¿Qué son las funciones de usuario?

De igual forma que ocurre con el navegador en el caso del HTML, **PHP** lee e interpreta las instrucciones contenidas en los scripts de forma *secuencial*.

Es decir, las instrucciones se van ejecutando en el mismo orden en el que aparecen en el documento original, con la **excepción** de las *funciones*.

En este caso, los bloques de instrucciones son *puestos a disposición* de PHP, pero **no se ejecutarán** hasta el momento en que sean requeridas de forma expresa.

¿Dónde deben insertarse?

Aunque en *versiones antiguas* de PHP era necesario **definir** la función antes de invocarla, a partir de la versión 4 **no es necesaria** esa organización secuencial.

La función pueden estar escrita dentro de cualquier script y en cualquier parte del documento, sin que tenga importancia alguna el lugar en el que se incluya la llamada a la misma.

También es posible –y bastante habitual– incluir funciones de uso frecuente en *documentos externos* de modo que pueden ser *compartidas*.

En este caso, además de *invocarla* es necesario indicar a PHP el lugar donde debe buscarla. Hablaremos de ello cuando estudiemos lo relativo a **include**.

Definición de la función

Las funciones de usuario requieren la siguiente sintaxis:

function *nombre*(){
.....
... instrucciones ...
.....
}

Es imprescindible respetar estrictamente la sintaxis que requiere de forma obligatoria los siguientes elementos:

– La palabra **function** debe estar

Funciones de usuario

Imaginémonos, allá por el mes de junio, sentados ante una mesa tratando de hacer la *declaración de la renta*. Para ese menester seguramente nos pertrecharíamos –además del impreso oficial de la declaración– con: calculadora, lápiz, goma de borrar e incluso con las disposiciones legales relativas al impuesto.

Seguramente iríamos leyendo y cumplimentando el *impreso oficial* –**script PHP**– partiendo de la primera página –**orden secuencial**– y continuando de forma ordenada hasta el final.

La *calculadora*, la *goma*, etcétera –**funciones**– estarían *disponibles* para ser utilizadas –**invocadas**– tantas veces como fuera preciso y, además, no serían *elementos exclusivos* de ese documento, sino que seguirían *disponibles* para otros usos –**scripts**– distintos, tales como *la planificación financiera de nuestras vacaciones*, por citar un ejemplo.

Las **funciones de usuario** son, como *la calculadora* o *la goma*, **herramientas** diseñadas para *facilitar las tareas* y susceptibles de ser usadas en una o varias situaciones –**scripts**– diferentes.

Ejemplos de funciones de usuario

Con este primer ejemplo obtendremos *una página en blanco*. El script contiene una función pero no hay ninguna instrucción que la invoque y por lo tanto no se ejecutaría.

```
<?
function a1(){

   for($i=1;$i<=10;$i++){
             echo $i,"<br>";
             }
}
?>
```

ejemplo64.php

```
<?
a1();
?>
<!-- Hemos escrito un script con una llamada
     a la función a1 que aún no está definida.
         Tendremos que hacerlo, pero no importa
         la parte del documento en la que lo hagamos
         La pondremos en este nuevo script PHP   //-->

<?
function a1(){
   for($i=1;$i<=10;$i++){
             echo $i,"<br>";
             }
}
?>
```

ejemplo65.php

En este otro ejemplo veremos las diversas situaciones que pueden plantearse respecto al **ámbito** de las variables.

```
<?
# definamos dos variables y asignémosles un valor
$a=5; $b=47;
```

escrita en minúsculas.

– El *nombre de la función*, que debe seguir criterios similares a los de los nombres de variables, aunque en este caso no se antepone el símbolo **$** ni ningún otro.

– Los paréntesis **()**, incluso cuando no contengan nada.

– Las *llaves* de *apertura* (**{**) y *cierre* (**}**) dentro de las cuales se escribirán las instrucciones correspondientes a ella.

Ejecución de la función

Las funciones PHP **no se ejecutan** en tanto no sean **invocadas**.

Para *invocar una función* la sintaxis es la siguiente:

*nombre***()**

Al ser *llamada* con esta sintaxis –desde cualquier script– se *ejecutarán* las instrucciones contenidas en ella.

Ámbito de las variables

Resumamos lo ya comentado cuando tratamos el tema de las variables.

– Las funciones no leen valores de variables definidas fuera de su ámbito salvo que dentro de la propia función se definan de forma expresa como **globales**.

– Si una función modifica el valor de una variable **global**, el nuevo valor persiste después de abandonar la función.

– Si dentro de una función se utiliza un nombre de variable idéntico al de otra externa a ella (sin definirla global) la nueva variable se inicia con valor nulo y los eventuales valores que pudiera ir conteniendo se pierden en el momento en que se acaba su ejecución.

Asignación de valores a variables

A las variables no globales se les pueden asignar sus valores iniciales de dos formas:

• Incluyéndolas en una línea de instrucciones *contenida* en la propia función.

