Valores de las variables

Cuando hablábamos de las variables, nos referíamos a su **ámbito** y comentábamos que las variables definidas dentro de una función pierden sus valores en el momento en el que abandonemos el ámbito de esa función, es decir, cuando finaliza su ejecución.

Decíamos también que si el **ámbito** en el que hubiera sido definida fuera *externo a una función* los valores sólo *se perderían* –**temporalmente**– mientras durara la eventual *ejecución* de las instrucciones de aquella y que, una vez acabado ese proceso, volvían a recuperar sus valores.

Bajo estas condiciones, si invocáramos repetidamente la misma función obtendríamos cada vez el mismo resultado.

Las posibles modificaciones que pudieran haberse efectuado (a través de las instrucciones contenidas en la función) en el valor inicial de las variables, se perderían cada vez que abandonáramos la función con lo cual, si hiciéramos *llamadas* sucesivas, se repetirían tanto el valor inicial como el resultado.

Variables estáticas

Para poder conservar el último valor de una variable definida *dentro de una función* basta con definirla como **estática**.

La instrucción que permite establecer una variable como **estática** es la siguiente:

static *nombre = valor;*

P. ej: si la variable fuera **$a** y el **valor inicial** asignado fuera **3** escribiríamos:

static $a=3;

La variable conservará el último de los valores que pudo habérsele asignado durante la ejecución de la **función** que la contiene. No retomará el **valor inicial** hasta que **se actualice** la página.

Variables *de variables*

Además del método habitual de asignación de nombres a las variables -poner el signo **$** delante de una palabra- existe la posibilidad de

Variables estáticas

```
<?
# Observa que hemos prescindido de los encabezados HTML.
# No son imprescindibles para la ejecución de los scripts
/* Escribamos una función y llamémosla sinEstaticas
    Definamos en ella dos variables sin ninguna otra especificación
        e insertemos las instrucciones para que al ejecutarse
        se escriban los valores de esas variables */

function sinEstaticas(){

# Pongamos aquí sus valores iniciales
        $a=0;
         $b=0;

# Imprimamos estos valores iniciales

        echo "Valor inicial de $a: ",$a,"<br>";
        echo "Valor inicial de $b: ",$b,"<br>";

/* Modifiquemos esos valores sumando 5 al valor de $a
   y restando 7 al valor de $b.
   $a +=5 y $b -=7 serán quienes haga esas
   nuevas asignaciones de valor
   ya lo iremos viendo, no te preocupes   */

        $a +=5;
        $b -=7;

# Visualicemos los nuevos valores de las variables
        echo "Nuevo valor de $a: ",$a,"<br>";
        echo "Nuevo valor de $b: ",$b,"<br>";

}
# Escribamos ahora la misma función con una modificación que será
# asignar la condición de estática a la variable $b
# Llamemos a esa función: conEstaticas

function conEstaticas(){

# Definimos $b como estática
        $a=0;
          static $b=0;

        echo "Valor inicial de $a: ",$a,"<br>";
        echo "Valor inicial de $b: ",$b,"<br>";

        $a +=5;
        $b -=7;

        echo "Nuevo valor de $a: ",$a,"<br>";
        echo "Nuevo valor de $b: ",$b,"<br>";

}

# Insertemos un texto que nos ayude en el momento de la ejecución

print ("Esta es la primera llamada  a sinEstaticas()<br>");

# Invoquemos la función sinEstaticas;

sinEstaticas();
 # Añadamos un nuevo comentario a la salida
print ("Esta es la segunda llamada  sinEstaticas()<br>");
print ("Debe dar el mismo resultado que la llamada anterior<br>");
# Invoquemos por segunda vez sinEstaticas;
sinEstaticas();
```

que *tomen como nombre* el *valor* de *otra variable* previamente definida.

La forma de hacerlo sería esta:

$$*nombre_variable_previa;*

Veamos un ejemplo.

Supongamos que tenemos una variable como esta:

$color=*"verde";*

Si ahora queremos definir una nueva variable que utilice como nombre el valor (*verde*) que está contenido en la variable previa (*$color*), habríamos de poner algo como esto:

$$color=*"es horrible";*

¿Cómo podríamos *visualizar* el valor de esta *nueva variable*?

Habría tres formas de escribir la instrucción:

print $$color;
o
print ${$color};
o también
print $verde;

Cualquiera de las instrucciones anteriores nos produciría la misma salida: *es horrible*.

