Funciones predefinidas PHP

Variables: funciones relacionadas

Algunas funciones de PHP relacionadas con las variables:

- **isset():** devuelve true si la variable que se pasa como parámetro existe y false en caso contrario.

```php
<?php
$DNI = "8868543-Z";
if (isset($DNI)) {
    echo "La variable DNI existe!!!</br>";
}
else
    echo "La variable DNI NO existe!!!</br>";

if (isset($apellidos)) {
    echo "La variable apellidos existe!!!</br>";
}
else{
    echo "La variable  apellidos NO existe!!!</br>";
}
?>
```

Variables: funciones relacionadas

- **unset():** libera la memoria ocupada por una variable, destruyendo su nombre y contenido.

```php
<?php
$DNI = "8868543-Z";
if (isset($DNI)) {
    echo "La variable DNI existe!!!</br>";
}
else{
    echo "La variable DNI NO existe!!!</br>";
}
unset($DNI);
if (isset($DNI)) {
    echo "La variable DNI existe!!!</br>";
}
else{
    echo "La variable DNI NO existe!!!</br>";
}
?>
```

Variables: funciones relacionadas

- **empty():** comprueba si una variable está vacía, no existe o su valor es 0.

```php
<?php
    if (empty($nombre)) {
        echo ("La variable nombre no existe</br>");
    }
    $numero_entero = 0 ;
    if (empty($numero_entero)) {
        echo ("La variable numero_entero no existe o tiene el valor 0</br>") ;
    }

?>
```

Variables: funciones relacionadas

- **gettype():** devuelve el tipo de dato almacenado en la variable Puede ser: *integer, double, string, boolean array, object, class, unknown type, etc.*

```php
<?php
    $correo = "luis@nccextremadura.org";
    if (settype($correo,"integer")) {
       echo ("Variable correo convertida a Entero<br>");
    }
    else {
        echo ("Imposible convertir al tipo Entero<br>") ;
    }
    echo ("Valor actual de correo es $correo<br>");

    $nota = 4.5;
    settype($nota,"integer");
    echo "La nota es $nota y el tipo es ".gettype($nota)."<br>";
    settype($nota,"string");
    echo "La nota es $nota y el tipo es ".gettype($nota)."<br>";
?>
```

Variables: funciones relacionadas

- **settype():** convierte la variable que se pasa como parámetro al especificado en la función.
- Devuelve true si se ha realizado con éxito y false en caso contrario.
- Tipos que permite:
  - booleano o bool
  - integer o int
  - float o double
  - string
  - array
  - object
  - null

```php
<?php
    $correo = "luis@nccextremadura.org";
    if (settype($correo,"integer")) {
       echo "Variable correo convertida a Entero<br>";
       echo "El tipo ahora es ".gettype($correo)." <br>";
       //Cadenas sin caracteres numéricos se transforma en 0
    }
    else {
        echo ("Imposible convertir al tipo Entero<br>") ;
    }
    echo ("Valor actual de correo es $correo<br>");

    $nota = 4.5;
    settype($nota,"integer");
    echo "La nota es $nota y el tipo es ".gettype($nota)."<br>";
    settype($nota,"string");
    echo "La nota es $nota y el tipo es ".gettype($nota)."<br>";
?>
```

Variables: funciones relacionadas

- Otra forma de convertir los datos a otro tipo es usando el typecasting o forzado de tipos:
  - (int)/(integer)
  - (bool)/(boolean)
  - (float)/(double)
  - (string)
  - (array): *(ya se verá)*
  - (object): *(ya se verá)*
  - (unset): *fuerza a null*

Variables: funciones relacionadas

```php
<?php
    $str1 = "valor5";
    $str2 = "4valor";
    $str3 = "867";
    $str4 = "867.56";
    $str5 = "867.56";
    $bool1 = true;
    $int1 = 567;
    $float1 = 55.89;

    // Realizar la conversión:
    $var3 = (int) $str3;
    $var4 = (int) $str4;
    $var5 = (float) $str5;
    $var6 = (string) $bool1;
    $var7 = (string) $int1;
    $var8 = (string) $float1;

    echo "\$str1 ='$str1' convertido a int vale ".(int)$str1."<br/>";
    echo "\$str2 ='$str2' convertido a int vale ".(int)$str2."<br/>";
    echo "\$str3 ='$str3' convertido a int en una variable vale $var3 y su tipo es ".gettype($var3)."<br/>";
    echo "\$str4 ='$str4' convertido a int en una variable vale $var4 y su tipo es ".gettype($var4)."<br/>";
    echo "\$str5 ='$str5' convertido a float en una variable vale $var5 y su tipo es ".gettype($var5)."<br/>";
    echo "\$bool1 = $bool1 convertido a string en una variable vale $var6 y su tipo es ".gettype($var6)."<br/>";
    echo "\$int1 = $int1 convertido a string en una variable vale $var7 y su tipo es ".gettype($var7)."<br/>";
    echo "\$float1 = $float1 convertido a string en una variable vale $var8 y su tipo es ".gettype($var8)."<br/>";
?>
```