• Insertando los nombres de variable y sus valores dentro del paréntesis que –de forma obligatoria– debe seguir al nombre de la función. En este caso la sintaxis sería:

function *nom* (*$a=v1,$b=v2*)

donde **$a** y **$b** son nombres de variables a utilizar en el ámbito de la

```php
# escribamos una funcion a1 y pidámosle que imprima sus valores

function a1(){
echo "Este es el valor de $a en la función a1: ",$a,"<br>";
echo "Este es el valor de $b en la función a1: ",$b,"<br>";
}
# hagamos una llamada a la funcion anterior
# no nos escribirá ningún valor porque esas variables no pertenecen
# al ámbito de la función y serán consideradas como vacias
# en el ambito de la funcion
a1();
# escribamos una nueva función, definamos como global $a
# y comprobemos que ahora si la hemos incluido en el ambito
# de la funcion
function a2(){
global $a;
echo "Este es el valor de $a en la función a2: ",$a,"<br>";
echo "Este es el valor de $b en la función a2: ",$b,"<br>";
}
# invoquemos esta nueva funcion y veamos que ahora
# si se visualiza el valor de $a pero no el de $b
a2();
# creemos una nueva funcion y ahora modifiquemos dentro de ella
# ambas variables
function a3(){
global $a;
$a +=45;
$b -=348;
echo "Este es nuevo valor de $a en la función a3: ",$a,"<br>";
echo "Este es el valor de $b en la función a3: ",$b,"<br>";
}
# invoquemos la funcion a3
a3();
# comprobemos -desde fuera del ámbito de la función
# que ocurrió con los valores de las variables
echo "El valor de $a HA CAMBIADO despues de ejecutar a3 es: ",$a,"<br>";
echo "El valor de $b NO HA CAMBIADO despues de ejecutar a3 es: ",$b,"<br>"
# probemos que ocurre con una variable superglobal
# veremos que sin ser definida expresamente en a4
# si pertenece a su ambito y por lo tanto visualizamos su contenido
function a4(){
print "La superglobales si están: ".$_SERVER['SERVER_NAME']."<br>";
}
# invoquemos esta nueva funcion
a4();
?>
```

ejemplo66.php

```php
<?
$a=-13; $b=7482; $c="Ambrosio";
# esta es una forma alternativa de asignar valores a una variable
# del ambito de la función
function a1($a=56, $b=25){
echo "El valor  de $$a en la función a1: ", $a,"<br>";
echo "El valor  de $$b en la función a1: ", $b,"<br>";
}
a1();
echo "El valor de $a despues de ejecutar la función es: ",$a,"<br><br>";



# Pasando valores desde la llamada a la función #
/* Definamos una función fun1 e incluyamos dentro de su paréntesis
nombres de variables, separados por comas pero ahora sin asignarles
ningún valor  */
function fun1($x,$y,$z){
        print "Valor de la variable x: ".$x."<br>";
        print "Valor de la variable y: ".$y."<br>";
        print "Valor de la variable z: ".$z."<br>";
}



# debemos hacer la llamada a la función pero ahora
# lo haremos de forma distinta.
```

función y **v1** y **v2** los valores asignados a cada una de ellas.

En este paréntesis pueden incluirse –separándolas con **comas**– cuantas parejas *var = val* sean necesarias.

• Una forma alternativa a la anterior sería la siguiente:

function *nom* (*$a,$b*)

donde habría que asignar los valores de cada una de la variables *desde la llamada a la función*, que ahora tendría esta sintaxis:

nombre (*valor1, valor2,...*);

en la que se escriben los *valores* separados por comas, y *encerrados entre comillas* cuando se trata de variables alfanuméricas.

Si el número de valores contenidos en la llamada fuera mayor que el número de variables definidas en la función, los excedentes serían ignorados y, si fuera inferior, se asignaría valor nulo a las variables a las que no se transfiriera ningún valor.

• También es posible incluir en la llamada a la función los *nombres de algunas variables* definidas en el ámbito externo a la función. Se haría de la siguiente forma:

nombre (*$var1, var2,...*);

```php
# Vamos a incluir en la llamada
# los valores que queremos asignar a las variables de la función
# Escribiremos dentro del paréntesis de la llamada
# los valores de cada una de las tres variables
# separados por comas
# (si se trata de una cadena, pongámosla entre comillas)
# y veremos con la función recoge esos valores asignados
#en la llamada

fun1(14,"Robustiano",23.4);
/* si esta llamada contuviera más de tres valores
   los ultimos serian ignorados */
fun1(49.3,"Eustaquio",78,"Lupicio",456);
# si  contuviera menos de tres valores
# PHP nos daría un mensaje de error
# advirtiendo que falta un valor
# pero nos devolvería los valores
fun1("Desiderio","Bailador");

# esos mensajes de error podríamos evitarlos
# poniendo una arroba delante de la llamada a la función
@fun1("Nuevo Desiderio","Nuevo Bailador");


# tambien podría utilizarse una sintaxis como esta
#  en la que dejamos en blanco (entre comillas)
#  el espacio correspondiente al segundo valor
#  aunque si incluimos las comas.
#  La variable que ocupa esa posición
#  sería considerada como nula
fun1("La luna",'',"verde");

# también podríamos incluir en la llamada nombres de variables
# definidas en el ámbito general del sript
# un este caso la función usaria esos valores

fun1($a,$b,$c);

?>
```

ejemplo67.php

Pasar por *referencia*

Tal como hemos visto, las funciones PHP pueden recibir **valores** de variables externas y utilizar esos **valores** sin que el valor original de las mismas –salvo que se les asigne la condición de globales *dentro* de la función– sufra modificación.

