1. PHP: ACRÓNIMO DE PERSONAL HOME PAGE

Lenguaje de propósito general. Propósito para desarrollo web.

Sintaxis similar a C / Java.

El código se almacena en archivo con extensión .php.

Los bloques de código se escriben entre **<?php** y **?>**, mientras que las sentencias se separan mediante "**;**".

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>PHP fácil</title>
</head>
<body>
<!-- Muestra una frase con HTML -->
Hola mundo<br>
<!-- Muestra una frase con PHP -->
<?php echo "Es muy fácil programar en PHP."; ?>
</body>
</html>
```

Si el código sólo va a contener código PHP y nada de html como, por ejemplo, cuando se codifiquen clases o interfaces, sólo se pondrá la etiqueta de apertura, para así indicar que es un archivo de php puro.

Código

Se tienen tres posibilidades a la hora de generar contenido en un documento PHP:

- **echo** expresión;
- **print** (expresión);
- **<?=** expresión ?>

Las que se van a utilizar son *echo* cuando se haga dentro de un bloque de instrucciones y *<?=* cuando sólo se vaya a mostrar el valor de una variable dentro de un fragmento HTML.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Echo y print</title>
</head>
<body>
<p><?php echo "Este texto se mostrará en la página web." ?></p>
<p><?= "Este texto se mostrará en la página web." ?></p>
<p><?php print("Este texto se mostrará en la página web.") ?></p>
</body>
</html>
```

Comentarios

Se pueden utilizar comentarios de una línea o de bloque:

```
<?php
// Este es un comentario de una sola línea
/*
        Este es
        un comentario
        que ocupa
        varias líneas
*/
?>
```

Errores

Si hay un error de ejecución, se produce un *Fatal Error*.

Fatal error: Uncaught Error: Call to undefined function *function()* in *PATH.*

Desde PHP 5 se lanzan como una excepción.

Variables

- No es necesario declararlas previamente.
- Comienzan por $, por ejemplo $nombre. Tras el $, el siguiente caracter debe ser una letra en minúscula (recomendación) o guión bajo _. Luego ya se pueden poner números.

- Son *case sensitive*: $var != $vAR
- No se declara su tipo, el tipado es dinámico. Se asigna en tiempo de ejecución dependiendo del valor asignado.
- Conveniente inicializarlas, si no, dan error.

```
<?php
$nombre = "Aitor";
$nombreCompleto = "Aitor Medrano";
$numero = 123;
$numero2 = 456;
$pi = 3.14;
$suerte = true;
$sinValor;

echo $sinValor;
?>
```

Tipos

Aunque a priori no hay tipos de datos, internamente PHP trabaja con cuatro tipos escalares: *boolean*, *integer*, *float* y *string* y cuatro tipos compuestos: *array*, *object*, *callable* e *iterable*. Existe un tipo especial para *null*.

Constantes

Son variables cuyo valor no varían. Existen dos posibilidades:

- define(NOMBRE, valor);
- const NOMBRE;

```
<?php
define("PI", 3.1416);
const IVA = 0.21;

echo PI, " ", IVA; // No se pone el símbolo dolar
?>
```

- Se declaran siempre en MAYÚSCULAS
- Hay un conjunto de constantes ya predefinidas, también conocidas como *magic constants*

Operadores

Ariméticos

Ejemplo	Nombre	Resultado
-$a	Negación	Opuesto de $a.
$a + $b	Suma	Suma de $a y $b.
$a - $b	Resta	Diferencia de $a y $b.
$a * $b	Multiplicación	Producto de $a y $b.
$a / $b	División	Cociente de $a y $b.
$a % $b	Módulo / Resto	Resto de $a dividido por $b.
$a ** $b	Potencia	Resultado de $a elevado a $b.

En el caso de **cadenas**, si se quiere concatenarlas, se utiliza el operador .:

```
<?php
$x = 33;
$y = 11;
$z = $x + $y;
echo "La suma de 33 y 11 es ".44."<br />";
echo "La suma de ".$x." y ".$y." es ".(33 + 11)."<br />";
echo "La suma de ".$x." y ".$y." es ".$z."<br />";
?>
```

Realmente, en vez de concatenar cadenas con variables, se puede imprimirlas directamente ya que se expanden automáticamente:

```
<?php
echo "La suma de $x y $y es $z <br />";
?>
```

En ocasiones, se necesita rodear el nombre de la variable entre llaves para poder unir más texto al resultado:

```
<?php
$color = "rojo";
echo "El plural de $color el ${color}s";
?>
```

Más adelante se estudiarán algunas funciones para el tratamiento de cadenas.