Variables: funciones relacionadas

- **is_integer(), is_double(), is_string():** estas funciones devuelven true si la variable pasada como parámetro coincide con el tipo que indica la función.
- **intval(), doubleval(), strval():** convierte el valor de una variable al tipo indicado en la función. (No sirve para objetos o array).

Arrays: funciones relacionadas

- Para mostrar todo el contenido de los arrays PHP nos ofrece dos funciones:

```
echo "<pre>";
```

  - La función **print_r()** muestra todo el contenido del array que le pasamos como parámetro, mostrando las claves y los elementos.

```
echo "</pre>";
```

  - La función **var_dump()** además de mostrar las claves y los elementos del array, muestra su tipo.

Nota: estas funciones se utilizan mucho en depuración de errores.
También sirven para mostrar objetos y variables simples.

Arrays: funciones relacionadas

- Ejemplo

```php
<?php
//Ejemmplo uso print_r y var_dump
    $modulos = array("Ingles"=>"Paloma", "EIE"=>"Pablo", "DIW" => "???",
                     "DAW"=>"Laura", "DWEC" =>"Belén", "DWES"=>"Ana");

    print_r($modulos);
    echo "<hr>";

    var_dump($modulos);
```

```
Array ( [Ingles] => Paloma [EIE] => Pablo [DIW] => ??? [DAW] => Laura
[DWEC] => Belén [DWES] => Ana )
```

```
C:\xampp\htdocs\DWES_P_U1\11_arrays_02_foreach_profesores_print_R.php:27:
array (size=6)
  'Ingles' => string 'Paloma' (length=6)
  'EIE' => string 'Pablo' (length=5)
  'DIW' => string '???' (length=3)
  'DAW' => string 'Laura' (length=5)
  'DWEC' => string 'Belén' (length=6)
  'DWES' => string 'Ana' (length=3)
```

Arrays: funciones relacionadas

```php
$poblacion["España"]["Castilla y León"]["Palencia"]=80000;
$poblacion["España"]["Castilla y León"]["Valladolid"]=370000;
$poblacion["España"]["Asturias"]["Oviedo"]=115000;
$poblacion["Alemania"]["Branderburgo"]["Berlín"]=40000;
var_dump($poblacion);
echo "<hr><br/>";
echo "elementos del array poblacion count " .count($poblacion)."<br>";
echo "todos elementos del array poblacion "
    . "COUNT_RECURSIVE " .count($poblacion,COUNT_RECURSIVE)."<br>";
echo "elementos del array poblacion size " .sizeof($poblacion)."<br>";
```

```
C:\xampp\htdocs\DWES_P_U1\11_arrays_03_multidimensionales.php:75:
array (size=2)
  'España' =>
    array (size=2)
      'Castilla y León' =>
        array (size=2)
          'Palencia' => int 80000
          'Valladolid' => int 370000
      'Asturias' =>
        array (size=1)
          'Oviedo' => int 115000
  'Alemania' =>
    array (size=1)
      'Branderburgo' =>
        array (size=1)
          'Berlín' => int 40000
```

---

elementos del array poblacion count 2
todos elementos del array poblacion COUNT_RECURSIVE 9
elementos del array poblacion size 2

Arrays: funciones relacionadas

- PHP cuenta con muchas funciones para el manejo de arrays. Vamos a ver algunas de ellas.
- **count():** devuelve el número de elementos de un array u objeto. LA función **sizeof()** es un alias de count().
- Si el array no está inicializado devolverá null.
- Por defecto no cuenta el número de elementos en un array multidimensional. Para que lo haga, hay que indicarle que cuente de forma recursiva con el parámetro COUNT_RECURSIVE.

