# PHP - Magic Constants

The magical constants in PHP are predefined constants. They are available to any script on which they run, and they change depending on where they are used. All these "magical" constants are resolved at compile time, unlike regular constants, which are resolved at runtime.

There are nine magical constants in PHP. These special constants are case insensitive.

## __LINE__

It returns the current line number of the file. The following **example** shows how you can use this magic constant.

</> Open Compiler

```
<?php
   $x="Hello World";
   echo "$x. The current Line number is " . __LINE__ . ".";
?>
```

It will produce the following **output** –

```
Hello World. The current Line number is 5.
```

## __FILE__

This magic constant returns the full path and filename of the file. If used inside an include, the name of the included file is returned. Take a look at the following **example** –

</> Open Compiler

```
<?php
   $x="Hello World";
   echo "$x. Current PHP script name is " . __FILE__ . ".";
?>
```

It will produce the following **output** –

```
Hello World. Current PHP script name is C:\xampp\htdocs\hello.php.
```



## __DIR__

This magical constant returns the directory of the file. If used inside an include, the directory of the included file is returned. This is equivalent to "dirname(__FILE__)". This directory name does not have a trailing slash unless it is the root directory. See the following **example** –

</> Open Compiler

```
<?php
   $x="Hello World";
   echo "$x. Directory of the Current PHP script name is " . __DIR__ . ".";
?>
```

It will show the following **output** on the browser –

```
Hello World. Directory of the Current PHP script name is C:\xampp\htdocs.
```

## __FUNCTION__

This magical constant returns the function name in which the constant is used, or {closure} for anonymous functions. The following **example** shows how it works –

</> Open Compiler

```
<?php
   function hello(){
      $x="Hello World";
      echo "$x. The function name is ". __FUNCTION__ . "";
   }
   hello();
?>
```

It will produce the following **output** –

```
Hello World. The function name is hello
```

If this magic constant is used outside the function, then it will give a blank output.

## __CLASS__

This constant returns the name of a class. The class name includes the namespace it was declared in. See the following **example** –

</> Open Compiler

```
<?php
   class myclass {
      public function __construct() {
         echo "Inside the constructor of ". __CLASS__ . PHP_EOL;
      }
      function getClassName(){
         echo "from an instance method of " . __CLASS__ . "";
      }
   }
   $obj = new myclass;
   $obj->getClassName();
?>
```

It will produce the following **output** –

```
Inside the constructor of myclass
from an instance method of myclass
```

## __METHOD__

The __METHOD__ constant returns the class method name. The following **example** shows how it works –

</> Open Compiler

```
<?php
   class myclass {
```

```
      public function __construct() {
         echo "Calling " . __METHOD__ . " of " . __CLASS__ ."<br>";
      }
      function mymethod(){
         echo "Calling " . __METHOD__ . " of " . __CLASS__ ."";
      }
   }
   $obj = new myclass;
   $obj->mymethod();
?>
```

It will produce the following **output** –

```
Calling myclass::__construct of myclass
Calling myclass::mymethod of myclass
```

## __TRAIT__

It returns the trait name. The trait name includes the namespace it was declared in. In PHP, traits are a mechanism for code reuse. A trait is similar to a class, but only intended to group functionality in a fine-grained and consistent way. It is not possible to instantiate a trait on its own.

Take a look at the following **example** –

</> Open Compiler

```
<?php
   trait mytrait {
      public function hello() {
         echo "Hello World from " . __TRAIT__ ."";
      }
   }
   class myclass {
      use mytrait;
   }
   $obj = new myclass();
   $obj->hello();
?>
```

It will produce the following **output** –

```
Hello World from mytrait
```

## __NAMESPACE__

This constant returns the name of the current namespace. In PHP, namespaces allow us to use classes / functions / constants of same name in different contexts without any conflict, thereby encapsulating these items. A namespace is a logical grouping of classes/functions depending on their relevance.

The following **example** shows how you can use this magic constant −

</> Open Compiler

```
<?php
   namespace myspace;
   class myclass {
      public function __construct() {
         echo "Name of the class: " . __CLASS__ . " in " . __NAMESPACE__ . "";
      }
   }
   $class_name = __NAMESPACE__ . '\myclass';
   $a = new $class_name;
?>
```

It will produce the following **output** −

```
Name of the class: myspace\myclass in myspace
```

## ClassName::class

Unlike the other magic constants, this magic constant does not start and end with the double underscore (__). It returns the fully qualified class name.

The following **example** shows how you can use this magic constant −

</> Open Compiler

```
<?php
   namespace myspace;
   class myclass {
```

```
      public function __construct() {
         echo "Name of the class: " . myclass::class ;
      }
   }
   use myspace;
   $a = new myclass;
?>
```

It will produce the following **output** –

```
Name of the class: myspace\myclass
```