



U.T. 3: Fundamentos de la inserción de código en páginas web (3) : Funciones

Funciones en PHP

- ❑ Permiten encapsular código por funcionalidad y reutilización
- ❑ Las variables dentro de las funciones tienen ámbito local, para utilizar variables globales es necesario usar la palabra reservada *global*
- ❑ Pueden
 - ❑ Recibir valores (parámetros o argumentos)
 - ❑ Devolver valores (return)

Funciones en PHP : Declaración

❑ Sintaxis:

```
function nombreFunción (param1,param2){  
instrucción1;  
instrucción2;  
return valor_de_retorno;  
}
```

Ejemplo:

```
function suma ($x, $y){  
$s = $x + $y;  
return $s;  
}  
$a=1;  
$b=2;  
$c=suma ($a, $b);  
print $c;
```

Definición

Invocación

❑ La primera línea es la cabecera de la función y consta de:

- ❑ La palabra clave *function* .
- ❑ El nombre de la función, que no debe llevar acentos, espacios en blanco, ni caracteres especiales.
- ❑ La lista de parámetros encerrada entre paréntesis, separados por comas.

Funciones en PHP : Declaración

```
function elmayor ($a,$b){  
    if $a > $b}  
        return $a;  
    return $b;  
}
```

Paso de parámetros

Invocación de la función

Retorno del resultado

```
<html><body>  
$x;  
$y;  
$c=elmayor($x,$y);  
echo $c;  
</body></html>
```


Funciones en PHP: Paso de parámetros

- ❑ Los parámetros se pasan por valor:
 - ❑ Cambiar el valor dentro de la función no lo cambia fuera.
- ❑ Los parámetros se pasan por referencia:
 - ❑ Cambiar el valor dentro de la función si lo cambia fuera
 - ❑ Usar **&**

```
function incrementa (&$a){  
    $a = $a + 1;  
}  
$a=1;  
incrementa ($a);  
Echo $a; // Muestra un 2
```

- ❑ En la llamada a la función no se usa & aunque los parámetros se pasen por referencia (Obsoleto desde 5.4.0)
-

Funciones en PHP: Paso de parámetros

❑ Parámetros por defecto

- ❑ Valores que toman los parámetros cuando se omiten en la llamada:

```
function muestranombre ($titulo = "Sr."){  
    echo "Estimado $titulo:\n";  
}  
muestranombre ();  
muestranombre ("Prof.");
```

Estimado Sr.:
Estimado Prof.:

- ❑ Los argumentos con valores por defecto deben ser siempre los últimos en la lista:

```
function muestranombre ($apel, $titulo= "Sr."){  
    print "Estimado $titulo $apel:\n";  
}  
muestranombre ("Fernández");  
muestranombre ("Fernández", "Prof.");
```

Estimado Sr. Fernández:
Estimado Prof. Fernández:

Funciones en PHP: Ámbito de las variables

❑ Los tres posibles ámbitos de una variable son:

❑ Variables locales

- ❑ Definidas dentro de una función y sólo pueden ser accedidas desde dentro de esta función.
- ❑ Si hay una variable local y otra global con el mismo nombre, dentro de la función se usa la local.
- ❑ Pierden su valor al salir de la función

❑ Variables locales estáticas

❑ Variables globales

```
function Test (){  
    $a = 0;  
    echo $a;  
    $a++;  
}
```

Funciones en PHP: Ámbito de las variables

❑ Variables locales estáticas

- ❑ Existen sólo en el ámbito local de la función
- ❑ No pierden su valor cuando el programa abandona la función
- ❑ Anteponer la palabra reservada ***static***.

```
function Test (){  
    static $a = 0;  
    echo $a;  
    $a++;  
}
```



Cada vez que se llame a la función Test(), se visualizará el valor de \$a y se incrementará.
La primera vez que se llame a Test() escribirá un 0, luego 1,2,...

