CHAPTER 13

Constants

A *constant* is a variable with a value that cannot be changed by the script. Therefore, such a value must be assigned at the same time that the constant is created. PHP provides two methods for creating constants: the `const` modifier and the `define` function.

Const

The `const` modifier is used to create class constants. Unlike regular properties, class constants do not have an access level specified because they are always publicly visible. They also do not use the dollar sign parser token (`$`). The naming convention for constants is all uppercase, with underscores separating each word.

```
class MyCircle
{
  const PI = 3.14;
}
```

Constants must be assigned a value when they are created. Like static properties, a constant may only be initialized with a constant value, and not with an expression. Class constants are referenced in the same way as static properties, except that they do not use the dollar sign.

```
echo MyCircle::PI; // "3.14"
```

The `const` modifier may not be applied to local variables or parameters. However, as of PHP 5.3, `const` can be used to create global constants. Such a constant is defined in the global scope and can be accessed anywhere in the script.

```
const PI = 3.14;
echo PI; // "3.14"
```


M. Olsson, *PHP 7 Quick Scripting Reference*, DOI 10.1007/978-1-4842-1922-5_13

Define

The `define` function can create both global and local constants, but not class constants. The first argument to this function is the constant's name and the second is its value.

```
define('DEBUG', 1);
```

Just like constants created with `const`, `define` constants are used without the dollar sign and their value cannot be modified.

```
echo DEBUG; // "1"
```

Like constants created with `const`, the value for `define` may be any scalar data type: integer, float, string, or bool. Unlike `const`, however, the `define` function allows an expression to be used in the assignment, such as a variable or the result of a mathematical expression.

```
define('ONE', 1);     // 1
define('TWO', ONE+1); // 2
```

Constants are case sensitive by default. However, the `define` function takes a third optional argument that may be set to `true` to create a case-insensitive constant.

```
define('DEBUG', 1, true);
echo debug; // "1"
```

To check whether a constant already exists, the `defined` function can be used. This function works for constants created with `const` or `define`.

```
if (!defined('PI'))
  define('PI', 3.14);
```

PHP 7 made it possible to create constant arrays using the `define` function. Support for constant arrays created with `const` has existed since PHP 5.6.

```
const CA = [1, 2, 3];    // PHP 5.6 or later
define('DA', [1, 2, 3]); // PHP 7 or later
```

Const and define

The `const` modifier creates a compile-time constant, so the compiler replaces all usage of the constant with its value. In contrast, `define` creates a run-time constant that is not set until run-time. This is the reason why define constants may be assigned with expressional values, whereas `const` requires constant values that are known at compile-time.

```
const PI = 3.14;   // compile-time constant
define('E', 2.72); // run-time constant
```

Only `const` may be used for class constants and only `define` for local constants. However, when creating global constants, both `const` and `define` are allowed. In these circumstances, using `const` is generally preferable, as compile-time constants are slightly faster than run-time constants. The main exception is when the constant is conditionally defined, or an expressional value is required, in which case `define` must be used.

Constant Guideline

In general, it is a good idea to create constants instead of variables if their values do not need to be changed. This ensures that the variables are not changed anywhere in the script by mistake, which in turn helps to prevent bugs.

Magic Constants

PHP provides eight predefined constants, as shown in Table 13-1. These are called *magic constants* because their values change, depending on where they are used.

Table 13-1. *Magic Constants*

Name	Description
__LINE__	Current line number of the file.
__FILE__	Full path and filename of the file.
__DIR__	Directory of the file.
__FUNCTION__	Function name.
__CLASS__	Class name including namespace.
__TRAIT__	Trait name including namespace.
__METHOD__	Class method name.
__NAMESPACE__	Current namespace.

Magic constants are especially useful for debugging purposes. For example, the value of `__LINE__` depends on the line in which it appears in the script.

```
if(!isset($var))
{
  echo '$var not set on line ' . __LINE__;
}
```