https://www.php.net/manual/es/ref.array.php

Arrays: funciones relacionadas

- **in_array():** busca en un array el valor pasado como parámetro.
- Devuelve true si lo encuentra o false en caso contrario.
- La búsqueda es sensible a mayúsculas y minúsculas.
- Si el tercer parámetro es “true” (por defecto false) también compara los tipos de los datos.
- El elemento a buscar puede ser un array.
- No funciona en matrices multidimensionales. Para buscar en un array de más de una dimensión se puede hacer:
  - Crear una función recursiva.
  - Si sabemos en qué columna está el valor que buscamos podemos utilizar la función array_ columna, que devuelve los valores del array de la columna indicada.

https://www.php.net/manual/es/ref.array.php

Arrays: funciones relacionadas

- Ejemplo de búsqueda estricta

```php
echo "<h2>Comprobar con parámetro strict a TRUE</h2>";
$a = array('1.10', 12.4, 1.13);
if (in_array('12.4', $a, true)) {
    echo "'12.4' Encontrado con chequeo STRICT\n<br/>";
}
else
    echo "'12.4' no encontrado con chequeo STRICT en el array es un nº no un string\n<br/>";
if (in_array('12.4', $a)) {
    echo "'12.4' Encontrado SIN chequeo STRICT, no comprueba los tipos\n<br/>";
}
else
    echo "'12.4' no encontrado con chequeo STRICT en el array es un nº no un string\n<br/>";
if (in_array(1.13, $a, true)) {
    echo "1.13 Encontrado con chequeo STRICT\n<br/>";
}
```

En el primer ejemplo hace una búsqueda estricta de la cadena '12.4' en el array. El resultado es false porque no son del mismo tipo.
En el segundo ejemplo hace una búsqueda no estricta de la cadena '12.4' en el array. El resultado es true.
En el tercer ejemplo hace una búsqueda estricta del número 1.13. El resultado es true.

Comprobar con parámetro strict a TRUE

'12.4' no encontrado con chequeo STRICT en el array es un nº no un string
'12.4' Encontrado SIN chequeo STRICT, no comprueba los tipos
1.13 Encontrado con chequeo STRICT

Arrays: funciones relacionadas

- Ejemplo de búsqueda de un array

```php
$a = array(array('DWES', 'DWEC'), array('DAW', 'DIW'), 'IEE');

if (in_array(array('DWES', 'DWEC'), $a)) {
    echo "Se encontró 'DWES,DWEC'<br/>\n";
}
if (in_array(array('DAE', 'DIW'), $a)) {
    echo "Se encontró 'DAE'<br/>\n";
}
else{
    echo "No se encontró 'DAE'<br/>\n";
}
if (in_array('DAE', $a)) {
    echo "Se encontró 'DAE'<br/>\n";
}
else
{
    echo "No se encontró 'DAE'<br/>\n";
}
if (in_array('IEE', $a)) {
    echo "Se encontró 'IEE'<br/>";
}
```

En el primer ejemplo busca en el array $a el array “DWES, DWEC” y el resultado es true (es el primer elemento del array $a.
En el segundo ejemplo busca el array ”DAE,”DIW en el array $a. El resultado es false porque DAE no lo encuentra.
En el tercer ejemplo busca la cadena “DAE” en el array $a, el resultado es false.
En el cuarto ejemplo, busca el a texto ‘IEE’ en el array $a, el resultado es true.

Utilizar un array como valor a buscar

Se encontró 'DWES,DWEC'
No se encontró 'DAE'
No se encontró 'DAE'
Se encontró 'IEE'

Arrays: funciones relacionadas

- Ejemplo de búsqueda en array multidimensional

El resultado de buscar la cadena “Susana” en el array bidimensional es false, la función in_array no realiza este tipo de búsquedas

No ha encontrado a Susana

```php
$agenda = array(array("Nombre"=>"Jorge",
                      "Dirección"=>"Madrid",
                      "Telefono"=>999999999,
                      "Correo"=>"jorge@correo.com"),
                array("Nombre"=>"Julia",
                      "Dirección"=>"Valencia",
                      "Telefono"=>235456987,
                      "Correo"=>"julia@correo.com"),
                array("Nombre"=>"Lucas",
                      "Dirección"=>"Orense",
                      "Telefono"=>222222222,
                      "Correo"=>"lucas@correo.com"),
                array("Nombre"=>"Susana",
                      "Dirección"=>"Ávila",
                      "Telefono"=>987546321,
                      "Correo"=>"susana@correo.com"),
    );
// no funciona en arrays recursivos
if(in_array("Susana",$agenda))
{
    echo "Ha encontrado a Susana<br/>";
}
else{
    echo "No ha encontrado a Susana<br/>";
}
```

