CHAPTER 4

String

A string is a series of characters that can be stored in a variable. In PHP, strings are often delimited by single quotes.

```
$a = 'Hello';
```

String Concatenation

PHP has two string operators. The dot symbol is known as the *concatenation operator* (`.`). It combines two strings into one. It also has an accompanying assignment operator (`.=`), which appends the right-hand string to the left-hand string variable.

```
$b = $a . ' World'; // Hello World
$a .= ' World';     // Hello World
```

Delimiting Strings

PHP strings can be delimited in four different ways. There are two common notations: double quote (`" "`) and single quote (`' '`). The difference between them is that variables are not parsed in single-quoted strings, whereas they are parsed in double-quoted strings.

```
$c = 'World';
echo "Hello $c"; // "Hello World"
echo 'Hello $c'; // "Hello $c"
```

Single-quoted strings tend to be preferred unless parsing is desired, which highlights that no parsing takes place. However, double-quoted strings are considered easier to read, which makes the choice more a matter of preference. The important thing is to be consistent.

In addition to single-quoted and double-quoted strings, there are two more notations: *heredoc* and *nowdoc*. These notations are mainly used to include larger blocks of text.

M. Olsson, *PHP 7 Quick Scripting Reference*, DOI 10.1007/978-1-4842-1922-5_4

Heredoc Strings

The heredoc syntax consists of the <<< operator followed by an identifier and a new line. The string is then included followed by a new line containing the identifier to close the string. Variables are parsed inside of a heredoc string, just as with double-quoted strings.

```
$s = <<<LABEL
Heredoc (with parsing)
LABEL;
```

Nowdoc Strings

The syntax for the nowdoc string is the same as for the heredoc string, except that the initial identifier is enclosed in single quotes. Variables are not parsed inside a nowdoc string.

```
$s = <<<'LABEL'
Nowdoc (without parsing)
LABEL;
```

Escape Characters

Escape characters are used to write special characters, such as backslashes and double quotes. These characters are always preceded by a backslash (\). Table 4-1 lists the escape characters available in PHP.

Table 4-1. *The Escape Characters Available in PHP*

Character	Meaning	Character	Meaning
\n	newline	\f	form feed
\t	horizontal tab	\$	dollar sign
\v	vertical tab	\'	single quote
\e	escape	\"	double quote
\r	carriage return	\\	backslash
\u{}	Unicode character		

For example, line breaks are represented with the escape character (\n) within strings.

```
$s = "Hello\nWorld";
```

Note that this character is different from the <br> HTML tag, which creates line breaks on web pages.

```
echo "Hello<br>World";
```

When using the single quote or nowdoc delimiter, the only escape characters that work are the backslash (\\) and single-quote (\') characters. Escaping the backslash is only necessary before a single quote or at the end of the string.

```
$s = 'It\'s'; // "It's"
```

PHP 7 introduced the Unicode escape character, which provides the ability to embed UTF-8 encoded characters into strings. Such a character is specified as a hexadecimal number inside curly brackets. The number can be up to six digits long, with leading zeros being optional.

```
echo "\u{00C2A9}"; // © (copyright sign)
echo "\u{C2A9}";   // ©
```

Character Reference

Characters within strings can be referenced by specifying the index of the desired character in square brackets after the string variable, starting with zero. This can be used both for accessing and modifying single characters.

```
$s = 'Hello';
$s[0] = 'J';
echo $s; // "Jello"
```

The strlen function retrieves the length of the string argument. This can be used to change the last character of a string, for example.

```
$s[strlen($s)-1] = 'y';
echo $s; // "Jelly"
```

String Compare

The way to compare two strings is simply by using one of the equality operators. This does not compare the memory addresses, as in some other languages.

```
$a = 'test';
$b = 'test';
$c = ($a === $b); // true
```