

Capítulo 3

Utilizar las funciones PHP

1. Preámbulo

El objetivo de este capítulo es presentar las funciones más útiles para el desarrollo de un sitio web.

PHP ofrece numerosas funciones; la descripción de cada función está disponible en línea en el sitio www.php.net.

● Versión 8

Desde la **versión 8**, es posible pasar parámetros a una función utilizando el nombre del parámetro en lugar de su posición. Esta funcionalidad se presenta en el capítulo Escribir funciones y clases PHP, pero puede utilizarse para las funciones propias del lenguaje PHP y, por tanto, para las funciones que se presentan en este capítulo. Sin embargo, en este capítulo, los nombres reales de los parámetros de las funciones no se presentan (están traducidos); para conocerlos, consulte la documentación en línea de las funciones...

Desde la **versión 8.1**, pasar el valor NULL en un parámetro que no es explícitamente opcional es obsoleto, y por lo tanto, genera una alerta E_DEPRECATED.

Ejemplo

```
<?php
$x = null;
$n = strlen($x);
?>
```

Resultado

Deprecated: strlen(): Passing null to parameter #1 (\$string) of type string is deprecated in `/app/scripts/index.php` on line 3

2. Manipular las constantes, las variables y los tipos de datos

2.1 Constantes

PHP ofrece una serie de funciones útiles sobre las constantes:

Nombre	Función
defined	Indica si una constante está definida o no.
constant	Devuelve el valor de una constante.

defined

La función `defined` permite saber si una constante está definida o no.

Sintaxis

booleano `defined(cadena nombre)`

nombre Nombre de la constante.

La función `defined` devuelve `TRUE` si la constante está definida y `FALSE` en caso contrario.

Ejemplo

```
<?php
// Probar si la constante CONSTANTE está definida.
$ok = defined('CONSTANTE');
if ($ok) {
    echo 'CONSTANTE está definida.<br />';
} else {
    echo 'CONSTANTE no está definida.<br />';
};
// Definir la constante CONSTANTE
define('CONSTANTE','valor de la CONSTANTE');
// Probar si la constante CONSTANTE está definida.
$ok = defined('CONSTANTE');
if ($ok) {
    echo 'CONSTANTE está definida.<br />';
} else {
```

```
    echo 'CONSTANTE no está definida.<br />';  
};  
?>
```

Resultado

CONSTANTE no está definida.
CONSTANTE está definida.

constant

La función constant devuelve el valor de una constante cuyo nombre se pasa como parámetro.

Sintaxis

```
mixto constant(cadena nombre)
```

Donde

nombre Nombre de la constante.

Esta función es útil para recuperar el valor de una constante cuyo nombre no se conoce a priori.

Ejemplo

```
<?php  
// definir el nombre de la constante en una variable  
$nombreConstante = 'OTRA CONSTANTE';  
// definir el valor de la constante  
define($nombreConstante,'valor de la OTRA CONSTANTE');  
// mostrar el valor de la constante  
echo $nombreConstante,' = ',constant($nombreConstante);  
?>
```

Resultado

OTRA CONSTANTE = valor de la OTRA CONSTANTE

Otras funciones permiten conocer el tipo de una constante (véase la sección Manipular las constantes, las variables y los tipos de datos - Tipos de datos).

2.2 Variables

PHP ofrece una serie de funciones útiles en las variables:

Nombre	Función
empty	Indica si una variable está vacía o no.
isset	Indica si una o varias variables están definidas o no.

Nombre	Función
unset	Elimina una o varias variables.
var_dump	Muestra la información sobre una o varias variables (tipo y valor).

empty

La función empty permite probar si una variable está vacía o no.

Sintaxis

booleano empty(*mixto variable*)

variable Variable que se va a probar.

empty devuelve TRUE si la variable está definida y FALSE en caso contrario.

Una variable se considera vacía si no ha sido asignada o si contiene una cadena vacía (""), una cadena igual a 0 ("0"), un 0, NULL, FALSE o una tabla vacía.

La función empty también se puede utilizar para probar si una expresión está vacía o no.

Ejemplo

```
<?php
// Prueba de una variable no inicializada.
$está_vacía = empty($variable);
echo '$variable no inicializada<br />';
if ($está_vacía) {
    echo '=> $variable está vacía.<br />';
} else {
    echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena vacía.
$variable = '';
$está_vacía = empty($variable);
echo '$variable = \'\'<br />';
if ($está_vacía) {
    echo '=> $variable está vacía.<br />';
} else {
    echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena igual a 0.
$variable = '0';
$está_vacía = empty($variable);
echo '$variable = \''.$variable.'\'<br />';
```

Capítulo 3

```

if ($está_vacía) {
    echo '=> $variable está vacía.<br />';
} else {
    echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene 0.
$variable = 0;
$está_vacía = empty($variable);
echo '$variable = ', $variable, '<br />';
if ($está_vacía) {
    echo '=> $variable está vacía.<br />';
} else {
    echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena no vacía.
$variable = 'x';
$está_vacía = empty($variable);
echo '$variable = \'', $variable, '\'  


```

Resultado

```

$variable no inicializada
=> $variable está vacía.
$variable = ''
=> $variable está vacía.
$variable = '0'
=> $variable está vacía.
$variable = 0
=> $variable está vacía.
$variable = 'x'
=> $variable no está vacía.

