VARIABLES PHP

Ejemplo : Variables PHP válidas/inválidas

```
<?php
$abc = 'Welcome';  //valid
$Abc = 'W3resource.com'; //valid
$9xyz = 'Hello world';  //invalid; empieza con un nº
$_xyz = 'Hello world';  //valid; empieza con un guión bajo
$_9xyz = 'Hello world';  //valid
?>
```

Variable PHP distingue entre mayúsculas y minúsculas

Considera el siguiente ejemplo :

```
<?php
$abc = 'Welcome';
echo "Value of abc : $abc";
echo "Value of ABC : $ABC";
?>
```

Salida del ejemplo

```
Value of abc : Welcome
Value of ABC :
```

PHP variables: Asignación por referencia

PHP (desde PHP4) ofrece otras formas de asignar valores a las variables: asignación por **referencia**. Esto significa que la nueva variable simplemente apunta a la variable original. Cambios en la nueva variable afectan a la original y viceversa.

Considera el siguiente ejemplo:

```
<?php
$foo='bob';
$bar=&$foo;
$bar="my $bar";
echo $bar;
echo '<br />';
echo $foo;
?>
```

Salida:

```
my bob
  my bob
```

PHP variable variables

Sabemos como declara variables en PHP. Pero ¿qué ocurre si el nombre de la variable es la variable en sí misma? En PHP existe el concepto de *Variable Variables*, se puede asignar una variable a otra variable.

Ejemplo:

```
<?php
$v='var1';
echo $v; // prints var1
$$v = 'var2';
echo $$v; // prints var2
echo $var1; // prints var2
?>
```

PHP Ámbito de las variables

En PHP las variabes se pueden declarar en cualquier parte del script. Declaramos las variables para un ámbito en particular. En PHP hay dos tipos de alcance, el alcance **local** donde se crean y se accede a las variables dentro de una función y el alcance **global** donde se crean y se accede a las variables fuera de una función.

Example:

```
<?php
//global scope
$x = 10;
function var_scope()
{
//local scope
$y=20;
echo "The value of x is :  $x "."<br />";
```

```
echo "The value of y is :  $y"."<br />";
}
var_scope();
echo "The value of x is :  $x"."<br />";
echo "The value of y is :  $y ";
?>
```

En el script anterior hay dos variables **x y** y una función **var_scope ()**. **x** es una variable global, ya que se declara fuera de la función y la variable **y** es una variable local ya que se crea dentro de la función **var_scope ()**. Al final de la secuencia de comandos, se llama a la función var_scope (), seguida de dos declaraciones de echo.

```
The value of x is :
The value of y is :  20
The value of x is :  10
The value of y is :
```

Hay dos declaraciones echo dentro de la función var_scope(). Imprime el valor de la variable **y** puesto que es decalrada localmente pero no puede imprimir el valor de **x** porque se crea fuera de la función.

El siguiente echo imprime el valor de la variable **x** puesto que es una variable global, es decir, que no se crea dentro de ninguna función.

El último echo no puede imprimir el valor de la variable **y** ya que es una variable local y se crea dentro de la función var_scope ().

La palabra clave global

Ya hemos aprendido que las variables declaradas fuera de una función son globales. Se puede acceder a ellas en cualquier parte del programa, excepto dentro de una función.

Para utilizar estas variables dentro de una función, las variables deben declararse globales en esa función. Para ello utilizamos la palabra **clave global antes de las variables**. Considera lo siguiente el ejemplo:

Example:

```php
<?php
$x=2;
$y=4;
$z=5;
$xyz=0;
function multiple()
{
    global $x, $y, $z, $xyz;
    $xyz=$x*$y*$z;
}
multiple();
echo $xyz;
?>
```

En el ejemplo anteriror x,**y,** z,**xyz** se han inicializado con los valores 2, 4, 5, 0. Dentro de la función **multiple()** declaramos x,**y,** z,**xyz** como global. Por lo tanto, todas las referencias de cada variable se referirán a la versión global. Ahora cuando se llame a la función **multiple ()** en cualquier parte del script la variable **$ xyz imprimirá 40,** ya que ya se conoce como global.

```
```