



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)
- 

[Dutch PHP Conference 2024](#)

## [Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

## [Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Enumerations](#)

[Errors](#)

[Exceptions](#)

[Fibers](#)

[Generators](#)

[Attributes](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Predefined Attributes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

## [Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[User Submitted Data](#)

[Hiding PHP](#)

[Keeping Current](#)

## [Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)

[Using remote files](#)

[Connection handling](#)

[Persistent Database Connections](#)

[Command line usage](#)

[Garbage Collection](#)  
[DTrace Dynamic Tracing](#)

## [Function Reference](#)

[Affecting PHP's Behaviour](#)  
[Audio Formats Manipulation](#)  
[Authentication Services](#)  
[Command Line Specific Extensions](#)  
[Compression and Archive Extensions](#)  
[Cryptography Extensions](#)  
[Database Extensions](#)  
[Date and Time Related Extensions](#)  
[File System Related Extensions](#)  
[Human Language and Character Encoding Support](#)  
[Image Processing and Generation](#)  
[Mail Related Extensions](#)  
[Mathematical Extensions](#)  
[Non-Text MIME Output](#)  
[Process Control Extensions](#)  
[Other Basic Extensions](#)  
[Other Services](#)  
[Search Engine Extensions](#)  
[Server Specific Extensions](#)  
[Session Extensions](#)  
[Text Processing](#)  
[Variable and Type Related Extensions](#)  
[Web Services](#)  
[Windows Only Extensions](#)  
[XML Manipulation](#)  
[GUI Extensions](#)

## Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

[Ресурсы »](#)

[« Объекты](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Типы](#)

Change language: Russian

# Перечисления

(PHP 8 >= 8.1.0)

## Основы перечислений

Перечисления — это ограничивающий слой над классами и константами классов, предназначенный для предоставления способа определения закрытого набора возможных значений для типа.

```
<?php
enum Suit
{
    case Hearts;
    case Diamonds;
    case Clubs;
    case Spades;
}

function do_stuff(Suit $s)
{
    // ...
}

do_stuff(Suit::Spades);
?>
```

Полное описание смотрите в главе о [перечислениях](#).

## Приведение типов

Если перечисление (enum) преобразовывается в объект (object), оно не изменяется. Если перечисление (enum) преобразовывается в массив (array), то создаётся массив с одним ключом name (для простых перечислений) или массив с двумя ключами name и value (для типизированных перечислений). Все остальные приведения типов приведут к ошибке.

[+add a note](#)

## User Contributed Notes 1 note

[up](#)

[down](#)

2

[esdras-schonevald](#) ¶

1 year ago

<https://gist.github.com/esdras-schonevald/71a6730e6191c5e9c053e2f65b839eec>

```
<?php

declare(strict_types=1);

/**
 * This is a sample
 * How to use Enum to create a custom exception cases
 * PHP 8.1^
 */

enum MyExceptionCase {
    case InvalidMethod;
    case InvalidProperty;
    case Timeout;
}
```

```
class MyException extends Exception {  
function __construct(private MyExceptionCase $case){  
match($case){  
MyExceptionCase::InvalidMethod => parent::__construct("Bad Request - Invalid Method", 400),  
MyExceptionCase::InvalidProperty => parent::__construct("Bad Request - Invalid Property", 400),  
MyExceptionCase::Timeout => parent::__construct("Bad Request - Timeout", 400)  
};  
}  
}
```

```
// Testing my custom exception class  
try {  
throw new MyException(MyExceptionCase::InvalidMethod);  
} catch (MyException $myE) {  
echo $myE->getMessage(); // Bad Request - Invalid Method  
}
```

[+add a note](#)

- [Типы](#)

- [Введение](#)
- [Система типов](#)
- [NULL](#)
- [Логические значения](#)
- [Целые числа](#)
- [Числа с плавающей точкой](#)
- [Строки](#)
- [Числовые строки](#)
- [Массивы](#)
- [Объекты](#)
- [Перечисления](#)
- [Ресурсы](#)
- [Callable и callback-функции](#)
- [Mixed](#)
- [Void](#)
- [Never](#)
- [Относительные типы классов](#)
- [Типы значений](#)
- [Итерируемые значения](#)
- [Объявления типов](#)
- [Манипуляции с типами](#)

- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