Una manera de *lograr* que los *valores una variable externa* puedan ser modificados por una función, es lo que se llama en *argot informático «pasar variables por referencia»*.

La forma de hacerlo es esta:

Hay que **anteponer** al nombre de la variable el símbolo **&** y PHP interpretará que la estamos *pasando por referencia*.

El **&** puede anteponerse tanto en *la definición de la función* como en *la llamada a la función*, tal como puedes ver en el ejemplo.

La segunda de las opciones *nos concede mayor libertad* dado que permite usar una sola función y *decidir en cada llamada* la forma de pasar los parámetros.

¡Cuidado!

```php
<? $a=3; $b=2;
function a1(&$a,$b){
     $a=pow($a,2);
     $b=pow($b,3);
echo "El cuadrado de a dentro de la función es: ",$a, "<br>";
echo "El cubo de b dentro de la función es: ",$b, "<br><br>";
}

a1($a,$b);

echo "Al salir de la función a conserva la modificación: ",$a, "<br>";
echo "Por el contrario, b no la conserva: ",$b, "<br><br>";

$c=8; $d=12;
function b1($a,$b){
     $a=pow($a,2);
     $b=pow($b,3);
echo "El cuadrado de a dentro de la función es: ",$a, "<br>";
echo "El cubo de b dentro de la función es: ",$b, "<br><br>";
}

b1(&$c,$d);

echo "Al salir de la función c conserva la modificación: ",$c, "<br>";
echo "Por el contrario, d no la conserva: ",$d, "<br><br>";
?>
```

ejemplo68.php

Si tratas de ejecutar **una función** en la que colocas el **&** en la *llamada* a la función y te aparece un mensaje como este:

«*Warning: Call-time pass-by-reference has been deprecated -argument passed by value; If you would like to pass it by reference, modify the declaration of function(). If you would like to enable call-time pass-by-reference, you can set allow_call_time_pass_reference to true in your INI file*».

lo que estará ocurriendo es que el **php.ini** del servidor tiene configurada en **Off** la directiva:

allow_call_time_pass_reference

y eso suele ocurrir con algunos *hostings* y también con la configuración por defecto de algunas versiones de PHP anteriores a la que estamos utilizando.

Otra forma de definir funciones de usuario

Existe otra opción de definición de funciones de usuario que puede resultar de mucho interés. En este caso la *función* se define en tres bloques:

– **Definición** de la función, *llave de apertura* y **cierre** del script PHP.

– **Contenido** de la función formado exclusivamente por código HTML, que se escribiría cuando fuera invocada la función que lo contiene.

– **Cierre** de la función (*llave de cierre*) contenido en un script PHP, es decir, entre las etiquetas de apertura **<?** y cierre **?>** de PHP.

Cuando es invocada una función definida de esta forma –puedes verlo en el ejemplo– PHP se limita a *escribir* en el documento final los textos contenidos entre la etiqueta de apertura y cierre de la función.

Las funciones de esta forma son particularmente útiles para la construcción de espacios web que contienen una serie de páginas en las que se repiten las mismas estructuras.

Ejercicio nº 26

En este ejercicio –**ejercicio26.php**– utilizaremos **una función** para construir tablas similares a las que hemos construido en el ejercicio nº 23. Pero incorporaremos una innovación respecto a aquel. Ahora la función debe permitir construir tablas de cualquier dimensión –nº de filas y/o columnas– y el número de estas habremos de incluirlo en la llamada a esa función.

Otras funciones de usuario

```
<? function Encabezado() { ?>
<!-- Hemos abierto la función y cerrado la etiqueta PHP
     todo esto es código HTML //-->
          <HTML>
          <HEAD>
          <TITLE>Titulo de mi página</TITLE></HEAD>
          <BODY BGCOLOR="#FF0000">
<!-- Esta nueva llamada a PHP
     insertando la llave de cierre de la función
         indicará a PHP que debe escribir todo lo
         contenido entre la { y esta } //-->
<? } ?>

<? function Pie() { ?>
          <HR>
          </BODY>
          </HTML>
<? } ?>
<!-- Utilizaremos esas dos funciones para
        crear una página web. Llamamos a la función Encabezado
        luego escribimos un texto y por ultimo insertamos
        el Pie de página con la función Pie //-->
<? Encabezado(); ?>
        Este es texto que aparecerá en el cuerpo de la página.
         Está fuera de los scripts de php y será considerado
         como un texto HTML. Debajo aparecerá la línea horizontal
        que insertaremos mediante una nueva llamada a la función Pie

<? Pie(); ?>
```

ejemplo68a.php

Anterior Índice Siguiente