Podemos preguntarnos ¿cómo se justifica que existan dos sintaxis tan similares como *$$color* y *${$color}*? ¿Qué *pintan* las llaves?.

La utilización de las llaves es una forma de evitar situaciones de interpretación confusa.

Supongamos que las variables tienen un nombre un poco *más raro*.

Por ejemplo que *$color* no se llama así sino *$color[3]* (podría ser que $color fuera un *array* –una lista de colores– y que esta variable contuviera el tercero de ellos).

En este supuesto, al escribir: *print $$color[3]* cabría la duda de si el número 3 pertenece (es un índice) a la variable *$color* o si ese número corresponde a *$$color*.

Con *print ${$color[3]}* no habría lugar para esas dudas. Estaríamos aludiendo de forma inequívoca a **3** como índice de la variable *$color*.

¿Qué ocurre cuando la variable previa cambia de valor?

Cuando la **variable** utilizada para definir una *variable de variable* *cambia de valor* **no se modifica** *ni el nombre de esta última ni tampoco su valor*.

Puedes ver este concepto, con un

```php
# Hagamos ahora lo mismo con la función conEstaticas

print ("Esta es la primera llamada a conEstaticas()<br>");

conEstaticas();

print ("Esta es la segunda llamada a conEstaticas()<br>");
print ("El resultado es distinto a la llamada anterior<br>");

conEstaticas();

?>
```

ejemplo11.php

Variables de *variables*

```php
<?
# Definamos una variable y asignémosle un valor
          $color="rojo";
# Definamos ahora una nueva variable de nombre variable
# usando para ello la variable anterior
          $$color=" es mi color preferido";

# Veamos impresos los contenidos de esas variables
print ( "El color ".$color. $$color ."<br>");
#o también
print ( "El color ".$color. ${$color}."<br>");
# o también
print ( "El color ".$color. $rojo."<br>");

# advirtamos lo que va a ocurrir al visualizar la página

print ("Las tres líneas anteriores deben decir lo mismo<br>");
print ("Hemos invocado la misma variable de tres formas diferentes<BR>");


# cambiemos ahora el nombre del color
$color="magenta";

/* La variable $rojo seguirá existiendo.
   El hecho de cambiar el valor a $color
   no significa que vayan a modificarse
   las variables creadas con su color anterior
   ni que se creen automáticamente variables
   que tengan por nombre el nuevo valor de $color   */

# Pongamos un mensaje de advertencia para que sea visualizado en la salida

print ("Ahora la variable $color ha cambiado a magenta<br>");
print ("pero como no hemos creado ninguna variable con ese color<br>");
print ("en las lineas siguientes no aparecerá nada <br>");
print ("detrás de la palabra magenta <br>");

# Escribimos los print advertidos
print (" El color ".$color.$$color."<br>");
print (" El color ".$color.${$color}."<br>");

# Comprobemos que la variable $rojo creada como variable de variable
# cuando  $color="rojo" aún existe y mantiene aquel valor

print ("Pese a que $color vale ahora ".$color."<br>");
print ("la vieja variable $rojo sigue existiendo <br>");
print ("y conserva su valor. Es este: ".$rojo);

?>
```

ejemplo12.php

Ejercicio nº 7

poco más de detalle, en el *código fuente* del ejemplo.

La solución del ejercicio

Las diferencias que habrás podido observar al realizar el *ejercicio nº7* están originadas por la interpretación que hacen PHP y el navegador de algunos caracteres especiales, tales como **$**, **"**, o **<**, que al ser interpretados como *símbolos del lenguaje* no se imprimen en pantalla.

Cuando pretendamos que aparezcan *escritos* tendremos que indicarlo de una forma especial.

En este enlace podrás ver la forma de hacerlo.

Abre tu editor creando un documento nuevo. Copia el código fuente del ejemplo anterior y pégalo en ese documento. Ahora guárdalo como **ejercicio7.php**. Abre en tu navegador *ejercicio7.php* y compáralo con el que visualizas al pulsar sobre **ejemplo12.php**. Comprueba que existen algunas diferencias entre ambos.

¡Cuidado!

Aunque en los enunciados no se advierta, como norma general, los ejercicios deberás guardarlos siempre en el directorio **practicas** que has creado al realizar el ejercicio nº 2

Anterior Índice Siguiente