50 ANIVERSARIO

IES ANTONIO SEQUEROS ALMORADÍ

TEMA 5

Funciones

2020/2021

Versión: 221209.0847

IES Antonio Sequeros

ÍNDICE

1. FUNCIONES

Una función es un conjunto de instrucciones que realizan una tarea concreta, para ello pueden o no recibir parámetros (variables que se le pasan a la función para operar con ellos) y devolver o no un resultado.

Una función forma una unidad lógica del programa y cumplen varios objetivos:

- Organizar el código
- Reutilizar el código (evita duplicidad de código)

En PHP se pueden definir funciones, de manera que no haga falta escribir el mismo código varias veces. Las funciones deben definirse antes de utilizarlas (aunque no sea en el mismo fragmento de código php). Las funciones pueden no tener argumento o tener argumentos por valor o por referencia.

Sintaxis

```
function nombre ( ) {
    .....
    ... instrucciones ...
    .....
    return valoradevolver
}
```

Los nombres de las funciones siguen las mismas reglas de los identificadores de PHP, es decir, deben comenzar por una letra o un guión bajo (_) y el resto de caracteres pueden ser letras, números o guiones bajos (se pueden utilizar caracteres no ingleses como acentos, eñes, etc), pero los nombres de funciones no distinguen entre mayúsculas o minúsculas.

Ejemplo de función y de su uso:

```
<?php
// ESTA ES LA DEFINICIÓN DE LA FUNCIÓN calculaHipotenusa
function calculaHipotenusa($arg1, $arg2)
{
    $hipotenusa = sqrt($arg1 * $arg1 + $arg2 * $arg2);
    return $hipotenusa;
}


// ESTO ES UN EJEMPLO DE USO DE LA FUNCIÓN
// calculaHipotenusa

$cateto1 = 12;
$cateto2 = 16;
$hipotenusa = calculaHipotenusa($cateto1, $cateto2);

print "<p>El triángulo de lados $cateto1, $cateto2 \n";
print "y $hipotenusa es rectángulo.</p>\n";
?>
```

```
<p>El triángulo de lados 12, 16 y 20 es
      rectángulo.</p>
```

PHP no distingue entre mayúsculas y minúsculas en los nombres de las funciones.

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

Las funciones pueden devolver o no un resultado. Para que devuelvan un resultado hemos de añadir a la función la instrucción return seguida de la variable o resultado que queremos devolver cuando se invocó a la función.

1.1. Ámbito de las variables

Recordemos el concepto de variables locales y globales.

- 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.

```php
<?php
$a = 1;
$b = 2;

function Suma()
{
  global $a, $b;

  $b = $a + $b;
}

Suma();
echo $b;
?>
```

```
3
```

- 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.

```php
<?php
$a = 1;
$b = 2;

function Suma()
{
  $b = $a + $b;
}

Suma();
echo $b;
?>
```

```
2
```

1.2. Paso de variables

Tenemos dos formas de pasarlos argumentos a las funciones, dependiendo de que los valores originales de las mismas puedan ser modificados o no

- Por Valor: No modifican los valores originales
- Por Referencia: Si modifican los valores originales

Por defecto, los argumentos de las funciones son pasados por valor (así, si el valor del argumento dentro de la función cambia, este no cambia fuera de la función). Para permitir a una función modificar sus argumentos, éstos deben pasarse por referencia.

Paso de parámetros por valor

```php
<?php
function swap($a,$b)
{
    $c = $a;
    $a = $b;
    $b = $c;
}
?>
```

Paso de parámetros por referencia

```php
<?php
function swap(&$a,&$b)
{
    $c = $a;
    $a = $b;
    $b = $c;
}
?>
```

Por valor

Se hace una copia de la variable pasada como argumento, las modificaciones realizadas durante la ejecución de la función, no afectará a la variable original

Paso de parámetros por valor

```php
<?php

function swap($a,$b)
{
    $c = $a;
    $a = $b;
    $b = $c;
}

$a = 1;
$b = 2;
swap($a,$b);
echo “a = “,$a;
echo “<br>”;
echo “b = “,$b;
?>
```

Resultado

```
a = 1
b = 2
```

Por Referencia

Se pasa la referencia de la variable como argumento, es decir, es la misma dirección de memoria, por lo que los cambios que reciba durante la ejecución de la función se realizarán también en la variable original.

Paso de parámetros por referencia

```php
<?php

function swap(&$a,&$b)
{
    $c = $a;
    $a = $b;
    $b = $c;
}

$a = 1;
$b = 2;
swap($a,$b);
echo “a = “,$a;
echo “<br>”;
echo “b = “,$b;
?>
```

Resultado

```
a = 2
b = 1
```

La forma de hacerlo es esta:

- Hay que anteponer al nombre de la variable el símbolo & (signo ‘et’) 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.

```php
<?php
function añadir_algo(&$cadena)
{
    $cadena .= 'y algo más.';
}
$cad = 'Esto es una cadena, ';
añadir_algo($cad);
echo $cad;    // imprime 'Esto es una
cadena, y algo más.'
?>
```

```
Esto es una cadena, y algo más
```

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.

