PHP - Strings

A string is a sequence of characters, like 'PHP supports string operations.' A string in PHP as an array of bytes and an integer indicating the length of the buffer. In PHP, a character is the same as a byte. This means that PHP only supports a 256-character set, and hence does not offer native Unicode support.

PHP supports single quoted as well as double quoted string formation. Both the representations 'this is a simple string' as well as "this is a simple string" are valid. PHP also has Heredoc and Newdoc representations of string data type.

Single-Quoted String

A sequence of characters enclosed in single quotes (the character ') is a string.

```
$str = 'this is a simple string';
```

Example

If you want to include a literal single quote, escape it with a backslash (\).

Open Compiler

```
<?php
   $str = 'This is a \'simple\' string';
   echo $str;
?>
```

It will give you the following **output** –

```
This is a 'simple' string
```

Example

To specify a literal backslash, double it (\\).

Open Compiler

```
<?php
   $str = 'The command C:\\*.* will delete all files.';
   echo $str;
?>
```

Here is its **output** –

```
The command C:\*.* will delete all files.
```

Example

The escape sequences such as "\r" or "\n" will be treated literally and their special meaning will not be interpreted. The variables too will not be expanded if they appear in a single quoted string.

Open Compiler

```
<?php
   $str = 'This will not expand: \n a newline';
   echo $str . PHP_EOL;
   $x=100;
   $str = 'Value of x = $x';
   echo $str;
?>
```

It will produce the following **output** –

```
This will not expand: \n a newline
Value of x = $x
```

Double-Quoted String

A sequence of characters enclosed in double-quotes (" ") is another string representation.

```
$str = "this is a simple string";
```

Single-quoted and double-quoted strings are equivalent except for their treatment of escape sequences. PHP will interpret certain escape sequences for special characters. For example, "\r" and "\n".

Sequence	Meaning
\n	linefeed (LF or 0x0A (10) in ASCII)
\r	carriage return (CR or 0x0D (13) in ASCII)
\t	horizontal tab (HT or 0x09 (9) in ASCII)
\v	vertical tab (VT or 0x0B (11) in ASCII)
\e	escape (ESC or 0x1B (27) in ASCII)
\f	form feed (FF or 0x0C (12) in ASCII)
\\\\	backslash
\\$	dollar sign
\\"	double-quote

How to Escape Octal and Hexademical Characters in PHP?

PHP supports escaping an Octal and a hexadecimal number to its ASCII character. For example, the ASCII character for P is 80 in decimal. 80 in decimal to Octal is 120. Similarly, 80 in decimal to hexadecimal is 50.

To escape an octal character, prefix it with "\"; and to escape a hexadecimal character, prefix it with "\x".

Open Compiler

```
<?php
   $str = "\120\110\120";
   echo "PHP with Octal: ". $str;
   echo PHP_EOL;

   $str = "\x50\x48\x50";
   echo "PHP with Hexadecimal: ". $str;
?>
```

Check the **output** −

```
PHP with Octal: PHP
PHP with Hexadecimal: PHP
```

As in single quoted strings, escaping any other character will result in the backslash being printed too. The most important feature of double-quoted strings is the fact that variable names will be expanded.

Example

A double-quoted string in PHP expands the variable names (PHP variables are prefixed with $ symbol). To actually represent a "$" symbol in a PHP string, escape it by prefixing with the "\" character.

Open Compiler

```
<?php
   $price = 200;
   echo "Price = \$ $price";
?>
```

You will get the following **output** –

```
Price = $ 200
```

String Concatenation Operator

To concatenate two string variables together, PHP uses the dot (.) operator –

Open Compiler

```
<?php
   $string1="Hello World";
   $string2="1234";

   echo $string1 . " " . $string2;
?>
```

Here, you will get the following **output** –

```
Hello World 1234
```

In the above example, we used the concatenation operator twice. This is because we had to insert a third string. Between the two string variables, we added a string with a single character, an empty space, to separate the two variables.

The standard library of PHP includes many functions for string processing. They can be found at PHP’s official documentation (https://www.php.net/manual/en/ref.strings.php).

The strlen() Function

The strlen() function is used to find the length of a string.

Example

Let's find the length of our string "Hello world!" –

</> Open Compiler

```
<?php
   echo strlen("Hello world!");
?>
```

It will produce the following **output** –

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends (that is, in a loop, we would want to stop the loop after the last character in the string).

The strpos() Function

The strpos() function is used to search for a string or character within a string.

- If a match is found in the string, this function will return the position of the first match.
- If no match is found, it will return FALSE.

Example

Let's see if we can find the string "world" in our string −

Open Compiler

```
<?php
   echo strpos("Hello world!","world");
?>
```

It will produce the following **output** −

```
6
```

As you can see, the position of the string "world" in our string is "6". The reason that it is "6", and not "7", is that the first position in the string is "0", and not "1".