Arrays: funciones relacionadas

- Ejemplo de búsqueda en array multidimensional

Una forma de buscar en un array multidimensional es usando la función array_columna() que devuelve un array con los valores de una columna de un array que se indicará mediante su clave. En este ejemplo devuelve un array con los valores de la columna con clave “Nombre” del array.
Ese array es el que indicamos en la búsqueda de la cadena “Susana” en la función in_array().
El resultado de la búsqueda es true.

```php
$agenda = array(array("Nombre"=>"Jorge",
                      "Dirección"=>"Madrid",
                      "Telefono"=>999999999,
                      "Correo"=>"jorge@correo.com"),
                array("Nombre"=>"Julia",
                      "Dirección"=>"Valencia",
                      "Telefono"=>235456987,
                      "Correo"=>"julia@correo.com"),
                array("Nombre"=>"Lucas",
                      "Dirección"=>"Orense",
                      "Telefono"=>222222222,
                      "Correo"=>"lucas@correo.com"),
                array("Nombre"=>"Susana",
                      "Dirección"=>"Ávila",
                      "Telefono"=>987546321,
                      "Correo"=>"susana@correo.com"),
    );
if(in_array('Susana', array_column($agenda, 'Nombre'))){
    echo "Ha encontrado a Susana<br/>";
}
else
    echo "No ha encontrado a Susana<br/>";
```

Ha encontrado a Susana

Arrays: funciones relacionadas

- **unset():** permite borrar un array completamente o un elemento del mismo, dependiendo de si le pasamos como parámetro el nombre del array o un elemento respectivamente.
- **shuffle(array):** reordena de forma aleatoria un array.
- **array_rand(array):** devuelve un índice aleatorio de un array.
- **array_fill(inicio, n, valor):** devuelve un array con n elementos todos inicializados con el valor indicado. El parámetro “inicio” indica el primer índice del array (es numérico), el resto de los índices serán números a partir de él.
- **array_unique():** elimina valores duplicados.

Arrays: funciones relacionadas

- **array_push():** inserta un elemento al final del array. Para añadir un elemento al final del array también se puede usar $array[]=valor; este último método es más recomendable puesto que evitamos llamar a una función.
- **array_unshift():** inserta un elemento al principio del array.
- **array_shift():** borraremos el primer elemento del array.
- **array_pop():** borramos el último. Podemos guardar en una variable el elemento que se ha borrado en ambos casos.
- **array_splice():** permite tanto insertar como borrar.
    - **Para insertar:** en el primer parámetro indicamos el array, en el segundo parámetro decimos en qué posición queremos insertar, en el tercero un 0 (es el que se usa para borrar), y en el cuarto el valor que queremos poner en dicha posición.
    - **Para borrar:** si en el tercer parámetro indicamos un valor distinto de 0 estamos señalando cuántos elementos queremos borrar.

Arrays: funciones relacionadas

```
$aDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");
// En la posición 2 (param2) insertamos el valor 88, el 0 indica que insertamos.
array_splice( $aDias, 2, 0, 88 );
// El array quedaría: "Lunes", "Martes", 88, "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo";

//Borramos el elemento en posición 2, el 1 indica cuantos elementos borramos y por qué  //valor lo sustituimos
array_splice($aDias, 2, 1, "DWES" );
// El array quedaría: "Lunes", "Martes", "DWES", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo";

// Desde la posición 2 borramos 2 elementos (el actual y el siguiente, las posiciones 2 y 3)
array_splice( $aDias, 2, 2);
// El array queda: "Lunes", "Martes", "Jueves", "Viernes", "Sábado", "Domingo"
```

Arrays: funciones relacionadas

- **sort()/rsort():** ordenan de forma ascendente/descendente un array.
- **asort()/arsort():** ordenan de forma ascendente/descendente un array asociativo.
- **array_reverse():** ordena en orden inverso un array, es decir el elemento con el último índice pasa a ser el primero.
- **array_multisort():** permite ordenar varios arrays al mismo tiempo indicando cada array y opcionalmente la forma de ordenarlo.
- **array_keys():** devuelve un array con las claves de un array asociativo.
- **array_keys_exists():** permite averiguar si un array asociativo contiene una clave dada.