Comparación

Ejemplo	Nombre	Resultado
$a == $b	**Igual**	**true si $a es igual a $b tras de la conversión de tipos.**
$a === $b	**Idéntico, Comparación estricta**	**true si $a es igual a $b, y son del mismo tipo de dato.**
$a != $b, $a <> $b	**Diferente**	**true si $a no es igual a $b después de la conversión de tipos.**
$a !== $b	**No idéntico**	**true si $a no es igual a $b, o si no son del mismo tipo.**
$a < $b	**Menor que**	**true si $a es estrictamente menor que $b.**
$a > $b	**Mayor que**	**true si $a es estrictamente mayor que $b.**
$a <= $b	**Menor o igual que**	**true si $a es menor o igual que $b.**
$a >= $b	**Mayor o igual que**	**true si $a es mayor o igual que $b.**
$a <=> $b	**Nave espacial**	**Devuelve -1, 0 o 1 cuando $a es respectivamente menor, igual, o mayor que $b.**
$a ?? $b ?? $c	**Fusión de null**	**El primer operando de izquierda a derecha que exista y no sea null.** **null si no hay valores definidos y no son null.**

Lógicos

Ejemplo	Nombre	Resultado
$a and $b, $a && $b	*And* (y)	true si tanto $a como $b son true.
$a or $b, $a || $b	*Or* (o inclusivo)	true si cualquiera de $a o $b es true.
$a xor $b	*Xor* (o exclusivo)	true si $a o $b es true, pero no ambos.
!$a	*Not* (no)	true si $a no es true.

Asignación

Ejemplo	Nombre	Resultado
$a = $b	Asignación	Asigna a $a el valor de $b
$a += $b	Asignación de la suma	Le suma a $a el valor de $b. Equivalente a $a = $a + $b
$a -= $b	Asignación de la resta	Le resta a $a el valor de $b. Equivalente a $a = $a - $b
$a *= $b	Asignación del producto	Asigna a $a el producto de $a por $b. Equivalente a $a = $a * $b
$a /= $b	Asignación de la división	Asigna a $a el conciente de $a entre $b. Equivalente a $a = $a / $b
$a %= $b	Asignación del resto	Asigna a $a el resto de dividir $a entre $b. Equivalente a $a = $a % $b
$a .= $b	Concatenación	Concatena a $a la cadena $b. Equivalente a $a = $a . $b
$a++	Incremento	Incrementa $a en una unidad. Equivalente a $a = $a + 1
$a--	Decremento	Decrementa $a en una unidad. Equivalente a $a = $a - 1

Prioridad de los operadores

Recuerda la prioridad. Primero los paréntesis, luego la negación (!), productos/divisiones, sumas/restas, comparaciones, lógicos y por último se realiza la asignación.

Ejercicio

Si $a=5 y $b=4, averigua el valor de $c si $c = $a*2 > $b+5 && !($b<>4).

Realizar en PHP. ¿Qué resultado se obtiene?.

Formularios

Los datos se envían via URL con el formato var1=valor1 & var2=valor2… Por ejemplo: ejemplo.php?nombre=Bruce+apellido1=Wayne

Se divide en dos pasos:

1. Generar un formulario con **action='archivo.php' method='GET'**
2. En el archivo .php se leen los datos con **$_GET['nombreVar']**

Se va a separar siempre que se pueda el código HTML del de PHP. Por ejemplo, el formulario se coloca en saluda.html:

```
<form action="saluda.php" method="get">
   <p><label for="nombre">Nombre: </label>
   <input type="text" name="nombre" id="nombre"></p>
   <p><label for="apellido1">Primer apellido:</label>
   <input type="text" name="apellido1" id="apellido1"></p>
   <p><input type="submit" value="enviar"></p>
</form>
```

Y se recogen los datos en saluda.php:

```
<?php
$nombre = $_GET["nombre"];
$apellido1 = $_GET["apellido1"];
echo "Hola $nombre $apellido1";
?>
```

Si se quisiera realizar todo en un único archivo (*lo cual no es recomendable*), puede hacerse de la siguiente manera:

```
<form action="" method="get">
   <p><label for="nombre">Nombre: </label>
   <input type="text" name="nombre" id="nombre"></p>
   <p><label for="apellido1">Primer apellido:</label>
   <input type="text" name="apellido1" id="apellido1"></p>
   <input type="submit" value="enviar">
</form>
<p>
   <?php
   if(isset($_GET['nombre'])) {
      $nombre = $_GET["nombre"];
      $apellido1 = $_GET["apellido1"];
      echo "Hola $nombre $apellido1";
   }
   ?>
</p>
```

El trabajo con formularios se estudiará en profundidad más adelante, donde se comprobará que además de GET, podemos enviar los datos con POST.

