GESTION DE DATOS

- Tipos de datos
- Variables
- Constantes
- Operadores

Tipos de datos

- **Booleanos**
- **Números enteros**
- **Números de coma flotante**
- **Cadenas de caracteres**
- **Matrices o arrays**
- **NULL**

```
<?php
 $booleano=FALSE;
 $entero=123;
 $real=123.45;
 $cadena="Saludines<br>" ;
 $dias  = array( 'Lunes','Martes','Jueves' );
 $LaNada ;
?>
```

PHP es muy flexible en la manejo de los tipos de datos:

- ✓ No obliga a definir qué tipo de datos contendrán las variables

  De hecho pueden contener datos de diferentes tipo a lo largo del script

- ✓ Permite combinar datos de distintos tipos en las operaciones

  ¿Se pueden sumar peras con manzanas?

  No, unas reglas convierten los operandos a un tipo común.

Tipos de datos

- Reglas automáticas de conversión de tipos de datos

  - En **operaciones lógicas**, los datos NULL, 0, '0' y '' se consideran FALSE. Cualquier otro dato se considera TRUE (incluida la cadena 'FALSE').

  - En **operaciones aritméticas** las cadenas se intentan leer como números y si no se puede se convierten en 0, TRUE se convierte en 1, y FALSE se convierte en 0.

  - En operaciones de comparación, si un operando es un número, el otro también se convertirá en un número. Sólo si ambos operandos son cadenas se compararán como cadenas de caracteres.

  - En operaciones de cadenas de caracteres, NULL y FALSE se convierten en '', y TRUE se convierte en '1'.

Tipos de datos

- **Operaciones aritméticas con enteros**:
  si resultado está en el rango de los enteros, devuelve un entero.
  si resultado sale del rango de los enteros, devuelve un float

```
$x = 12  +  10  ;
$x  = 12  + 2147483659 ;
```

- **Operaciones aritméticas de enteros con float:**
  el resultado es float

```
$x = 12  +  10.125  ;
```

- **Operaciones aritméticas con cadenas**
  Sí cadena comienza con dígitos, los extrae y opera con ello como entero
  Sí cadena no comienza con dígitos, toma el valor 0 (cero)

```
$x = 12  + ' 1eval '  ;
$x = 12  + ' eval1 ' ;
```

- **Operaciones aritméticas con booleanos:**
  TRUE lo toma como 1 y FALSE como 0 (cero)

```
$x = 12  +  ( 5 > 3 )  ;
```

- **Operaciones de concatenación** devuelven tipo string
  Los números (enteros y reales) se convierten en cadenas
  Valor booleano TRUE se convierte en cadena '1' y FALSE en cadena vacia

```
$x = 5  .  2.3  .  ( 5 > 3 ) ;
```

- **Operaciones de comparación:**
  Números: El 0 lo toma como FALSE,
  otros valores lo toma como TRUE
  Cadenas: La cadena vacía (sin contenido) la toma como FALSE,
  el resto como TRUE

```
$x  = '      ';
IF  ( 5  == ' 5ev '  and  $x != FALSE )
            echo ' Hola ' ;
```

Tipos de datos

- La función **var_dump** sirve para obtener información sobre un dato

```php
<?php
 $booleano=FALSE;   $entero=123;    $real=123.45;   $LaNada ;
 $cadena="Saludines<br>" ;   $dias  = array( 'Lunes','Martes','Jueves' );
 echo  var_dump($booleano) , '<br>' ,  var_dump($real) , '<br>' ;
 echo  var_dump($dias[0]) , '<br>' , var_dump( $cadena + $entero ) , '<br>' ;
 echo  var_dump($LaNada) , '<br>' ,  var_dump( TRUE and (5<3) ) , '<br>' ;
 echo  var_dump($dias) , '<br>' ;
?>
```

```
bool(false)
float(123.45)
string(5) "Lunes"
int(123)
NULL
bool(false)
array(3) { [0]=> string(5) "Lunes" [1]=> string(6) "Martes" [2]=> string(6) "Jueves" }
```

Tipos de datos

- La función **gettype** devuelve una string con el tipo del dato recibido como argumento

```
<?php
 $booleano=FALSE;   $entero=123;    $real=123.45;   $LaNada ;
 $cadena="Saludines<br>" ;   $dias  = array( 'Lunes','Martes','Jueves' );
 echo  gettype($booleano);  //  muestra  boolean
 echo gettype( $real );   //  muestra double
 echo gettype( $dias );   //  muestra array
 if  ( gettype( $LaNada )  == 'NULL'    ) {
       echo ' la variable  $cadena es de tipo : ' . gettype( $cadena ) ;
       echo  ' y  $cadena + $real  es de tipo : ' . gettype( $cadena+$real ) ;
       $OtraVar = gettype( $cadena+$real );
      echo ' la variable  $OtraVar es de tipo : ' . gettype( $OtraVar ) ;
    }
?>
```