Arrays: funciones relacionadas

- **Ejemplo array_multisort**

```php
$array1 = array(10, 100, 200, 0);
$array2 = array(3, 1, 2, 4);
array_multisort($array1, $array2);
var_dump($array1);
var_dump($array2);
```

```
0 => int 0
1 => int 10
2 => int 100
3 => int 200
```

Para empezar, los arrays tienen que tener la misma cantidad de contenido.
Al ordenar el $array1, se ordena el array2 a la vez,
es decir, para ordenar de forma ascendente, el primer número sería el 0 en el array1,
y en el array2, su contenido correspondiente sería el 4 y se colocaría en primera posición junto al 0.

```
0 => int 4
1 => int 3
2 => int 1
3 => int 2
```

Los arrays tienen que tener el mismo número de elementos.
Se ordena el primer array, y los demás se ordenan en función del orden establecido para el primero.
En el ejemplo, el elemento 0 del primer array pasa de la última posición a la primera, y por tanto el último elemento del segundo array también pasa a la primera posición.
El elemento del primer array con valor 10 pasa a ocupar la posición 1, en el segundo array pasa lo mismo.

Arrays: funciones relacionadas

- **Ejemplo array_multisort**

```php
$miArray = array(array("10", 11, 100, 100, 100, 100, "a"),
                 array(   1,  2, "2",   3,   4,   8,   1));
array_multisort($miArray[0], SORT_ASC, SORT_STRING,
                $miArray[1], SORT_NUMERIC, SORT_DESC);
var_dump($miArray);
```

```
0 =>
  array (size=7)
    0 => string '10' (length=2)
    1 => int 100
    2 => int 100
    3 => int 100
    4 => int 100
    5 => int 11
    6 => string 'a' (length=1)
1 =>
  array (size=7)
    0 => int 1
    1 => int 8
    2 => int 4
    3 => int 3
    4 => string '2' (length=1)
    5 => int 2
    6 => int 1
```

Todos los arrays se ordenan en función del primero, y si dos o más elementos del array son idénticos, se ordenan en función del segundo array dado, y así con los demás arrays.
Si hay array en el que coinciden valores, en ese caso los valores iguales del primer array se ordenan en función de las directrices del segundo.
El primer subarray se ordena de forma ascendente comparando los elementos como cadenas.
El segundo subarray se ordena con el orden del primero, pero si coinciden valores, se ordenan de forma numérica y orden descendente (Por ejemplo los elementos con valor 100)

Arrays: funciones relacionadas

Usamos la instrucción array_keys para obtener las claves del primer array y luego usamos un bucle for para ir accediendo a cada elemento mediante su clave **(útil si no conocemos las claves)**

- **Ejemplo array_keys()**

```
$superheroes = array("spiderman" => array("name" => "Peter Parker",
                                          "email" => "peterparker@mail.com"),
                     "superman" => array("name" => "Clark Kent",
                                         "email" => "clarkkent@mail.com"),
                     "ironman" => array("name" => "Tony Stark",
                                        "email" => "tonystark@mail.com"));
$keys = array_keys($superheroes); //contiene spiderman, superman e ironman

for($i = 0; $i < count($superheroes); $i++) {
      echo "<b>$keys[$i]: </b><br>";
      foreach($superheroes[$keys[$i]] as $clave => $value) {
          echo $clave . " : " . $value . "<br>";
      }
}
```

spider-man:
name : Peter Parker
email : peterparker@mail.com
super-man:
name : Clark Kent
email : clarkkent@mail.com
iron-man:
name : Tony Srark
email : tonystark@mail.com

Arrays: funciones relacionadas

- **array_slice():** devuelve los elementos del array a partir de la posición indicada. En el segundo parámetro se indica la posición a partir de la cual se extraerán los elementos, y en el tercer parámetro cuantos elementos se van a extraer. Si se omite se extraen hasta el final del array.
- **array_diff():** devuelve un nuevo array con los valores en el primer array que no estén en el resto de arrays pasados como parámetros
- **array_diff_key():** devuelve un array con las claves del primer array pasado en el primer parámetro que no están en ninguno de los otros. Si cambiamos el orden de los arrays en la llamada a la función, cambia el resultado.