Condiciones

La condición simple se realiza mediante la instrucción **if.** Entre paréntesis se pone la *condición* que se evalúa a true o false. Si no se ponen llaves, en vez de abrir un bloque, se ejecutará sólo la siguiente instrucción.

SIEMPRE LLAVES

Es recomendable poner llaves siempre, aunque en el momento de codificar sólo haya una única instrucción. De este modo, se queda preparado para añadir más contenido en el futuro sin provocar *bugs*.

```
<?php
$hora = 8; // La hora en formato de 24 horas
if ($hora === 8) {
   echo "Suena el despertador.";
}
echo "<br>";
if ($hora === 8)
   echo "Suena el despertador.";
?>
```

Las condiciones compuestas mediante ***if-else***:

```
<?php
$hora = 17; // La hora en formato de 24 horas
if ($hora <= 12) {
   echo "Son las " . $hora . " de la mañana";
} else {
   echo "Son las " . ($hora - 12) . " de la tarde";
}
?>
```

Las condiciones anidadas mediante varios ***if-else***:

```
<?php
$hora = 14; // La hora en formato de 24 horas
if ($hora === 8) {
   echo "Es la hora de desayunar.";
} else if ($hora === 14) {
   echo "Es la hora de la comida.";
} else if ($hora === 21) {
   echo "Es la hora de la cena.";
} else {
   echo "Ahora no toca comer.";
}
?>
```

La sentencia **switch** también permite trabajar con condiciones múltiples:

```
<?php
$hora = 14; // La hora en formato de 24 horas
switch ($hora) {
   case 9:
      echo "Es la hora de desayunar.";
      break;
   case 14:
      echo "Es la hora de la comida.";
      break;
   case 21:
      echo "Es la hora de la cena.";
      break;
   default:
      echo "Ahora no toca comer";
}
?>
```

NOTA: No olvidar el break

Un error muy común es olvidar la instrucción break tras cada caso. Si no se pone, ejecutará el siguiente caso automáticamente.

Finalmente, también se tiene el operador ternario condición "?":

```
<?php
$hora = 14;
$formato = ($hora > 12) ? 24 : 12;
echo "El formato es de $formato horas"
?>
```

Ejercicio

Buscar la sintaxis del operador ternario condición y cómo se usa.

```
<?php
$nombre = $_GET['nombre'] ?: "desconocido"
?>
```

¿Qué valos obtiene el siguiente código? ¿Está bien codificado?

Bucles

Mediante la instrucción **while**:

```
<?php
$i = 1;
while ($i <= 10) {
   echo "Línea " . $i;
   echo "<br>";
   $i++;
}
?>
```

Mediante la instrucción **do-while**:

```
<?php
do {
   $dado = rand(1, 6);
   // rand() devuelve un valor aleatorio
   echo "Tirando el dado... ";
   echo "ha salido un " . $dado . ".";
   echo "<br>";
} while ($dado != 5);
echo "¡Bien! Saco una ficha de casa.";
?>
```

Mediante la instrucción **for**:

```
<?php
// Bucle ascendente
for ($i = 1; $i <= 10; $i++) {
   echo "Línea " . $i;
   echo "<br>";
}
// Bucle descendente
for ($i = 10; $i >= 0; $i--) {
   echo "Línea " . $i;
   echo "<br>";
}
?>
```

Más adelante se estudiará el bucle **foreach** para recorrer arrays.

PHP, del mismo modo que Java y C, permite romper los bucles mediante la instrucción **break**. A su vez, **continue** permite saltar a la siguiente iteración.

SI PUEDES, EVITA BREAK Y CONTINUE

Opcional: Uso de variables *flag* para controlar la salida de los bucles. Por ejemplo:

```
<?php
$salir = false;
for ($i = 1; $i <= 10 && !$salir; $i++) {
  if ($i === 5) {
    echo "Salgo cuando i=5";
    $salir = true;
  }
}
?>
```