1.3. FUNCIONES CON ARGUMENTOS PREDETERMINADOS

En PHP se pueden definir funciones con argumentos predeterminados, de manera que si en la llamada a la función no se envía un argumento, la función asume un valor predeterminado. Lógicamente, los argumentos predeterminados deben ser los últimos en la lista de argumentos, para evitar ambigüedades.

Los argumentos predeterminados se establecen en la definición de la función, igualando el nombre del argumento a su valor predeterminado.

El ejemplo siguiente muestra una función que calcula diferentes tipos de media (aritmética, geométrica, armónica). Los argumentos de la función son los números cuya media se debe calcular y el tipo de media a calcular. En el ejemplo, el tipo de media predeterminado es la media aritmética.

```php
<?php
// ESTA ES LA DEFINICIÓN DE LA FUNCIÓN calculaMedia

function calculaMedia($arg1, $arg2, $arg3 = "aritmética")
{
    if ($arg3 == "aritmética") {
        $media = ($arg1 + $arg2) / 2;
    } elseif ($arg3 == "geométrica") {
        $media = sqrt($arg1 * $arg2) / 2;
    } elseif ($arg3 == "armónica") {
        $media = 2 * ($arg1 * $arg2) / ($arg1 + $arg2);
    }
    return $media;
}

// ESTO SON EJEMPLOS DE USO DE LA FUNCIÓN calculaMedia
$dato1 = 12;
$dato2 = 16;

// EL TERCER ARGUMENTO INDICA EL TIPO DE MEDIA A CALCULAR
$media = calculaMedia($dato1, $dato2, "geométrica");
print "<p>La media geométrica de $dato1 y $dato2 es $media.</p>\n";

// AL NO PONER EL TERCER ARGUMENTO, DEVUELVE LA MEDIA ARITMÉTICA
$media = calculaMedia($dato1, $dato2);
print "<p>La media aritmética de $dato1 y $dato2 es $media.</p>\n";
?>
```

```
<p>La media geométrica de 12 y 16 es
      6.9282032302755.</p>
<p>La media aritmética de 12 y 16 es
      14.</p>
```

1.4. BIBLIOTECAS: INCLUDE, REQUIRE, INCLUDE_ONCE, REQUIRE_ONCE

Las bibliotecas son archivos php que se pueden incluir en cualquier otro archivo php. Las bibliotecas se suelen utilizar para centralizar fragmentos de código que se utilizan en varias páginas. De esa manera, si se quiere hacer alguna modificación, no es necesario hacer el cambio en todos las páginas sino únicamente en la biblioteca.

Por ejemplo, si definimos en la biblioteca una función que imprima la cabecera de las páginas, desde cualquier página se puede incluir la biblioteca mediante la construcción include y llamar a la función como si se hubiera definido en la propia página:

biblioteca.php

```
<?php
function cabecera($titulo) {
    print "<!DOCTYPE html>\n";
    print "<html lang=\"es\">\n";
    print "<head>\n";
    print "  <meta charset=\"utf-8\" />\n";
    print "  <title>$titulo</title>\n";
    print "  <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\" />";
    print "  <link rel=\"stylesheet\" type=\"text/css\" href=\"estilo.css\"/>\n";
    print "</head>\n";
    print "\n";
    print "<body>\n";
    print "  <h1>$titulo</h1>\n";
    print "\n";
}
?>
```

Ejemplo1.php

```
<?php
include "biblioteca.php";
cabecera("Página de ejemplo");
print "  <p>Esta página es válida</p>";
?>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title>Página de ejemplo</title>
  <meta name="viewport" content="width=device-width,
      initial-scale=1.0" />
  <link rel="stylesheet" type="text/css"
      href="estilo.css" />
</head>

<body>
  <h1>Página de ejemplo</h1>
  <p>Esta página es válida</p>
</body>
</html>
```

Ejemplo2.php

```
<?php
include "biblioteca.php";
cabecera("Otra página de ejemplo");
print "  <p>Esta página también es válida</p>";
?>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title>Otra página de ejemplo</title>
  <meta name="viewport" content="width=device-width,
      initial-scale=1.0" />
  <link rel="stylesheet" type="text/css"
      href="estilo.css" />
</head>

<body>
  <h1>Otra página de ejemplo</h1>

  <p>Esta página también es válida</p>
</body>
</html>
```

Se pueden crear todas las bibliotecas que se necesiten e incluir cualquier número de bibliotecas en cualquier punto de un programa. Las bibliotecas pueden a su vez incluir otras bibliotecas.

Normalmente, las bibliotecas suelen contener funciones, definiciones de constantes o inicialización de variables, pero en realidad pueden incluir cualquier tipo de código php, que se ejecutará en la posición en la que se incluya la biblioteca.

Existe una construcción similar a include, la construcción require. La diferencia con respecto a include es que require produce un error si no se encuentra el archivo (y no se procesa el resto de la página), mientras que include sólo produce un aviso (y se procesa el resto de la página).

En un mismo archivo php se pueden incluir varias construcciones include o require, pero si las bibliotecas incluidas contienen definiciones de funciones, al incluir de nuevo la definición de la función se genera un error.

Para que no ocurra este problema se pueden utilizar las funciones include_once o require_once , que también incluyen los ficheros pero que, en caso de que los ficheros ya se hayan incluido, entonces no los incluyen.

Cuestiones

1. Realizar los ejercicios de: IAW_T05_Funciones - Practicas 01 - Funciones 01