Arrays: funciones relacionadas

- **implode():** extrae los valores contenidos en un array en formato de cadenas de texto especificando en el primer parámetro el texto delimitador entre ellos (si no se indica nada será una cadena vacía).
- **explode():** obtiene un array a partir de una cadena de texto, indicando en el primer parámetro el carácter separador.
- **str_split():** crea un array a partir de una cadena pudiendo especificar el nº de caracteres que va a haber en cada elemento del array. Si no se especifica nada cada elemento contendrá un carácter incluyendo los espacios.

Arrays: funciones relacionadas

- **Ejemplo str_split():**

```php
$nombres = "Cursos gratis";
echo "usamos la función str_split() para crear un array a partir de la cadena '$nombres' "
        . "en el que cada elemento contendrá un carácter EL ESPACIO ES EL ELEMENTO 6<br>";
$aCaracteres1 = str_split( $nombres );
// Devuelve: Array ( [0] => C [1] => u [2] => r [3] => s [4] => o [5] => s
// [6] => [7] => g [8] => r [9] => a [10] => t [11] => i [12] => s )
print_r( $aCaracteres1 );
echo "<br>usamos la función str_split() para crear un array a partir de la cadena '$nombres' "
        . "en el que cada elemento contendrá  4 caracteres<br>";
$aCaracteres2 = str_split( $nombres, 4 );
// Devuelve: Array ( [0] => Curs [1] => os g [2] => rati [3] => s )
print_r( $aCaracteres2 );
```

usamos la función str_split() para crear un array a partir de la cadena 'Cursos gratis' en el que cada elemento contendrá un carácter EL ESPACIO ES EL ELEMENTO 6
Array ([0] => C [1] => u [2] => r [3] => s [4] => o [5] => s [6] => [7] => g [8] => r [9] => a [10] => t [11] => i [12] => s)
usamos la función str_split() para crear un array a partir de la cadena 'Cursos gratis' en el que cada elemento contendrá 4 caracteres
Array ([0] => Curs [1] => os g [2] => rati [3] => s)

Strings: funciones relacionadas

Buscar:

- **strlen():** devuelve la longitud de la cadena en número de caracteres. No tiene en cuenta el carácter fin de cadena como en otros lenguajes.
- **strpos():** Devuelve la posición (comenzando por el 0) de un determinado carácter dentro de una cadena. Diferencia entre mayúsculas y minúsculas.
- **stripos():** Igual que strpos pero ignora mayúsculas y minúsculas.
- **strrpos():** busca comenzando por el final de la cadena.

Strings: funciones relacionadas

Comparar:

- Las cadenas se pueden comparar usando los comparadores de igualdad == o ===, pero solo resultará cierta la comparación si son exactamente iguales.
- **strcmp():** compara bit a bit dos cadenas de caracteres. Si el resultado es 0 las cadenas son iguales, si es menor de 0 la primera cadena es menor que la segunda y si es mayor de 0 la primera cadena es mayor que la segunda (funciona con palabras con caracteres propios del español).
- La comparación se hace en función de los códigos ASCII. Diferencia entre mayúsculas y minúsculas. Para que no se tenga en cuenta usar **strcasecmp**.

Strings: funciones relacionadas

Mayúsculas y minúsculas:

- **strtolower()/ strtoupper():** convierte a minúsculas o mayúsculas respectivamente una string.
- **lcfirst()/ucfirst():** convierte el primer carácter se una string a minúscula o mayúscula respectivamente.
- **ucwords():** convierte a mayúsculas la primera letra de cada palabra de una cadena.

Extraer texto:

- **substr(cadena, inicio, tamaño):** extrae de la cadena que se pasa como parámetro la subcadena que comienza en la posición indicada (empezando a contar desde 0) el número de caracteres indicado en el parámetro “tamaño”, si no se indica, se extrae hasta el final.

Strings: funciones relacionadas

Sustituir cadenas:

- **str_replace(txt_buscar,txt_sustituto,cadena) :** sustituye en la cadena pasada como parámetro todas las apariciones del texto que se busca por el texto sustituto.

- Hay muchas más funciones relacionadas con las cadenas de caracteres.

