<!-- PHP y HTML

Es un lenguaje interpretado y flexible que permite programar con gran rapidez

En el desarrollo web es habitual utilizar PHP incrustado dentro de ficheros HTML.

Al solicitar la página al servidor web el servidor procesa los bloques PHP

y los sustituye por su salida.

Los ficheros PHP deben tener una extensión .PHP

-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Ejemplo 1 cambio web</title>

</head>

<body>

<h1>

<!-- Abrir bloque de PHP -->

<?php

$saludo = "Hola Mundo";

echo $saludo;

?>

<!-- Cerrar bloque de PHP -->

</h1>

</body>

</html>

<?php

/\* Sintaxis básica

Cuando un fichero contiene solo PHP (puro) se recomienda no cerrar

la etiqueta del último bloque. Puede dar problemas si la respuesta del

servidor involucra varios ficheros.

Las sentencias en PHP acaban en ;\*/

phpinfo();

<?php

/\*

PHP es un lenguaje no fuertemente tipado:

no es necesario declarar el tipo de dato al declarar una variable.

Las variables no se declaran, se crean la primera vez que se les asigna un valor

El tipo de dato depende del valor con el que se inicialicen.

Ventajas: agiliza la escritura de programas.

Inconvenientes: puede dar lugar a un código de baja calidad

Nosotros usaremos el standar de codificación de Zend:

Variables con estilo "$camelCase".

Las concatenaciones con espacios a ambos lados del punto.

\*/

// La información se muestra con echo y print

$variable = 5 ;

echo 'Nuestra variable vale: ' . $variable . '<br>'; //Con puntos

print 'Nuestra variable vale: ' . $variable . '<br>';//Con puntos

echo 'Nuestra variable vale: ' , $variable , '<br><hr>'; //Con comas

//Tipos de datos básicos: entero, flotante, booleano, cadena

$anInteger = 3;

$aFloat = 3.1416;

$aString = "Hola ";

$anotherString = 'mundo';

$aBoolean = true; // recomendación Zend, en minúsculas.

$anotherBoolean = false;

print($aString . $anotherString . '<hr>');

echo gettype($aString) . '<hr>';

//podemos usar texto con comillas simples y dobles.

//más habitual las simples y dejamos las dobles para el html

//sin embargo las dobles permiten embeber variables en cadenas de texto:

echo "Un numero entero es $anInteger <br>";

echo 'Un numero entero es $anInteger <br>';

// ojo!! echo no muestra los "false".

echo 'Una variable booleana cierta se imprime como ' . $aBoolean . ' y la otra como ' . $anotherBoolean;

/\*

Variables no inicializadas

Al usar una variable no inicializada se genera el error E\_NOTICE

Pero no se interrumpe la ejecución del script

Si una variable no incializada aparece dentro de una expresión

-> La expresión se calcula tomando el valor por defecto para ese tipo de dato

\*/

$var1 = 100;

$var3 = 100 + $var2; //var2 no existe, se toma como 0

echo "<br> $var3 <br>";

isset($var1); //True si existe

empty($var1); //True si no existe o su valor es 0 o ""

<?php

//Definir una constate

define ("MAXSIZE", 100);

echo MAXSIZE;

//Otra forma

const CONSTANTE = "Hola Mundo";

echo CONSTANTE;

//La asignación se realiza por copia

//Las variables representan posiciones de memoria diferentes

$a = "Hola";

$b = $a; //asignación por copia

//Para definir una referencia

$var1 = 100;

$var2 = &$var1; //asignación por referencia

echo "$var2<br>";

$var2 = 300;

echo "$var1<br>";

<?php

// NÚMEROS

echo PHP\_INT\_SIZE.'<br>';

echo PHP\_INT\_MIN.'<br>';

echo PHP\_INT\_MAX.'<br>';

$a = 0b100; //binario

$a = 0100; //octal

$a = 0x100; //hexadecimal

$a = 3/2; //La división entre enteros no da problemas

echo $a.'<br>';

/\* La conversión entre integer y float es automática

Si se recibe float cuando se espera un integer -> se trunca

Si al realiza una operación con un entero tiene deciamales -> se convierte a float

También se pueden utilizar los operadores de conversión

\*/

$b = 7.5;

$a = (int) $b; //casting a int

//CADENAS

$var = "Paco";

$a = "Hola $var <br>"; //comillas mágicas las variables se sustituyen

$b = 'Hola $var';

echo $a;

echo $b;

//Otros tipos de datos

//array: colecciones de elementos

//object: para la programación orientada a objetos

//callable: tipo de datos para expresar las funciones de callback (funciones que se pasan a otras funciones)

//null: variable inicializada a null o variable no inicializada

//resource: representa recursos externos, como una conexión a una BD

<?php

/\*

\* Las estructuras de control son muy similares a Java

\*/

$var = 3;

if ( $var < 0 ) echo "Es menor que cero";

if ( $var > 0) {

echo "Es mayor que cero";

} else {

echo "Es menor o igual que cero";

}

$var = 3;

if ( $var == 1) {

echo "Es un uno";

} elseif ( $var == 2) {

echo "Es un dos";

