PHP

- PHP básico

Ing. Magda Perozo

Lenguaje PHP básico

1. Sintaxis básica
2. Tipos de datos
3. Variables
4. Constantes
5. Expresiones y operadores
6. Estructuras de control
7. Funciones
8. Tablas
9. Bibliotecas de funciones

Sintaxis básica

- PHP es sensible a las mayúsculas
- ¿Cómo se incrusta en la página web?

  `<?PHP ... ?>`

  recomendado, siempre disponible

  `<?= expresión ?>`

  equivale a <? echo expresión ?>
- Las instrucciones se separan con un ; como en C.
- La marca final ?> implica un ;
- Comentarios: como en C, /* … */ y //

Sintaxis básica

- Para imprimir: **echo** y **print**

  echo: muestra una o más cadenas
  echo cadena1 [, cadena2…]; // no es una función

```
echo "hola que tal";
echo "Hola ", "que tal";
```

  print: muestra una cadena
  print cadena; // no es una función

```
print "Hola chamo";
print  "Hola " . "Chamo";
```

Sintaxis básica

- Ejemplo:

```
<HTML>
<HEAD>
<TITLE>Mi primer programa en PHP</TITLE>
</HEAD>

<BODY>
Esto es html<br>
<?PHP
 print "Esto es PHP"."<br>";
 echo "Esto es PHP", "<br>";
?>

</BODY>
</HTML>
```

Sintaxis básica

- Imprimir html en php

Código PHP

```
print (“<P>Párrafo 1</P>”);
print (“<P>Párrafo 2</P>”);
```

Código HTML

```
<P>Párrafo 1</P><P>Párrafo 2</P>
```

Salida

```
Párrafo 1

Párrafo 2
```

Sintaxis básica

- **Ejercicio 1: programa que muestra un mensaje**
  - Ilustra cómo incrustar código PHP en un documento HTML y cómo imprimir desde PHP

Sintaxis básica

- Inclusión de ficheros externos:
  - **include**()
  - **require**()
- Ambos incluyen y evalúan el fichero especificado
- Diferencia: en caso de error include() produce un warning y require() un error fatal
- Se usará require() si al producirse un error debe interrumpirse la carga de la página
  - **Include_once**()
  - **Require_once**()

  Trabajan igual que las anteriores, pero previene no incluir mas de una vez el mismo código.

  Utiliza mas recursos.

- Ejemplo:

Sintaxis básica

```
<HTML>
<HEAD>
   <TITLE>Título</TITLE>
<?PHP
// Incluir bibliotecas de funciones
   require ("$libdir/conecta.php");
   require ("$libdir/fecha.php");
   require ("$libdir/cadena.php");
   require ("$libdir/globals.php");
?>
</HEAD>
<BODY>
<?PHP
   include ("cabecera.html");
?>
// Código HTML + PHP
. . .
<?PHP
   include ("pie.html");
?>
</BODY>
</HTML>
```

Tipos de datos

- PHP soporta 8 **tipos de datos primitivos**:
  - boolean, integer, double, string
  - array, object
  - resource, NULL
- El tipo de una variable no se suele especificar. Se decide en tiempo de ejecución en función del contexto y puede variar
- Funciones de interés:
  - La función gettype() devuelve el tipo de una variable
  - Las funciones is_*type* comprueban si una variable es de un tipo dado:

    is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar(), is_string()
  - La función var_dump() muestra el tipo y el valor de una variable. Es especialmente interesante con los arrays

Tipos de datos

- Tipo **integer** (números enteros)
  - 27, -5, 0
- Tipo **double** (números reales)
  - 1.234, -5.33
- Tipo **boolean** (lógico)
  - Valores: *true*, *false* (insensibles a las mayúsculas)
  - El 0 y la cadena vacía tienen valor *false*

Tipos de datos

- Tipo string:
  - Las cadenas se encierran entre comillas simples o dobles:
    - ‘simples’: admite los caracteres de escape \’ (comilla simple) y \\ (barra). Las variables **NO** se expanden
    - “dobles”: admite más caracteres de escape, como \n, \r, \t, \\, \$, \”. Los nombres de variables **SÍ** se expanden

```
$a = 9;
print ‘a vale $a’; // muestra a vale $a
print “a vale $a”; // muestra a vale 9
```

  - Acceso a un carácter de la cadena:
    - La forma es $inicial = $nombre{0};

Variables

- Las variables siempre van precedidas de un $
- El nombre es sensible a las mayúsculas
- Comienzan por letra o subrayado, seguido de letras, números o subrayado
- Ámbito: globales al fichero (excepto funciones) o locales a una función
- Variables predefinidas:

  $GLOBALS, $_SERVER, $_GET, $_POST, $_COOKIES, $_FILES, $_ENV, $_REQUEST, $_SESSION
- Ejemplo:

```
$valor = 5;
print "El valor es: " . $valor . "\n";
print "El valor es: $valor<br>"; // ojo: comillas dobles

Resultado:
   El valor es: 5
```

Constantes

- Definición de constantes:

```
define ("CONSTANTE", "hola");
print CONSTANTE;
```