Strings: funciones relacionadas

- Cuando estamos trabajando con caracteres propios del español, algunas de estas funciones no son las adecuadas.
- Por ejemplo: la palabra “cigüeña” tiene 7 caracteres, sin embargo, la función **strlen(“cigüeña”)** nos devuelve el número 9.
- Esto ocurre porque los caracteres extraños como la “ñ” o la “ü” ocupan más de un byte (caracteres multibyte => se necesitan dos o más bytes consecutivos para representar un único carácter).

```
el tamaño con strlen de cigüeña= 9
0: c
1: i
2: g
3: 
4: 
5: e
6: 
7: 
8: a
```

En el ejemplo vemos que el tamaño de la palabra “cigüeña con la función strlen es 9
A continuación se muestra la palabra carácter a carácter usando un bucle.

Strings: funciones relacionadas

- Para trabajar con los caracteres multibyte existen en PHP funciones especiales que empiezan por **mb_**.
- Por ejemplo, para obtener el número de caracteres de la palabra “cigüeña podemos usar la función **mb_strlen(“cigüeña”)**.
- Esto ocurre porque los caracteres extraños como la “ñ” o la “ü” ocupan más de un byte (caracteres multibyte => se necesitan dos o más bytes consecutivos para representar un único carácter).

el tamaño con mb_strlen de cigüeña= 7

0: c-99
1: i-105
2: g-103
3: -
4: -
5: e-101
6: -

En el ejemplo vemos que el tamaño de la palabra “cigüeña con la función mb_strlen es 7
Al mostrar la palabra carácter a carácter, se siguen viendo mal.
El número que se ve en el ejemplo corresponde con el código ASCII del carácter

Strings: funciones relacionadas

- Si quisiéramos mostrar carácter a carácter una cadena multibyte podemos utilizar la función preg_split()):

preg_split('//u', 'cigüeña','UTF-8', null, PREG_SPLIT_O_EMPTY)

- Esta función divide una cadena en un array utilizando una expresión regular. En este caso, la expresión regular /u indica que la cadena se trate como UTF-8, por tanto, en cada elemento del array habrá un carácter de la cadena.
- Esto resulta útil si tenemos que mostrar o comparar cadenas carácter a carácter, por ejemplo en un juego del ahorcado. El parámetro null indica que la función debe devolver todos los substrings sin límite de nº, y el último parámetro indica que nos devuelva los elementos no vacíos.

0: c-99
1: i-105
2: g-103
3: ü-252
4: e-101
5: ñ-241
6: ó-243
7: n-110

Usando preg_split obtenemos un array en el que cada elemento corresponde a un carácter.
El número que se ve en el ejemplo corresponde con el código ASCII del carácter

Strings: funciones relacionadas

Algunas funciones multibyte:

- **mb_strtolower()/mb_strtoupper():** convierte a minúsculas o mayúsculas respectivamente una string con caracteres multibyte.

```
$cadena1="cigüeñón";
$cadena2="CigÜEÑÓN";
$cadena3="Cigüeñón";
```

```
$cadena1=strtolower($cadena1);
$cadena2= strtolower($cadena2);
$cadena3= strtolower($cadena3);
```

strlower(cadenacadena1) cigüeñón, strlower(cadena2)= cigÜeÑÓn, strlower(cadena3)= cigüeñón

La función strtolower() no convierte los caracteres multibyte

```
$cadena1= mb_strtolower($cadena1, "UTF-8");
$cadena2= mb_strtolower($cadena2, "UTF-8");
$cadena2= mb_strtolower($cadena3, "UTF-8");
```

mb_strlower(cadenacadena1) cigüeñón, mb_strlower(cadena2)= cigüeñón, mb_strlower(cadena3)= cigüeñón

Strings: funciones relacionadas

Algunas funciones multibyte:

- **mb_substr():** extrae una subcadena desde la posición que le indiquemos.

```
$cadena4 = "cigüeñón";
echo "<h2>seleccionar subcadenas</h2>";
echo "<h3>Cogemos de $cadena4 4 caracteres comenzando en la posición 3</h3>";
echo "subcadena substr : ".substr($cadena4,3,4)."<br>";
echo "<h3>Cogemos de $cadena4 5 caracteres comenzando en la posición 3</h3>";
echo "subcadena de 5 caracteres desde la posición 3: substr(\"$cadena4\",3,5): ".substr($cadena4,3,5)."<br>";
echo "subcadena de 5 caracteres desde la posición 3: mb_substr(\"$cadena4\",3,5): ".mb_substr($cadena4,3,5);
```

subcadena de 5 caracteres desde la posición 3: substr("cigüeñón",3,5): üeñ
subcadena de 5 caracteres desde la posición 3: mb_substr("cigüeñón",3,5): üeñón