Funciones en PHP: Ámbito de las variables

❑ Variables globales

- ❑ Se definen fuera del cuerpo de una función
- ❑ Accesibles desde cualquier punto del código.
- ❑ Deben ser declaradas globales dentro de la función para usarlas o bien usar el array `$GLOBALS` .

```
$a = 1; /* ámbito global */  
function Prueba(){  
    echo $a; /* variable global */  
}  
Prueba();
```



Nada.
Porque \$a no está
definida dentro de la
función

```
$a = 1;  
function Prueba2(){  
    global $a;  
    echo $a;}  
Prueba2();
```

→ 1

```
$a = 1;  
function Prueba3(){  
    echo $GLOBALS["a"];  
}  
Prueba3();
```

→ 1

Funciones en PHP: Devolución de valores

- ❑ El valor devuelto desde una función
 - ❑ Puede ser asignado a una variable
 - ❑ Puede ser utilizado dentro de una expresión.
 - ❑ Devuelve sólo un único valor
 - ❑ Para devolver múltiples valores usar un *array*
 - ❑ Usar la palabra reservada *return*
 - ❑ Cuando aparece *return* la función deja de ejecutarse
 - ❑ Si después de *return* hay más líneas de código, no se ejecutarán nunca.
-

Funciones en PHP: N° variable de parámetros

- ❑ PHP permite una lista de valores de longitud variable como parámetro
- ❑ Funciones a usar:
 - ❑ **func_num_args()** → número de argumentos pasados a la función.
 - ❑ **func_get_args()** → array con los argumentos pasados a la función
 - ❑ **func_get_arg(num)** → el argumento que está en la posición *num* en la lista de argumentos. La primera posición es la 0

```
function prueba(){  
    $num_args = func_num_args();  
    echo "Numero de argumentos:$num_args<br/>\n";  
    if ($num_args >= 2) {  
        echo "El 2º argumento es:".func_get_arg(1)."<br/>\n";  
        $parametros=func_get_args();  
        echo "Array con todos los argumentos:<br />\n";  
        print_r($parametros);  
    }  
    prueba (1, 2, 3);  
}
```

```
Numero de argumentos: 3  
El 2º argumento es: 2  
Array con todos los argumentos:
```

```
Array  
(  
    [0] => 1  
    [1] => 2  
    [2] => 3  
)
```

Funciones en PHP: Nº variable de parámetros

- ❑ PHP 5.6: las listas de argumentos de las funciones pueden incluir el token ... para indicar que aceptan un número variable de parámetros.
- ❑ Los argumentos serán pasados a la variable dada como un array

```
<?php
function sum(...$números) {
    $acu = 0;
    foreach ($números as $n) {
        $acu += $n;
    }
    return $acc;
}

echo sum(1, 2, 3, 4); // El resultado sería 10
?>
```


Funciones en PHP: Funciones variables

- ❑ Definimos una variable: **\$func=prueba;**
- ❑ Si llamamos a esa variable con paréntesis: `$func()` → PHP buscará una función con el mismo nombre que su contenido y la ejecutará si existe.
- ❑ Las funciones variables no funcionarán con `echo()`, `print()`, `unset()`, `isset()`, `empty()`, `include()`, `require()` y derivados.

```
function prueba($arg = ' '){  
    echo "Estamos en la función prueba(); y el argumento es '$arg'.<br>\n";  
}  
$func = "prueba";  
$func('hola'); // Esto llama prueba('hola')
```

Estamos en la función prueba(); y el argumento es 'hola'.

Funciones en PHP: Funciones recursivas

- ❑ Una función se llama recursiva cuando en algún punto de su cuerpo se llama a sí misma.
- ❑ Cuidado! puede llamarse a sí misma indefinidamente.
- ❑ Es muy importante la condición de salida.

```
<?php
function recursividad($a){
    if ($a < 20) {
        echo "$a\n";
        recursividad($a + 1);
    }
}
?>
```


Funciones en PHP: Funciones anónimas

- ❑ Las funciones anónimas, también se conocen como ***clausuras*** (closures).
- ❑ Permiten la creación de funciones que no tienen un nombre especificado.
- ❑ Las clausuras también se pueden usar como valores de variables.

```
<?php
    $saludo = function($nombre){
        printf("Hola %s\r\n", $nombre);};

    $saludo('Mundo');
    $saludo('PHP');
?>
```

Funciones en PHP

❑ `bool function_exists (string $function_name)`

- ❑ Comprueba la lista de funciones definidas, las incluidas (internas) y las definidas por el usuario, para ***function_name***.
- ❑ Recibe el nombre de la función buscada como cadena.
- ❑ Devuelve ***TRUE*** si ***function_name*** existe y es una función, si no, ***FALSE***.

```
<?php
    if (function_exists('imap_open')) {
        echo "Las funciones de IMAP están disponibles.<br />\n";
    } else {
        echo "Las funciones de IMAP no están disponibles.<br />\n";
    }
?>
```