¿Qué es una variable?

Podríamos decir que es un espacio de la *memoria RAM* del ordenador que se *reserva* –a lo largo del tiempo de ejecución de un script– para almacenar un determinado tipo de datos cuyos valores son susceptibles de ser modificados por medio de las instrucciones contenidas en el propio programa.

Nombres de variables

En PHP todos los nombres de variable tienen que empezar por el símbolo **$**.

Las nombres de las variables han de **llevar una letra** inme- diatamente después del sím- bolo **$** –**$pepe1** es un nombre válido, pero **$1pepe** *no es un nombre válido*–.

Para PHP las letras *mayúsculas* y las *minúsculas* son **distintas**. La variable **$pepe** es **distinta** de **$Pepe**.

Tipos de variables

En PHP no es necesario definir el **tipo de variable**, por lo tanto, **una misma variable** puede contener una *cadena de caracteres* en un momento del proceso y, posteriormente, un *valor numérico*, susceptible de ser operado matemáticamente.

Definición de variables

PHP no requiere una definición previa de las variables. Se definen en el momento en que son necesarias y para ello basta que se les asigne un valor.

La sintaxis es esta:

$variable=*valor*;

El *valor* puede ser una *cadena* (texto o texto y números que no requieren ser operados matemáticamente) o sólo un *número*. En el primero de los casos habría que escribir el valor **entre comillas**.

Ámbito de las variables

Los valores de una variable definida en cualquier parte de un script *–siempre que no sea dentro de una función–* pueden ser utilizados desde cualquier otra parte de ese script, **excepto desde dentro de las funciones** que contuviera el propio

Practicando con variables y sus *ámbitos*

Podemos comparar la **memoria** de un ordenador con el *salón* de un restaurante y la **ejecución de un programa** con los servicios que van a darse en la *celebración del final de año*. La forma habitual de hacer una **reserva** de mesa –**espacio de memoria**– para ese evento sería facilitar un nombre –**nombre de la variable**– y especificar además cuantos comensales –**tipo de variable**– prevemos que van a asistir.

Cuando acudamos a la cena de *San Silvestre* podremos sentarnos en esa mesa un número determinado de comensales –**daremos un valor a la variable**– y a lo largo de ella podremos levantarnos o incorporar *un nuevo invitado* –**modificación del valor de la variable**– siempre que sea *alguien de nuestro* **ámbito** quien realice la invitación.

Probablemente no permitiríamos que el *cocinero* decidiera quien debe sentarse o levantarse, pero si lo permitiríamos a cualquiera de nuestros invitados. La diferencia estaría –**ámbito de la variable**– en que el *cocinero* no pertenece a nuestro ámbito mientras que los invitados a nuestra mesa sí.

Quizá si celebráramos el evento otro día cualquiera no necesitaríamos hacer *una reserva previa* y bastaría con acudir a la hora deseada y *hacer la reserva* justo en el momento de sentarse.

El *restaurante* de **PHP** no necesita que hagamos ninguna reserva previa. Otros muchos lenguajes de programación, por el contrario, si la necesitan.

Siguiendo con lo que nos ocupa, aquí tienes un ejemplo del uso de las variables y la forma de utilizarlas en los diferentes ámbitos.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<?
# Definimos la variable $pepe como vacía
$pepe="";

# Definimos las variables $Pepe y $Pepa (ojo con mayúsculas y minúsculas)
$Pepe="Me llamo Pepe y soy serio y formal";
$Pepa="Me llamo Pepa y también soy seria y formal";

?>

<!-- esto es HTML, hemos cerrado el script -->

<center><b>Vamos a ver el contenido de las variables</b></center>

<!--   un nuevo script PHP -->

<?
echo "<br> El valor de la variable pepe es: ",$pepe;
echo "<br> No ha puesto nada porque $pepe esta vacía";
echo "<br> El valor de la variable Pepe es: ",$Pepe;
?>

<center><b><br>Invocando la variable desde una función</b></center>

<?
/* Escribiremos una function llamada vervariable
   Observa la sintaxis. La palabra function delante
   y el () al final seguidos de la llave.
   Hasta que no cerremos la llave todas las líneas
   serán consideradas parte de la función */

function vervariable(){
echo "<br> Si invoco la variable Pepe desde una función";
echo "<br>me aparecerá en blanco";
```

script o desde las que pudieran estar contenidas en un *fichero externo*.

Si una variable es definida **dentro de una función** sólo podrá ser utilizada dentro esa función.

Si en una *función* aludimos a una **variable externa** a ella PHP considerará esa llamada como si la variable tuviera valor **cero** (en caso de ser tratada como número) o una **cadena vacía** (**""** es una cadena vacía).

Igual ocurriría si **desde fuera** de una función hiciéramos alusión 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 globales

Lo comentado anteriormente, admite algunas **excepciones**.

Las funciones pueden utilizar valores de *variables externas a ellas* pero ello requiere **incluir** *dentro de la propia función* la siguiente instrucción:

global *nombre de la variable*;

Por ejemplo: **global $a1;**

En una instrucción –**global**– pueden definirse como tales, de forma simultánea, varias variables. Basta con escribir los nombres de cada una de ellas separados por comas.

P. ej.: **global $a1, $a2, $a3;**

Variables superglobales

A partir de la versión **4.1.0** de **PHP** se ha creado un nuevo tipo de variables capaces de *comportarse como globales* sin necesidad de que se definan como tales.

Estas variables que *no pueden ser creadas por usuario*, recogen de forma automática *información muy específica* y tienen nombres preasignados que no pueden modificarse.

Las estudiaremos un poco más adelante. Por ahora, sólo citar los nombres de algunas de ellas:

$_SERVER, **$_POST**, **$_GET** o **$_ENV** son los de las más importantes.

```
}
/* esta llave de arriba señala el final de la función.
   Los contenidos que hay en adelante ya no pertenecen a ella */

/* Haremos una llamada a la funcion vervariable.
   Las funciones no se ejecutan hasta que no se les ordena
   y se hace de esta forma que ves aquí debajo:
   nombre de la funcion seguido de los famosos paréntesis */

vervariable();
?>

<!-- mas HTML puro  -->
<center><b><br>Ver la variable desde la función
                        poniendo <i>global</i></b></center>

<?
# una nueva funcion

function ahorasi(){
        # aqui definiremos a $Pepe como global
        # la función leerá su valor externo
                        global $Pepe;

   echo "<br><br> Hemos asignado ámbito global a la variable";
   echo "<br>ahora Pepe aparecerá";
   echo "<br>El valor de la variable  Pepe es: ", $Pepe;

}
# hemos cerrado ya la funcion con la llave.
# Tendremos que invocarla para que se ejecute ahora
ahorasi();
?>

<center><b><br>Un solo nombre y dos <i>variables distintas</i></b><br>
Dentro de la función el valor de la variable es <br></center>

<?
function cambiaPepa(){

        $Pepa="Ahora voy a llamarme Luisa por un ratito";

        echo "<br>",$Pepa;
}

cambiaPepa();
?>
<center>... pero después de salir de la función
                    vuelvo al valor original...</center>
<?
echo "<br>",$Pepa;
?>


</BODY>
</HTML>
```

Ver ejemplo9.php

Ejercicio nº 5

Escribe un script (guárdalo como **ejercicio5.php**) en el que una misma variable tome dos valores distintos sin utilizar ninguna función. Luego añade al script una función que presente ese mismo nombre de variable con un valor distinto de los anteriores, comprobando que esta última opción no modificó el último valor de aquellos.