```

isset

La función `isset` permite probar si una o varias variables están definidas o no.

Sintaxis

```
booleano isset(mixto variable[,...])
```

`variable` Variable que se va a probar; pueden ser varias, separadas por una coma.

`isset` devuelve `TRUE` si la variable está definida y `FALSE` en caso contrario.

Si se facilitan varios parámetros, la función devuelve TRUE únicamente si se definen todas las variables.

Una variable se considera como no definida si no se ha visto asignada o si contiene NULL. A diferencia de la función `empty`, una variable que contiene una cadena vacía (`""`), una cadena igual a 0 (`"0"`), un 0, un `FALSE` o una tabla vacía, no se considera como no definida.

Ejemplo

```
<?php
// Prueba de una variable no inicializada.
$está_definida = isset($variable);
echo '$variable no inicializada<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena vacía.
$variable = '';
$está_definida = isset($variable);
echo '$variable = \'\'<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena igual a 0.
$variable = '0';
$está_definida = isset($variable);
echo '$variable = \''.$variable.'\'<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene 0.
$variable = 0;
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena no vacía.
$variable = 'x';
```

Capítulo 3

```
$está_definida = isset($variable);
echo '$variable = \''.$variable.'\''<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
?>
```

Resultado

```
$variable no inicializada
=> $variable no está definida.
$variable = ''
=> $variable está definida.
$variable = '0'
=> $variable está definida.
$variable = 0
=> $variable está definida.
$variable = 'x'
=> $variable está definida.
```

unset

La función unset permite eliminar una o varias variables.

Sintaxis

```
unset(mixto variable)
```

variable Variable que se va a eliminar (para eliminar varias, deben estar separadas por una coma).

Después de la eliminación, la variable se encuentra en el mismo estado que si no hubiera sido asignada. El uso de la función isset en una variable eliminada devuelve FALSE.

Ejemplo

```
<?php
// Definir una variable.
$variable = 1;
// Mostrar la variable y probar si está definida.
$está_definida = isset($variable);
echo '$variable = ',$variable.'<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
```

```
// Eliminar la variable.
unset($variable);
// Mostrar la variable y probar si está definida.
$está_definida = isset($variable);
echo '$variable = ', $variable, '<br />';
if ($está_definida) {
    echo '=> $variable está definida.<br />';
} else {
    echo '=> $variable no está definida.<br />';
}
?>
```

Resultado

```
$variable = 1
=> $variable está definida.
$variable =
=> $variable no está definida.
```

■ Observación

Al asignar un 0 o una cadena vacía a una variable, no se borra.

var_dump

La función `var_dump` muestra información sobre una o varias variables (tipo y contenido).

Sintaxis

```
var_dump(mixto variable, [, ...])
```

variable Variable que se va a mostrar (pueden ser varias, separadas por una coma).

La función `var_dump` es especialmente interesante en las fases de desarrollo.

Ejemplo

```
<?php
// mostrar la información sobre una variable no inicializada
$variable = NULL;
var_dump($variable);
// inicializar la variable con un número entero
$variable = 10;
// mostrar la información sobre una variable
echo '<br />';
var_dump($variable);
// modificar el valor (y el tipo) de la variable
$variable = 3.14; // número decimal
```


Capítulo 3

```
// mostrar la información sobre una variable
echo '<br />';
var_dump($variable);
// modificar el valor (y el tipo) de la variable
$variable = 'abc'; // cadena de caracteres
// mostrar la información sobre la variable
echo '<br />';
var_dump($variable);
?>
```

Resultado

```
NULL
int(10)
float(3.14)
string(3) "abc"
```

Para una variable no inicializada, `var_dump` devuelve `NULL`. Para un número, `var_dump` indica el tipo (`int` = entero, `float` = número decimal), seguido por el valor entre paréntesis. Para una cadena, `var_dump` indica el tipo (`string`), seguido de la longitud entre paréntesis, seguido por el valor entre comillas.

PHP también ofrece las funciones `print_r` y `var_export`, que son similares a la función `var_dump`. La función `print_r` muestra o devuelve el contenido de la variable en una forma más legible, sin mencionar el tipo de datos. La función `var_export` muestra o devuelve una cadena que ofrece un código PHP de definición de la variable.

Observación

En la sección Tipos de datos de este capítulo, estudiaremos otras funciones que permiten determinar el tipo de una variable y realizar conversiones de tipos (de número a cadena, de cadena a número...).

2.3 Tipos de datos

2.3.1 Conversiones

PHP es capaz de realizar las conversiones automáticas implícitas de tipo.

Cuando un valor/expresión se asigna a una variable, la variable pasa a ser el tipo de valor/expresión.