} elseif ($var == 3) {

echo "Es un tres";

} else {

echo "No es uno, ni dos, ni tres";

}

// Si no hay un break al final de un case, la ejecución continua con el siguiente

$var = 3;

switch ( $var ) {

case 1:

echo "Es un uno";

break;

case 2:

echo "Es un dos";

break;

case 3:

echo "Es un tres";

break;

default:

echo "No es uno, ni dos, ni tres";

}

<?php

for ( $i = 0; i < 5; $i = $i + 1) {

echo "$i <br>";

}

$i = 0;

while ($i < 5) {

echo "$i <br>";

$i = $i + 1;

}

$i = 0;

do {

echo "$i <br>";

$i = $i + 1;

} while ($i < 5);

// Sentencia break: sirve para salir de un bucle

$i = 0;

while ($i < 5) {

echo "$i <br>";

$i++;

if ( $i == 2) {

break;

}

}

//Sentencia continue: fuerza una nueva iteración del bucle

for ( $i = 0; $i < 5; $i++) {

if ( $i == 2) {

continue;

}

echo "$i <br>";

}

//Es posible incluir otros ficheros utilizando las sentencias include y require

// include "mifichero.php"

// require "mifichero.php": para la ejecución si no encuentra el fichero

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>ejemplo</title>

</head>

</head>

<body>

<?php

define('SIZE', 10);

echo "<table border=1>";

$number = 1;

for ($row = 1; $row <= SIZE; $row++) {

if ($row % 2 == 0)

echo "<tr bgcolor = gainsboro>";

else

echo "<tr>";

for ($column = 1; $column <= SIZE; $column++) {

echo "<td>", $number, "</td>";

$number = $number + 1;

}

echo "</tr>";

}

echo "</table>";

?>

</body>

</html>

<?php

/\*

\* El tipo array unifica todas las estructuras de datos (vectores, listas o diccionario)

\* Los elementos de un array se identifican por una clave que puede ser un entero o una cadena

\* El orden está determinado por la inserción

\*/

//las cuatro sentencias son equivalentes

$frutas = ['manzana', 'naranja', 'uva'];

$frutas = array('manzana', 'naranja', 'uva');

$frutas = [0 => 'manzana', 1 => 'naranja', 2 => 'uva'];

$frutas = array(0 => 'manzana', 1 => 'naranja', 2 => 'uva');

//Podemos combinar tipos de datos

$array = [2, 'naranja', 3.1416];

echo "Me gustan la $array[2]";

//Añadir elementos

$frutas[] = 'manzana'; // si fruta está vacío, posición 0

$frutas[] = 'naranja'; // ahora posición 1

$frutas[2] = 'uva'; //ahora posición 2 porque lo ponemos, o cualquier otra.

//Quitar elementos

//unset($frutas[0]);

//Recorrer un array

echo 'Prueba foreach<br>';

foreach ($frutas as $fruta){

echo $fruta . '<br>';

}

echo 'Prueba foreach clave - valor<br>';

foreach ($frutas as $clave => $fruta){

echo $clave . ": " . $fruta . '<br>';

}

//Ejemplo de un array sin claves

$arr = array ( 10, 20, 30, 40);

print\_r($arr);

echo "<br>";

$arr[] = 5;

print\_r($arr);

echo "<br>";

$arr[10] = 6;

$arr[] = 5;

print\_r($arr);

echo "<br>";

/\*

\* Array asociactivo

\* Se usan las claves para referenciar cada valor

\*/

$alumno = array (

'id' => 5,

'nombre' => 'Manuel',

'apellido' => 'García López',

'edad' => 23

);

$alumno['sexo'] = "V"; //y podemos añadir elementos

print\_r($alumno);

//Otro ejemplo

$arr1 = [

0 => 444,

1 => 222,

2 => 333

];

print\_r($arr1);

echo "<br>"."pos 0: ".$arr1[0]."<br>";

$arr1[0] = 555;

print\_r($arr1);

echo "<br>";

$arr2 = array (

"1111A" => "Marta Lopez Vela",

"1112A" => "Luis Perez Tovar",

"1113A" => "Elena Molina Garzarán"

);

$arr2["1113A"] = "Elena Molina Gonzalvo";

//Recorrer el array

foreach ($arr2 as $nombre) {

echo "$nombre <br>";

}

foreach ($arr2 as $codigo => $nombre) {

echo "Código: $codigo Nombre: $nombre <br>";

}

//Modificar el array dentro de un bucle

foreach ($arr1 as &$cantidad) {

$cantidad = $cantidad \* 2;

}

print\_r($arr1);

//Arrays multidimensionales

$filas = [

0 => [11, 12],

1 => [21, 22],

3 => [31, 32]

];

echo "<br>Multi:".$filas[0][0];

print\_r($filas[0]);

<!DOCTYPE html>

<html><head>

<meta charset="UTF-8">

<title>Ejemplo sobre el uso de arrays</title>

</head><body>

<?php

echo "Array ordenado inicializado en la declaración <hr>";

$array1 = [5,"hola", 2.3, true, false, "elemento5"]; // equivalente

$array1 = array(5,"hola", 2.3, true, false, "elemento5");

//accedemos a los elementos del array con corchetes y su posición.

