

Hasta aquí hemos visto como utilizar funciones en PHP

- Como declararlas
- como llamarlas

Hemos tenido que utilizarlas sin apenas tener conocimientos puesto que no se puede hacer absolutamente nada sin funciones.

Vamos ahora a hacer repaso de lo que ya sabemos , y ver cosas nuevas...

- ¿Qué son las funciones? ¿Para qué sirven?
- Funciones predefinidas.
- Funciones propias. Declaración y ejecución (llamada).
- Parámetros de funciones.

¿para qué sirven las funciones?

- Sobre todo para automatizar tareas.
- Ventaja permite eliminar código repetitivo en los programas (se podría comparar con los macros.). Útil para aquellas tareas que realizan de forma repetitiva

Ejemplo: imaginemos que tenemos que hacer a menudo un cálculo concreto . Creamos una función y solo tenemos que llamarla cuando la necesitamos.

Hay dos grandes grupos de funciones

- Las predefinidas: que vienen con el lenguaje PHP; las puedes utilizar cuando quieras (ej. echo, print)
- Funciones propias las crea el programador con el objetivo de utilizarlas en un futuro

¿dónde acudir para ver todas las funciones PHP?

En Google. PHP listado de funciones y métodos . Nos lleva a PHP.net como siempre

Listado de Funciones y Métodos

Lista todas las funciones y métodos del manual

[a](#) [b](#) [c](#) [d](#) [e](#) [f](#) [g](#) [h](#) [i](#) [j](#) [k](#) [l](#) [m](#) [n](#) [o](#) [p](#) [q](#) [r](#) [s](#) [t](#) [u](#) [v](#) [w](#) [x](#) [y](#) [z](#) [_](#)

- [a](#)
 - [abs](#) - Valor absoluto
 - [acos](#) - Arco coseno
 - [acosh](#) - Arco coseno hiperbólico
 - [addslashes](#) - Escapa una cadena al estilo de C
 - [addslashes](#) - Escapa un string con barras invertidas
 - [apache_child_terminate](#) - Finaliza un proceso de Apache después de esta llamada
 - [apache_getenv](#) - Obtiene una variable del entorno subprocess_env de Apache
 - [apache_get_modules](#) - Obtiene una lista de los módulos cargados en el servidor Apache
 - [apache_get_version](#) - Obtiene la versión del servidor Apache
 - [apache_lookup_uri](#) - Realiza una petición parcial por la URI especificada y devuelve toda la información sobre ella
 - [apache_note](#) - Obtiene y establece las notas de petición de apache
 - [apache_request_headers](#) - Obtiene todas las cabeceras HTTP
 - [apache_reset_timeout](#) - Restaura el temporizador de Apache
 - [apache_response_headers](#) - Obtiene todas las cabeceras HTTP de respuesta
 - [apache_setenv](#) - Establece una variable subprocess_env de Apache
 - [APCIterator::current](#) - Obtener el elemento actual

echo|

Todas las funciones aquí vienen con la sintaxis y los argumentos que pueden tener. ¿Qué son los argumento o parámetros también denominaos parámetros de una función?

Es lo que a entre paréntesis en la sintaxis.

Hay funciones predefinidos que no reciben argumento ninguno.

Algunas funciones pueden recibir argumento y otras no

Estos argumentos son como variables que utiliza la función para almacenar valores y utilizarlas en la tarea que tiene que realizar

Volviendo a la pagina de php.net . Además de la sintaxis esta pago nos da una descripción de qué es lo que hace una función

Ojo!! En la pagina dice que echo no es realmente una función aunque vulgarmente decimos que es una función . NO es una función sino un constructor). También proporciona un ejemplo de la misma

EJEMPLO DE FUNCION PREDEFINIDA

-strtolower(cambia una cadena de texto a minúsculas)

-strtoupper (a mayúsculas)

-ucwords (la primera en mayúscula)

1. Primero creamos un documento llamado funciones_predefinidas
2. Buscamos en php.net las funciones strtolower
3. Pulsamos en la que sea y nos sale un ejemplo de sintaxis

strtolower

Change language: Spanish Edit Report a Bug

(PHP 4, PHP 5)
strtolower — Convierte una cadena a minúsculas

Descripción

+

```
string strtolower ( string $string )
```

Devuelve una **string** con todos los caracteres alfabéticos convertidos a minúsculas.

Observe que 'alfabético' se determina por la localización actual. Por ejemplo, los caracteres de localización "C" como umlaut-a (Ä) no serán convertidos.

