

Web Development 2



Departamento de Lenguajes y
Sistemas Informáticos



Universitat d'Alacant
Universidad de Alicante

Strings

Sergio Luján Mora
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Index

- Strings
- Functions

Strings (1)

- A string is series of characters
- A string literal can be specified in three different ways;
 - single quoted
 - double quoted
 - *heredoc* syntax

Strings (2)

- Single quoted ('):
 - To specify a literal single quote, you will need to escape it with a backslash (\)
 - If a backslash needs to occur before a single quote or at the end of the string, you need to double it
 - If you try to escape any other character, the backslash will also be printed!

Strings (3)

- Double quoted ("):
 - Similar to C and C++ char strings
 - Escape sequences for special characters:
`\n`, `\r`, `\t`, etc.

Strings (4)

- Escaped characters:

Sequence	Meaning
\n	Line feed (LF or 0x0A inASCII)
\r	Carriage return (CR or 0x0D in ASCII)
\t	Horizontal tab(HT or 0x09 in ASCII)
\\	Backslash
\\$	Dollar sign
\"	Double quote
\'	Single quote
\[0-7]{1,3}	A character in octal notation
\x[0-9A-Fa-f]{1,2}	A character in hexadecimal notation

Strings (5)

- Double quoted ("):
 - Variable names will be expanded :


```
$name = "Juan";
echo "My name is $name";
// Prints: My name is Juan
```
 - Arrays and objects → Enclose the variable name, indices, and brackets in curly braces:


```
$name[1] = "Juan";
echo "My name is ${name[1]}";
echo "My name is {$name[1]}";
// Prints: My name is Juan
```

Strings (6)

- *Heredoc* syntax:
 - From Perl programming language
 - Similar to double quote ("), but you don't need to escape double quote character
 - One should provide an identifier (followed by new line) after <<<, then the string, and then the same identifier to close the quotation
 - The closing identifier *must* begin in the first column of the line

Strings (7)

- *Heredoc* syntax:
 - Useful for long text fragments:

```
$cadena = <<<DELIMITADOR
text
text
text
...
text
DELIMITADOR
```

Strings (8)

	Single quote	Double quote	Heredoc
Escaped characters	Only single quote (') and backslash (\)	\n, \r, \t, \l, \\$, \", \[0-7]{1,3}, \x[0-9A-Fa-f]{1,2}	Not needed
Variables are expanded	No	Yes	Yes
Speed	Fastest	Equal	Equal

Strings (and 9)

- Concatenation operator ('.'):

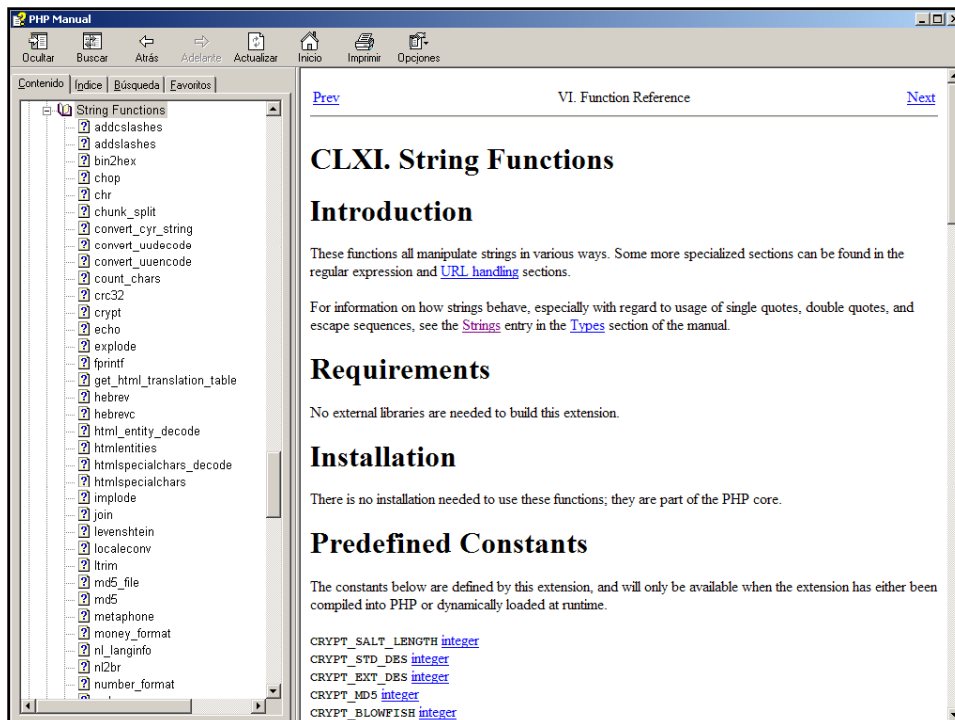

```
$cad1 = $cad1 . $cad2;
```
- Concatenating assignment operator ('.='):


```
$cad1 .= $cad2;
```
- Characters within strings may be accessed and modified by specifying the zero-based offset of the desired character after the string using square array-brackets:


```
$character5 = $cad1[4];
```

Functions (1)

- Powerful string handling and manipulating functions



The screenshot shows the PHP Manual interface. On the left, a sidebar lists various string functions under the heading "String Functions". The main content area displays the "VI. Function Reference" for "CLXI. String Functions".

CLXI. String Functions

Introduction

These functions all manipulate strings in various ways. Some more specialized sections can be found in the regular expression and [URL handling](#) sections.

For information on how strings behave, especially with regard to usage of single quotes, double quotes, and escape sequences, see the [Strings](#) entry in the [Types](#) section of the manual.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

- `CRYPT_SALT_LENGTH` [integer](#)
- `CRYPT_STD_DES` [integer](#)
- `CRYPT_EXT_DES` [integer](#)
- `CRYPT_MD5` [integer](#)
- `CRYPT_BLOWFISH` [integer](#)

Functions (2)

- `strlen($string)`: Returns the length of the given `$string`
- `strpos($string, $substring)`: Returns the numeric position of the first occurrence of `$substring` in `$string`
- `substr($string, $start, $length)`: Returns the portion of `$string` specified by the `$start` and `$length` parameters

Functions (3)

- `trim($string)`: Returns a string with whitespace stripped from the beginning and end of `$string`
 - See also `ltrim()` and `rtrim()`
- `strrev($string)`: Returns `$string` reversed
- `strtoupper($string)`: Returns `$string` with all alphabetic characters converted to uppercase
- `strtolower($string)`: Returns `$string` with all alphabetic characters converted to lowercase

Functions (4)

- `str_replace($string, $replacement, $start)`: Replaces a copy of `$string` delimited by the `$start` with the string given in `$replacement`
- `chr($int)`: Returns a one-character string containing the character specified by `$int`
- `ord($string)`: Returns the ASCII value of the first character `$string`
- `printf()`, `sprintf()`: Output a formatted string (similar to C)

Functions (and 5)

- `explode($delimiter, $string)`: Returns an array of strings, each of which is a substring of `$string` formed by splitting it on boundaries formed by the string `$delimiter`
- `implode($glue, $array)`: Join array elements with a `$glue` string
 - `join()` is an alias of `implode()`