Tipos de datos

- **Forzado explicito de tipos** (Casting)

  Establece el tipo para un dato sin tener en cuenta su contenido

  - **(boolean)**
  - **(integer)**
  - **(float) (real) (double)**
  - **(string)**
  - **(array)**
  - **(unset)** convierte al tipo NULL

```
<?php
 $n1=123;  // por su contenido  $n1 es de tipo entero
 $n2= (float) $n1;     $n3= (string) $n1;
 $n4= (boolean) $n1;
 $n5= (array)  $n1;
 $n6= (unset) $n1;
 echo  var_dump($n1,$n2,$n3,$n4), '<br>';
 echo  var_dump ($n5) , '<br>';
 echo  var_dump( $n6);
?>
```

```
int(123) float(123) string(3) "123" bool(true)
array(1) { [0]=> int(123) }
NULL
```

Tipos de datos

- Igual que 1/3 no puede representarse con exactitud en decimal, otros números como 0.1 no pueden representarse con exactitud en binario.

```php
<?php
 var_dump( (integer)(0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1) );
  echo "<br>";
  var_dump( (integer)(0.1*10) ) ;
  echo "<br>";
  var_dump( (0.1*10) == (0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1) );
?>
```

```
int(0)
int(1)
bool(false)
```

Tipos de datos

❑ Forzado de tipo con la función settype

Con **settype** podemos cambiar el tipo de variables ya definidas

settype(variable , tipo)

Valores admitidos para tipo: 'integer'
'float'
'string'
'array'
'boolean'

```
<?php
  $v1=12;
  settype( $v1 , 'array' );
  echo  ' la variable  $v1 es de tipo : ' . gettype( $v1 ) ;
  echo "<br>";
  echo  var_dump($v1) ;
  echo "<br>";
?>
```

```
la variable   $v1  es de tipo : array
array(1) { [0]=> int(12) }
```

Tipos de datos

- Los enteros pueden expresarse en:
  - Decimal
  - Hexadecimal: precedidos de **0x**
  - Octal: precedidos de **0**

```
<?php
 echo  var_dump( 0123 ) .  "<br>“ ;
 echo  var_dump( 0x123) , "<br>“  ;
 echo  var_dump( (integer) '0123' );
?>
```

```
int(83)
int(291)
int(123)
```

- Los float pueden expresarse en:
  - Notación de punto: **1.23**
  - Notación científica: **123e-2**

```
<?php
 echo  var_dump( 1.23 ) .  "<br>“ ;
 echo  var_dump( 123e-2) , "<br>“  ;
?>
```

```
float(1.23)
float(1.23)
```

Variables

- Su nombre siempre debe empezar con el signo $ seguido de una letra o un signo de subrayado y de cualquier combinación de letras, números o signos de subrayado (no espacios).
- PHP es sensible al uso de mayúsculas/minúsculas en el nombre de las variables.

- Ámbito de variables:

  Superglobal: Reconocibles en todo el script, dentro y fuera de funciones.

  Global: Se reconocen en todo el script, excepto dentro de las funciones.

  Local: Sólo se reconocen dentro de la función en la que son definidas.

  Estático: Igual que local, pero mantienen su valor de una ejecución de la función a la siguiente.

Variables

- Los valores de una variable definida en cualquier parte de un script –siempre que no sea dentro de una función– son visibles en cualquier otra parte de ese script, excepto dentro de las funciones contenidas en el script
- Si una variable es definida dentro de una función sólo podrá ser utilizada dentro de esa función.
- Si en una función aludimos a una variable externa a ella, PHP considerará esa llamada como si la variable tuviera valor cero (al ser tratada como número) o una cadena vacía