Parámetros

las funciones siempre van a devolver un dato, por lo que el primer elemento de la sintaxis indica el tipo de dato que va devolver (en este caso string). También nos indica que hay un argumento de tipo string

Primero cambiamos ser texto a minúscula....

```
<?php
```

```
$palabra="JUAN";
```

```
echo(strtolower($palabra));
```

```
?>
```

Después a mayúscula

```
<body>
```

```
<?php
```

```
$palabra="juan";
```

```
echo(strtoupper($palabra));
```

```
?>
```

Acuérdete una función no se ejecuta hasta que no es llamada. Y se llama a una función por su nombre seguida de apertura y cierre de paréntesis.

Hay que mencionar que en este ejemplo nos encontramos con las funciones anidadas : función predefinida detrás de otra (strtoupper dentro de echo)

Vamos ahora a ver ejemplo de función propia (realizada por el programador) que recibe unos argumentos).

```
<?php
```

```
function suma($num1, $num2){  
  
    $resultado=$num1+$num2;  
  
    return $resultado;  
  
}
```

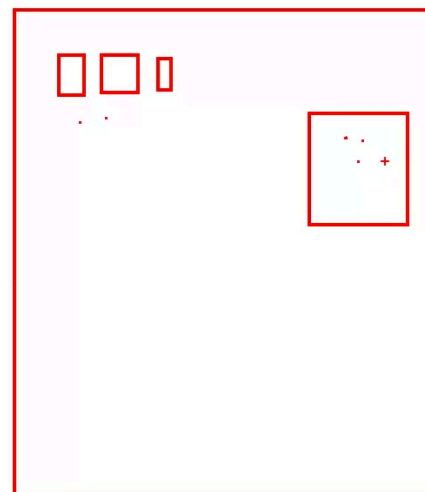
```
?>
```



```
<?php
```

```
function suma($num1, $num2){  
  
    $resultado=$num1+$num2;  
  
    return $resultado;  
  
}
```

```
?>
```



```
<?php

function suma($num1, $num2){

    $resultado=$num1+$num2;

    return $resultado;

}

suma(5,7);
```

num1=5
num2=7|

En el ejemplo, la función está preparad para recibir 2 argumentos, por lo que en la llamada (suma (5,7) le pasamos 2 argumentos.

¿Cuál es el proceso?

Cuando el flujo del programa llega a la línea de la llamad, encuentra que estamos haciendo una llamada a una función, por lo que el flujo de ejecución busca en toe el programa dónde hay una función con ese nombre y además tiene que confirmar si está preparada para recibir 2 argumentos que el estamos pasando.

El primer valor viaja y se almacena como o si fuera una variable , en el primer parámetro, y el segundo en el segundo.

Resultado en el ejemplo: 12

Si ejecutamos la función, no amos a ver nada

¿Por qué? Porque no hemos dicho que el valor return lo imprima en pantalla, por lo que añadimos a la llamada un echo para que lo imprima.

```
function suma($num1, $num2) {  
  
    $resultado=$num1+$num2;  
  
    return $resultado;  
  
}  
  
echo (suma (5, 7) );  
+  
?>
```

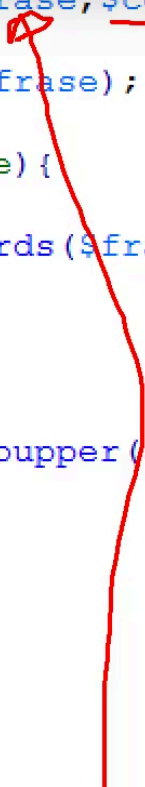
Por otro lado , con respecto al paso de parámetros, PHP tiene lo que otros lenguajes no tienes, que es el llamado PARÁMETRO POR DEFECTO.

Ejemplo para entenderlo.

en este ejemplo vamos a repasar, lo que son: funciones propias, funciones predefinidas, funciones anidadas, paso de parámetros, parámetros por defecto.

1. Partimos de un bloque vacío
2. Queremos generar una función que pase de mayúsculas a minúsculas, y además que sea capaz de convertir la primera letra de una frase en mayúscula
3. Creamos una función propia
4. Esta función recibe 2 argumentos
 - a. La llamada a la frase
 - b. Introducimos el concepto nuevo de parámetro por defecto
\$conversion=true (es un parámetro que creamos, que por defecto va a tener valor true.) A esto se le llama parámetro o argumento por defecto (característica que soporta solo php)
5. Luego dentro de la función decimos que lo que se almacena dentro de \$frase, lo pasa a minúsculas y luego lo vuelves a almacenar dentro de \$frase.

```
function frase_mayus($frase,$conversion=true){  
    $frase=strtolower($frase);  
    if($conversion==true){  
        $resultado=ucwords($frase);  
    }else{  
        $resultado=strtoupper($frase);  
    }  
    return $resultado;  
}  
  
echo(frase_mayus("esto es una prueba"));
```



se efectúa la llamada a la función con echo, con el valor “esto es una prueba”, por lo que por defecto se almacena en el parámetro \$frase)

Explicación:

Como solo le hemos pasado un argumento, la frase viaja y se almacena en argumento \$frase.

