Teoría 3

Constantes

```php
<?php
/* CONSTANTES:
 *     - Es MUY aconsejable, aunque no obligatorio, definirlas siempre en mayúscula,
 *       para diferenciarlas de variables.
 *     - Por lo tanto se usan en mayúscula ya que tienen en cuenta la diferencia.
 *     - Pueden ser de cualquiera de los tipos primitivos (boolean, integer, float y string).
 *     - Dentro de un string entre "" no se evaluan.
 *     - No se puede redefinir su valor.
 *     - Son "Globales", es decir que su valor puede ser utilizado en todos los niveles
 *       internos de un módulo.
 */

define('MESES',12);
define('DIAS',31);
define('SALUDO','Hola');

echo 'MESES: ' . MESES . '<br>';
echo 'DIAS : ' . DIAS . '<br>';
echo 'SALUDO: ' . SALUDO . '<br>';

echo 'Meses: ' . Meses . '<br>';  // case-sensitive


//MESES = 11;  // Observar que no se puede hacer ya que es una constante ! !
?>
```

Constantes mágicas

```php
<?php
/* Constantes "mágicas" (no son realmente constantes, pero no se les puede cambiar el valor):
 *    Son un conjunto de valores predefinidos de PHP, que pueden ser de utilidad.
 *    Son cinco; entre ellas:
 *     __LINE__         El número de línea actual.
 *     __FILE__         El nombre completo del archivo que se está ejecutando.
 *     __FUNCTION__ El nombre de la función actual.
 *
```

```
 *     http://www.php.net/manual/es/language.constants.predefined.php
 */

echo __LINE__ . '<br>';
echo __FILE__ . '<br>';
echo __FUNCTION__ . '<br>';
echo __CLASS__ . '<br>';
echo __METHOD__ . '<br>';
echo __LINE__ . '<br>';
?>
```

Constantes predefinidas

```
<?php
/* Constantes predefinidas
 * (mas de 100 constantes definidas y disponibles en PHP)
 * http://www.php.net/manual/es/reserved.constants.php
 */

echo PHP_VERSION . '<br>';
echo PHP_OS . '<br>';

// print_r(get_defined_constants()); // Para ver todas las constantes
definidas
?>
```

Alcance de variables

```
<?php
/* Alcance de variables en PHP:
    - Sin global las variables NO pasan a las funciones, por lo tanto son
otras
      internas con igual nombre que se pierden cuando termina la funcion.
    - Con global son las variables externas, por lo tanto tienen el valor y
      ademas se pueden modificar internamente y mantendrán su valor luego de
      terminar la funcion.
    - NO USAR GLOBAL, (ni tampoco sin global), es decir USAR SIEMPRE
PARAMETROS.
    - Los únicos valores globales que no dan problemas son las constantes.
      ¿Por qué?
*/
    /**
     * Procedimiento de prueba SIN global
     */
    function modifica1(){
        echo '1.1 ' . $textoejemplo . '<br>';
```

```
        $textoejemplo = 'Modificado por el codigo de modifica1';
        echo '1.2 ' . $textoejemplo . '<br>';
    }
    /**
     * Procedimiento de prueba CON global
     */
    function modifica2(){
        global $textoejemplo;
        echo '2.1 ' . $textoejemplo . '<br>';
        $textoejemplo = 'Modificado por el codigo de modifica2';
        echo '2.2 ' . $textoejemplo . '<br>';
    }

    $textoejemplo = 'Valor original';
    echo 'a  ' . $textoejemplo . '<br>';
    modifica1();
    echo 'b  ' . $textoejemplo . '<br>';
    modifica2();
    echo 'c  ' . $textoejemplo . '<br>';
?>
```

Tipos de variables

PHP provee ocho tipos de datos, divididos en tres categorías:

Escalares	boolean, integer, float, string
Compuestos	array, object
Especiales	NULL, resource

El tipo de una variable queda determinado por el valor que contiene, y puede cambiar durante la existencia de la variable.

Los literales TRUE y FALSE representan los valores booleanos “verdadero” y “falso”, respectivamente. Ambas son insensibles a mayúsculas y minúsculas, es decir que TRUE, true, True o tRuE significan “verdadero”.

```
<?php
$bandera = TRUE;
$texto = 'string';
$cantidad = 7;
$importe = 120.30;
//$inexistente = NULL;

echo 'GETTYPE:<br><br>';
echo gettype($bandera) . '<br>';
echo gettype($texto) . '<br>';
echo gettype($cantidad) . '<br>';
echo gettype($importe) . '<br>';
echo gettype($inexistente) . '<br>';
```

```
echo '<br><br>VAR_DUMP:<br><br>';
echo var_dump($bandera) . '<br>';
echo var_dump($texto) . '<br>';
echo var_dump($cantidad) . '<br>';
echo var_dump($importe) . '<br>';
echo var_dump($inexistente);
?>
```

Conversiones entre tipos

Se puede convertir un valor de un tipo a otro, prefijándolo con el nombre del tipo encerrado entre paréntesis. Por ejemplo:

```
<?php
/*
 * Conversiones a boolean:
 * - a FALSE <-  FALSE, 0, 0.0, "", "0", NULL
 * - a TRUE  <-  TRUE, cualquier otro valor
 *
 * Conversiones a integer:
 * - 0 <- FALSE, 0.0, '0', '0.0', 'aaaa', 'a1aaa'
 * - 1 <- TRUE, 1.0, '1', '1aaaa'
 *
 * Conversiones a string:
 * - ''  <- FALSE
 * - '1' <- TRUE, 1, 1.0
 * - '1.01' <- 01.01
 */

var_dump((bool) 'false');
var_dump((int) '1aaaa');
var_dump((string) 01.01);
var_dump((real) '01.01');

$cinco = 5;
var_dump($cinco);
$cinco = (string) $cinco;
var_dump($cinco);
$cinco = (boolean) $cinco;
var_dump($cinco);
?>
```

Operador de asignación y operadores de comparación

- El operador '=' se usa para asignar un valor a una variable.
- El operador '==' (igual) se usa para verificar la igualdad de los valores de dos expresiones, sin tener en cuenta los tipos.
- El operador '===' (idéntico) se usa para verificar la igualdad de los valores **y** los tipos de dos expresiones.

http://ar2.php.net/manual/es/language.operators.comparison.php

http://ar2.php.net/manual/es/types.comparisons.php

```php
<?php
$a = TRUE;
$b = FALSE;
$c = FALSE;
if ($c = $b) { // $a = $b -> FALSE
    echo '$c = $b iguales';
    echo '<br>';
}
if ($c = $a) { // $b = $a -> TRUE
    echo '$c = $a iguales';
    echo '<br>';
}
if ($a == $c) {
    echo '$a === $c iguales';
    echo '<br>';
}
?>
```