Syntax ¶

Constants can be defined using the `const` keyword, or by using the define()-function. While define() allows a constant to be defined to an arbitrary expression, the `const` keyword has restrictions as outlined in the next paragraph. Once a constant is defined, it can never be changed or undefined.

When using the `const` keyword, only scalar (bool, int, float and string) expressions and constant arrays containing only scalar expressions are accepted. It is possible to define constants as a resource, but it should be avoided, as it can cause unexpected results.

The value of a constant is accessed simply by specifying its name. Unlike variables, a constant is *not*prepended with a `$`. It is also possible to use the constant() function to read a constant's value if the constant's name is obtained dynamically. Use get_defined_constants() to get a list of all defined constants.

> **Note**: Constants and (global) variables are in a different namespace. This implies that for example **`true`** and *$TRUE* are generally different.

If an undefined constant is used an Error is thrown. Prior to PHP 8.0.0, undefined constants would be interpreted as a bare word string, i.e. (CONSTANT vs "CONSTANT"). This fallback is deprecated as of PHP 7.2.0, and an error of level **`E_WARNING`** is issued when it happens. Prior to PHP 7.2.0, an error of levelE_NOTICE has been issued instead. See also the manual entry on why $foo[bar] is wrong (unless `bar` is a constant). This does not apply to (fully) qualified constants, which will always raise a Error if undefined.

> **Note**: To check if a constant is set, use the defined() function.

These are the differences between constants and variables:

- Constants do not have a dollar sign (`$`) before them;
- Constants may be defined and accessed anywhere without regard to variable scoping rules;
- Constants may not be redefined or undefined once they have been set; and
- Constants may only evaluate to scalar values or arrays.

Example #1 Defining Constants

```php
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // Emits an Error: Undefined constant "Constant"
               // Prior to PHP 8.0.0, outputs "Constant" and issues a warning.
?>
```

Example #2 Defining Constants using the `const` keyword

```php
<?php
// Simple scalar value
const CONSTANT = 'Hello World';

echo CONSTANT;

// Scalar expression
const ANOTHER_CONST = CONSTANT.'; Goodbye World';
echo ANOTHER_CONST;

const ANIMALS = array('dog', 'cat', 'bird');
echo ANIMALS[1]; // outputs "cat"

// Constant arrays
define('ANIMALS', array(
    'dog',
    'cat',
    'bird'
));
```

```
echo ANIMALS[1]; // outputs "cat"
?>
```

Note:

As opposed to defining constants using define(), constants defined using the `const` keyword must be declared at the top-level scope because they are defined at compile-time. This means that they cannot be declared inside functions, loops, `if` statements or `try`/ `catch` blocks.