- No llevan $ delante
- Sólo se pueden definir constantes de los tipos escalares (boolean, integer, double, string)

Expresiones y operadores

- Operadores aritméticos:
  +, -, *, /, %, ++, --
- Operador de asignación:
  =
  operadores combinados: .=, +=, etc
  $a = 3; $a += 5; → a vale 8
  $b = “hola ”; $b .= “chamo”; → b vale “hola chamo”
  → Equivale a $b = $b . “chamo”;
- Operadores de comparación:
  ==, !=, <, >, <=, >= y otros
- Operador de control de error: @. Antepuesto a una expresión, evita cualquier mensaje de error que pueda ser generado por la expresión
- Operadores lógicos:
  and (&&), or (||), !, xor
  and/&& y or/|| tienen diferentes prioridades
- Operadores de cadena:
  concatenación: . (punto)
  asignación con concatenación: .=

Expresiones y operadores

- Precedencia de operadores (de mayor a menor):

```
++, --
*, /, %
+,-
<, <=, >, >=
==, !=
&&
||
```

Estructuras de control

- if-else
- while
- do .. while
- for
- foreach
- switch

Estructuras de control

- if-else

```
if (expresión1)
    sentencia 1
else if (expresión2)
    sentencia 2
...
else if (expresión n)
    sentencia n
else
    sentencia n+1
```

- Mismo comportamiento que en C
- Las sentencias compuestas se encierran entre llaves
- elseif puede ir todo junto

Estructuras de control

- while

```
while (expresión)
   sentencia
```

- Mismo comportamiento que en C

Estructuras de control

- for

```
for (expresión1; expresión2; expresión3)
   sentencia
```

- Mismo comportamiento que en C

Estructuras de control

- switch

```
switch (expresión)
{
   case valor 1:
         sentencia 1
         break;
   case valor 2:
         sentencia 2
         break;
   …
   case valor n:
         sentencia n
          break;
   default
         sentencia n+1
}
```

- Mismo comportamiento que en C, sólo que la expresión del case puede ser integer, float o string

Estructuras de control

- **Ejercicio 2: programa que calcula una tabla de multiplicar**
  - Ilustra cómo manejar variables y cómo usar bucles

Funciones

- Ejemplo:

```
function suma ($x, $y)
{
   $s = $x + $y;
   return $s;
}

$a=1;
$b=2;
$c=suma ($a, $b);
print $c;
```

Funciones

- Por defecto los parámetros se pasan por valor
- Paso por referencia:

```
function incrementa (&$a)
{
   $a = $a + 1;
}

$a=1;
incrementa ($a);
print $a; // Muestra un 2
```

Funciones

- Argumentos por defecto

```
function muestranombre ($titulo = "Sr.")
{
    print "Estimado $titulo:\n";
}
muestranombre ();
muestranombre ("Prof.");
```

- Salida:

```
Estimado Sr.:
Estimado Prof.:
```

Funciones

- Los argumentos con valores por defecto deben ser siempre los últimos:

```
function muestranombre ($nombre, $titulo= "Sr.")
{
    print "Estimado $titulo $nombre:\n";
}
muestranombre (“Fernández”);
muestranombre (“Fernández”, "Prof.");
```

- Salida:

```
Estimado Sr. Fernández:
Estimado Prof. Fernández:
```

Tablas

- Sintaxis:

```
array ([clave =>] valor, ...)
```

- La clave es una cadena o un entero no negativo. El valor puede ser de cualquier tipo válido en PHP, incluyendo otro array
- Ejemplos:

```
$color = array ('rojo'=>101, 'verde'=>51, 'azul'=>255);
$medidas = array (10, 25, 15);
```

- Acceso:

```
$color['rojo'] // No olvidar las comillas
$medidas[0]
```

- El primer elemento es el 0

Tablas

- La estructura de control **foreach** permite iterar sobre arrays
- Sintaxis:

```
foreach (expresión_array as $valor)
   sentencia
foreach (expresión_array as $clave => $valor)
   sentencia
```

- Ejemplos:

```
foreach ($color as $valor)
   print "Valor: $valor<BR>\n";
foreach ($color as $clave => $valor)
   print "Clave: $clave; Valor: $valor<BR>\n";
```

- Salida:

```
Valor: 101
Valor: 51
Valor: 255
Clave: rojo; Valor: 101
Clave: verde; Valor: 51
Clave: azul; Valor: 255
```

Bibliotecas de funciones

- Existen muchas bibliotecas de funciones en PHP
- Algunos ejemplos:
  - Funciones de manipulación de cadenas
  - Funciones de fecha y hora
  - Funciones de arrays
  - Funciones de ficheros
  - Funciones matemáticas
  - Funciones de bases de datos
  - Funciones de red
- Algunas bibliotecas requieren la instalación de componentes adicionales
- Todas las funciones de biblioteca están comentadas en la documentación de PHP

Bibliotecas de funciones

- **Ejemplo 3: programa que muestra la fecha actual**
  - Ilustra cómo usar comentarios, tablas y funciones (propias y de biblioteca). También cómo usar el manual de PHP