**标量类型声明：**

默认情况下，如果能做到的话，PHP将会强迫错误类型的**值转为函数期望的标量类型**。 例如，一个函数的一个参数期望是string，但传入的是integer，最终函数得到的将会是一个string类型的值。 可以基于每一个文件开启严格模式。在严格模式中，只有一个与类型声明完全相符的变量才会被接受，否则将会抛出一个TypeError。 唯一的一个例外是可以将integer传给一个期望float的函数。 使用 declare 语句和strict\_types 声明来启用严格模式。

标量类型声明 有两种模式: 强制 (默认) 和 严格模式。可以使用下列类型参数（无论用强制模式还是严格模式）： **字符串(string), 整数 (int), 浮点数 (float), 以及布尔值 (bool)**。

强制：强制类型转换

严格：抛type error

// Coercive mode

function sumOfInts(int ...$ints) {

return array\_sum($ints);

}

var\_dump(sumOfInts(2, '3', 4.1));

**返回值类型声明**

PHP 7 增加了对返回类型声明的支持。 类似于参数类型声明，返回类型声明指明了函数返回值的类型。

function arraysSum(array ...$arrays): array

**通过 define() 定义常量数组**

<?php

define('ANIMALS', [ 'dog', 'cat', 'bird' ]);

echo ANIMALS[1];

// 输出 "cat"

?>

**匿名类**

<?php

interface Logger {

public function log(string $msg);

}

class Application {

private $logger;

public function setLogger(Logger $logger) {

$this->logger = $logger;

}

} $app = new Application;

$app->setLogger(new class implements Logger { public function log(string $msg) { echo $msg; } });

var\_dump($app->getLogger());

?>

**闭包**

<?php

class A {private $x = 1;

}

// PHP 7 之前版本的代码

$getXCB = function() {return $this->x;};

$getX = $getXCB->bindTo(new A, 'A');

// 中间层闭包

echo $getX();

// PHP 7+ 及更高版本的代码

$getX = function() {return $this->x;};

echo $getX->call(new A);

其他：

Null运算符??

整数除法

太空船操作符 < = >