $precio_iva;
	}

	$precio = 10;
	$precio_final = precio_con_iva($precio);

	echo "El precio con IVA es ". $precio_final;
?>
```

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

```
<?php
	function precio_con_iva($precio) {
		$precio_iva = $precio * 1.21;
		return $precio_iva;
	}

	$precio = 10;
	echo "El precio con IVA es ". precio_con_iva($precio);
?>
```

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

En ocasiones puede interesar poner valores por defecto para los argumentos, de manera que si se realiza la llamada a la función no se indique valor, entonces se aplicará el valor por defecto.

Este comportamiento se vió con la función **date**.

```
<?php
        echo date("Y");
        echo date("Y", time());
?>
```

En ese caso los argumento con valor por defecto han de ir al final.

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

```
<?php
	function precio_con_iva($precio, $iva=0.21) {
		$precio = $precio * (1 + $iva);
		return $precio;
	}

	$precio = 10;
	$precio_iva = precio_con_iva($precio);
	echo "El precio con IVA es ".$precio_iva
?>
```

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

Los atributos se pueden pasar por **valor** o por **referencia**. En los ejemplos vistos hasta ahora se ha utilizado el **paso por valor**.

Esto significa que la variable original no va a cambiar de valor.

Se puede realizar el **paso por referencia** y entonces el valor de la variable original sí que puede cambiar.

No es recomendable realizar esta acción si no se tiene un dominio muy alto del lenguaje de programación.

3.- Características básicas de PHP

Funciones - Argumentos/parámetros

```
<?php
	function precio_con_iva(&$precio, $iva=0.21) {
		$precio = $precio * (1 + $iva);
	}

	$precio = 10;
	$precio_iva = precio_con_iva($precio);

	echo "El precio con IVA es ".$precio_iva
?>
```

Ejercicio

```
function calculo_numerico ($a, $b=5, $c) {
        $resultado = $a * $b * $c;
        return $resultado;
}
```

¿La definición de esta función es correcta?

Ejercicio

Inclusión de ficheros externos

Realizar los ejercicios 6 y 7 del boletín