$elemento0 = $array1[2];

echo "El elemento 1 es $elemento0 y también $array1[0]";

echo "<hr> Podemos ver el contenido con var\_dump o print\_r";

echo "<br>";

var\_dump($array1);

echo "<br>";

print\_r($array1);

echo "<br><br>Usando la etiqueta html pre";

echo "<pre>";

var\_dump($array1);

echo "</pre>";

echo "<pre>";

print\_r($array1);

echo "</pre>";

//la podemos recorrer con el bucle foreach de dos maneras:

echo "<hr> Recorrido mediante bucle foreach simple <br>";

foreach ($array1 as $element){

echo $element . '<br>';

}

//foreach clave->elemento

echo "<hr> Recorrido mediante bucle foreach con posicion->elemento <br>";

foreach ($array1 as $position=>$element){

echo $position . ": " . $element . '<br>';

}

?>

</body></html>

<!DOCTYPE html>

<html><head>

<meta charset="UTF-8">

<title>Ejemplo</title>

</head><body>

<?php

//inicializamos

// $array1 = array();

// $array1 = [];

echo "Array ordenado inicializado elemento por elemento <hr>";

$array1[]="elemento 0 porque estába vacía";

$array1[]="elemento 1 supongo";

$array1[6]="elemento 6 porque lo digo yo";

$array1[200]="elemento 200 porque lo digo yo tb!";

$array1[]="elemento 201??? Lo veremos";

//volcamos la variable para estudiarla y depurar nuestro código.

//Con var\_dump

var\_dump($array1);

//O con print\_r

echo "<pre>";

print\_r($array1);

echo "</pre>";

//la podemos recorrer con el bucle foreach de dos maneras:

echo "<hr> Recorrido mediante bucle foreach simple <br>";

foreach ($array1 as $element){

echo $element . '<br>';

}

//foreach clave->elemento

echo "<hr> Recorrido mediante bucle foreach con posicion->elemento <br>";

foreach ($array1 as $position=>$element){

echo $position . ": " . $element . '<br>';

}

?>

</body></html>

<!DOCTYPE html>

<html><head>

<meta charset="UTF-8">

<title>Imprimir datos presonales</title>

</head><body>

<?php

//inicializamos

echo "Array ordenado inicializado en la declaración <hr>";

$array1 = array (

'id' => 5,

'nombre' => 'Manuel',

'apellido' => 'García López',

'edad' => 23

);

//foreach clave->elemento

echo "Recorrido mediante bucle foreach con posicion->elemento <br>";

foreach ($array1 as $position=>$element){

echo $position . ": " . $element . '<br>';

}

echo 'añadimos un elemento más y hacemos un volcado: <br>';

$array1['nacionalidad'] = 'Española';

var\_dump($array1);

?>

</body></html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Repaso de los puntos anteriores</title>

</head>

<body>

<?php

//4 formas de hacer lo mismo exactamente:

//forma1

$jugadoresEquipo[] = 'pepe';

$jugadoresEquipo[] = 'juan';

//forma2

$jugadoresEquipo = array('pepe', 'juan');

//forma3

$jugadoresEquipo = ['pepe', 'juan'];

//forma4

$jugadoresEquipo = [0 => 'pepe', 1 => 'juan'];

//inicializamos

echo "Arrays de dos dimensiones <hr>";

$filas = [

0 => [11, 12],

1 => [21, 22],

3 => [31, 32]

];

echo '<table border=1px>';

foreach ($filas as $fila) {

echo "<tr>";

foreach ($fila as $celda) {

echo "<td>$celda</td>";

}

echo "</tr>";

}

echo "</table>";

$filas = [

0 => [11, 12],

1 => [21, 22],

3 => [31, 32]

];

echo 'Array doble inicializado en la declaración <hr>';

$ciudades = array(

'Madrid' => array(

'poblacion' => 3000000,

'gentilicio' => 'Madrileño'

),

'Zaragoza' => array(

'poblacion' => 700000,

'gentilicio' => 'Zaragozano'

),

'Barcelona' => array(

'poblacion' => 3000000,

'gentilicio' => 'Barcelonés'

),

'Bilbao' => array(

'extension' => 500000,

'gentilicio' => 'Bilbaino'

),

);

echo "<br>Multi:" . $ciudades['Madrid']['gentilicio'] . "<br>";

echo "<br>Multi:" . $ciudades['Bilbao']['extension'] . "<br>";

echo "<pre>";

print\_r($ciudades);

echo "</pre>";

?>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Podemos hacer un var\_dump para investigar</h1>

<?php

echo "<pre>";

print\_r($\_SERVER);

echo "</pre>";

?>

<h1>Podemos usar foreach</h1>

<?php

foreach ($\_SERVER as $key => $value) {

echo "<li>$key: $value</li>";

}

?>

<h1>Y podemos acceder a aquello que buscamos</h1>

Fichero:

<?php

echo $\_SERVER['SCRIPT\_FILENAME'];

?>

</body>

</html>