Pero no le hemos pasado un segundo argumento.

Como no se lo hemos pasado lo que hace el segundo parámetro es tomar el valor por defecto (TRUE).

Esto hace que el flujo del programa entre en el ir de la función frase_mayus, puesto que se cumple su función (\$conversion==true).

Conclusión: ejecuto y la frase estará en mayúsculas.

OJO!! Si en la llamada ponemos dos argumentos ("esto es una prueba", false), cambiamos el valor por defecto de true. Ya no es true sino false, por lo que lo que hará es pasar toda la frase a mayúsculas.

```
<?php

function frase_mayus($frase,$conversion=true){

    $frase=strtolower($frase);

    if($conversion==true){

        $resultado=ucwords($frase);

        |
    }else{

        $resultado=strtoupper($frase);

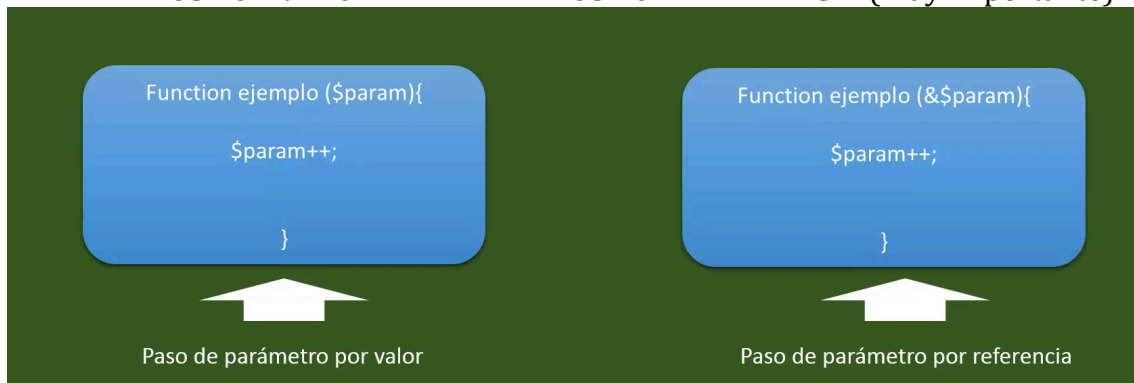
    }

    return $resultado;

}

echo(frase_mayus("esto es una prueba",false));
```

PARAMETROS POR VALOR Y PARÁMETROS POR REFERENCIA (muy importante)



Cuando pasamos los parámetro a una función, podemos hacerlo de dos maneras:

- Pasarlos por valor
- Pasarlos por referencia


```
<?php

function incrementa($valor1){

    $valor1++;

    return $valor1;

}

echo incrementa(5);

?>

</body>
</html>
```

I

Este es un ejemplo de paso de parámetro por valor (se pasa el valor 5 al parámetro \$valor1)

En el caso de parámetro por referencia la sintaxis se le añade un símbolo &, Esto implica que cuando llamemos a la función y le pasemos el valor, este será pasado por referencia.

- Si no hay ampersand por valor
- Si si lo hay por referencia

¿Qué implica pasarlo por valor y qué implica pasarlo por referencia? Veámoslo con un ejemplo

```
<?php

function incrementa($valor1){

    $valor1++;

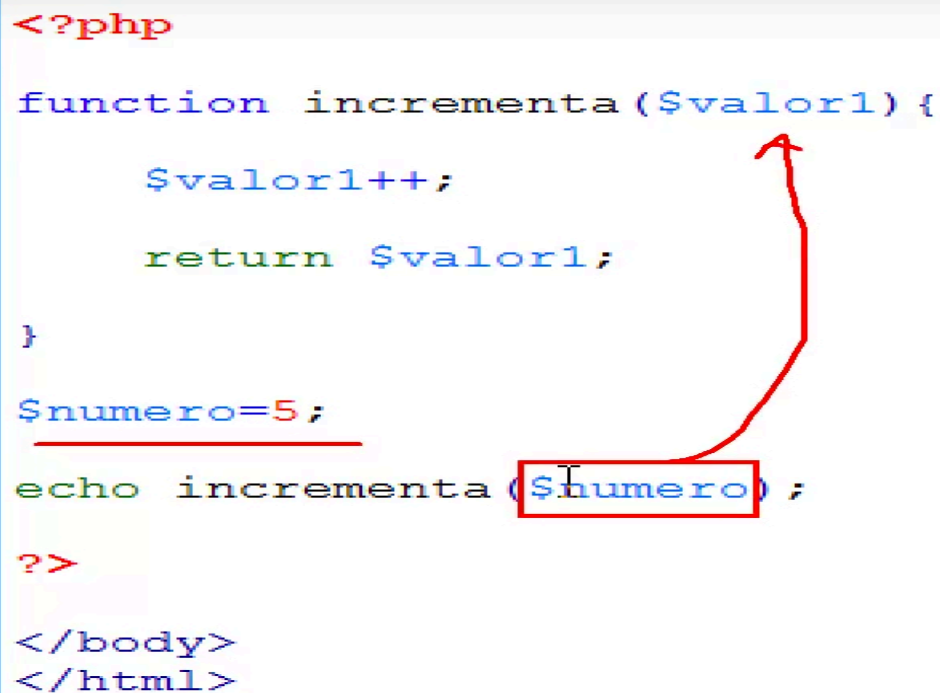
    return $valor1;

}

$numero=5;
echo incrementa($numero);

?>

</body>
</html>
```