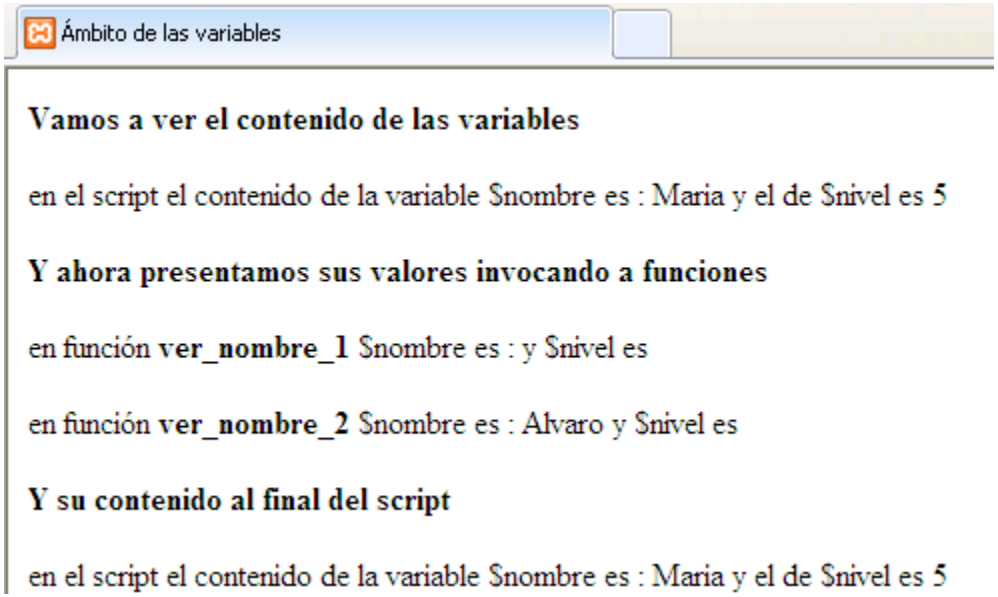
  Igual ocurriría si desde fuera de una función aludimos a una variable definida en ella.
- Si definimos dos variables con el mismo nombre, una dentro de una función y otra fuera, PHP las considerará distintas. La función utilizará –cuando sea ejecutada– sus propios valores sin que sus resultados modifiquen la variable externa.

Variables

```
<HTML>  <HEAD><TITLE>Ámbito de las variables </TITLE></HEAD>
<BODY>
 <?php
   $nivel =5; $nombre="Maria";  // Definimos variables  globales en el script
 ?>
                          <!-- esto es HTML, hemos cerrado el script -->
                      <b>Vamos a ver el contenido de las variables</b><br>
 <?php
  echo "<br> en el script  contenido de \$nombre es : $nombre  y el de \$nivel es $nivel";
 ?>
              <b><br><br>Y ahora presentamos sus valores invocando a funciones</b><br>
<?php
  function ver_nombre_1()
  { echo "<br> en función <b>ver_nombre_1 </b> \$nombre es : $nombre y  \$nivel es $nivel <br>"; }

  function ver_nombre_2()
  { $nombre = 'Alvaro';
    echo "<br> en función <b>ver_nombre_2</b>  \$nombre es : $nombre y  \$nivel es $nivel  <br>";}

  ver_nombre_1();
  ver_nombre_2();
 ?>
                      <br><b>Y su contenido al final del script </b><br>
 <?php
   echo "<br> en el script  el contenido de la variable \$nombre es : $nombre  y el de \$nivel es $nivel";
 ?>
</BODY>
</HTML>
```

Variables

```
<HTML>  <HEAD><TITLE>Ámbito de las variables </TITLE></HEAD>
<BODY>
  <?php    $nivel =5; $nombre="Maria";  // Definimos variables  globales en el script    ?>
                        <!-- esto es HTML, hemos cerrado el script -->
                         <b>Vamos a ver el contenido de las variables</b><br>
  <?php   echo "<br> en el script  contenido de \$nombre es : $nombre  y el de \$nivel es $nivel";  ?>   <b><br><br>Y ahora presentamos sus valores invocando
      a funciones</b><br>
  <?php
  function ver_nombre_1()      {echo "<br> en función <b>ver_nombre_1 </b> \$nombre es : $nombre y  \$nivel es $nivel <br>"; }

  function ver_nombre_2()     { $nombre = 'Alvaro';    echo "<br> en función <b>ver_nombre_2</b>  \$nombre es : $nombre y  \$nivel es $nivel  <br>";}

  ver_nombre_1();
  ver_nombre_2();
    ?>
                    <br><b>Y su contenido al final del script </b><br>
  <?php       echo "<br> en el script  el contenido de la variable \$nombre es : $nombre  y el de \$nivel es $nivel";    ?>
</BODY>
</HTML>
```



Variables

- ¿ Como acceder en las funciones a variables globales del script ?