1. creamos una función incrementa y vamos incrementando en un valor
2. imprimimos en pantalla la función a la que asignamos el valor 5, en cuanto el parámetro recibe este valor , ejecuta la siguiente línea valor 1++ =debe devolver 6
3. ahora en lugar de poner directamente el valor del parámetro en la llamada le ponemos una variable en el que se ha almacenado el valor.

se almacena en la función el valor de la variable dada

```
<?php

function incrementa(&$valor1) {

    $valor1++;

    return $valor1;

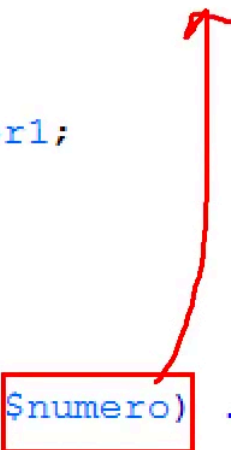
}

$numero=5;

echo incrementa($numero) . "<br>";

echo $numero;

?>
```



4. Sin el ampersand devuelve en el primer echo un 6. En el segundo un 5
5. Si incluyo el símbolo ampersand, cambio el sentido: El argumento que recibe ahora va a ser por referencia

Está claro que sin el ampersand el valor almacenado en la llamada (\$num), viaja y se almacena en \$valor.

OJO!! Sin el ampersand se pierde toda conexión entre la función y lo que hay después

Explico: Ese valor estará encapsulado en la función y la función no sabrá nada de lo que hay.

Sin embargo con el ampersand el número viaja a valor1 y se almacena en valor1. En este caso la clave está en que hay un vínculo con el origen (una referencia), al incrementar valor1 también se incrementa \$numero porque están vinculados.

Conclusión: nos devolverá 2 seises:

- El que nos devuelve la función
- Lo que hay almacenado en la variable \$numero que ha sido modificado desde dentro de la función, porque le hemos pasado el argumento por referencia.

UTILIDAD

CUANDO NECESITA QUE UNA FUNCIÓN MODIFIQUE VALORES QUE SE ENCUENTRAN FUERA DE ELLA, CONCRETAMENTE VALORES QUE LE ESTÁS PASANDO POR PARÁMETROS. BUENO PUES EN ESTE CASO LE TENDRÁ QUE PASAR EL VALOR POR REFERENCIA

OTRO EJEMPLO DE PASO POR VALOR Y PASO POR REFERENCIA

En el ejemplo el parámetro recogido queremos que lo ponga :


1. Primero en el minúsculas
2. Segundo en mayúscula la primera letra, por ultimo con return conseguimos que devuelva el \$param

```
<?php
function cambia_mayus($param){
    $param=strtolower($param);
    $param=ucwords($param);
    return $param;
}
?>
```

HOLA MUNDO

hola mundo

Hola Mundo|



luego imprimimos la función con un echo

```
<?php

function cambia_mayus($param){

    $param=strtolower($param);

    $param=ucwords($param);

    return $param;

}

$cadena="hOlA mUnDo";

echo cambia_mayus($cadena) . "<br>";

echo $cadena;

?>
```

Qué hará?

Imprimir 2 veces Hola mundo

1. La respuesta a la función hola mundo
2. Variable cadena hola mUnDo

Si ahora añadimos el ampersand, conseguimos crear un vínculo o referencia con el argumento o variable que le estamos pasando . ESTÁN LOS DOS VINCULADOS DE MANERA QUE SI MODIFICO UNO ESTOY MODIFICANDO EL OTRO

3. **Nos devolverá ahora 2 hola mundo iguales**