  - Declarando las variable en la función como global

    global $var1 , $var2

```
<?php
 $a = 1; $b = 2;
  function Suma()
    { global  $a, $b;
       $b = $a + $b;
     }
Suma();
echo $b;
?>
```

  - Usando el array **$GLOBALS**, que es un array asociativo con el nombre de la variable global como clave

```
<?php
$a = 1;  $b = 2;
 function Suma()
  { $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
  }
Suma();
echo $b;
?>
```

Variables

Variables Estáticas

- Una variable estática existe sólo en el ámbito local de la función, pero no pierde su valor cuando la ejecución del programa abandona este ámbito

static $var1 , $var2

- Sólo se inician la primera vez que se llama la función

- Sólo se les puede asignar inicialmente un dato (no una expresión ni una variable)

```
function XXX()
{
  static  $x =1;          // asignación válida
  static  $w =1 + 10 ; // asignación inválida
   ..........
 }
```

```
<?php
function test()
{ static  $count = 0 ;
  $count++;
  echo $count;
  if ($count < 10)
   {   test();    }
  $count--;
}

test();
?>
```

Variables

- Variables Superglobales

A partir de PHP 4.1.0, se dispone de un conjunto de matrices predefinidas que contienen variables del servidor web, el entorno y entradas del usuario.

Estas matrices son un poco especiales porque son automáticamente globales.

Por esta razón, son conocidas a menudo como "superglobales".

$_GET

$_POST

$_FILES: <input type='file'/>

$_SESSION

$_COOKIE

$_REQUEST: GET, POST y COOKIE

- Directiva **request_order** ='GP' del php.ini

$_SERVER (HTTP_USER_AGENT , REMOTE_ADDR , …)

$_ENV

- Directiva **variables_order** del php.ini sin la E hace que este vacia

$_GLOBALS

Variables

- Aunque haya varios scripts dentro de un mismo archivo se consideran uno solo; las globales de uno son reconocidas en el otro

```
<html>
   <head>
      <title>&Aacute;mbito global de las variables</title>
   </head>
   <body>
      <?php
         $a=10;
      ?>
      <script type='text/javascript'>
         var i;
         for (i=0;i<<?php echo $a;?>;i++){
            document.write(i+'<br/>');
         }
      </script>
   </body>
</html>
```

Variables

- **Variables variables**

  Son nombres de variables que se pueden definir y usar dinámicamente

  Una **variable variable** toma el valor de una variable y lo trata como el nombre de una variable.

```
<?php
 $a = 'hola';        //  define una variable de nombre a
$$a = 'mundo';    //  define una variable de nombre hola
 echo "$a ${$a} ";  // muestra contenido de variable a y variable hola
echo "$a $hola";  //   igual que linea anterior
?>
```

- Para usar variables variables con matrices, hay que resolver un problema de ambigüedad.

Si se escribe **$$a[1]** el intérprete necesita saber si nos referimos a utilizar $a[1] como una variable, o bien utilizar $$a como variable y el índice [1] como índice de dicha variable.

La sintaxis para resolver esta ambigüedad es: **${$a[1]}** para el primer caso y **${$a}[1]** para el segundo.

CONSTANTES

- No pueden cambiar de valor durante la ejecución del script.
  - –Se declaran con define define('PI',3.14);
  - –No van precedidas de $ y, por convenio, se escriben completamente en mayúsculas.

- PHP dispone de una serie de constantes predefinidas cuyo valor depende de dónde se utilicen (no son verdaderas constantes, aunque son llamadas constantes mágicas)

  __FILE__

  __DIR__

OPERADORES

++ y --

(integer), (float), (string), (boolean) y (array)

!

*, / y %

+, - y .

<, <=, > y >=

==, !=, === y !==

&&

||

=, +=, -=, *=, /=, .= y %=

and

xor

or

OPERADORES

- Los operadores unitarios (**++** y **--**) pueden usarse en modo prefijo y sufijo.
- **%** convierte los operandos a enteros antes de efectuar el cociente y utiliza para el resultado el signo del numerador.
- **&&** y **||** tienen precedencia sobre los operadores de asignación, pero **and** y **or** no.

```php
<?php
    $premisa1=TRUE;
    $premisa2=FALSE;
    $resultado=$premisa1 and  $premisa2; //= precede a and
    echo (integer)$resultado;
    echo '<br />';
    $resultado=$premisa1 &&  $premisa2; //&& precede a =
    echo (integer)$resultado;
?>
```

- El operador **===** (idéntico) solo devuelve TRUE si los datos comparados tienen el mismo valor y son del mismo tipo (float, integer, boolean, …). El operador **!==** devuelve TRUE si los datos comparados no tienen el mismo valor o no son del mismo tipo.

```php
<?php
        $a=0;
        $b='0';
        if($a==$b){
                echo 'Son iguales';
        }else{
                echo 'Son distintos';
        }
        echo '<br />';
        if($a===$b){
                echo 'Son id&eacute;nticos';
        }else{
                echo 'No son id&eacute;nticos';
        }
?>
```