PHPUnit Cheat Sheet

Installation

```
composer require --dev phpunit/phpunit
```

Writing Tests

```php
<?php
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    public function testGreetsWithName(): void
    {
        $greeter = new Greeter();
        $greet = $greeter->greet('Alice');
        $this->assertSame('Hello, Alice!', $greet);
    }
}
```

Expecting exceptions

```php
$this->expectException($exceptionClassName);
$this->expectExceptionCode($code);
$this->expectExceptionMessage($message);
$this->expectExceptionMessageMatches($regex);
```

Attributes PHPUnit 10

Attribute	Description
`#[Test]`	Explicitly mark as test method
`#[DataProvider($method)]`	Use input from data provider
`#[TestWith($data)]`	Provide data directly
`#[TestDox($text)]`	Customize the TestDox output
`#[CoversClass($class)]`	The test should cover this class
`#[UsesClass($class)]`	Uses but does not cover this class
`#[RequiresPhp($version)]`	Skip if PHP version doesn't match
`#[RequiresPhpExtension($e)]`	Skip if PHP extension is missing

See https://docs.phpunit.de/en/13.0/attributes.html for the full list.

Incomplete & Skipped Tests

```php
$this->markTestIncomplete($message);
$this->markTestSkipped($message);
```

Test Doubles

A **stub** provides predefined responses to method calls. A **mock** is configured with expectations about the calls (including counts, order, and arguments) they should receive.

Configuring Test Stubs

```php
$stub = $this->createStub(MyClass::class);
$method = $stub->method('doSomething');

$method->willReturn($value);
$method->willThrowException(new Exception);
$method->willReturnArgument(0);
$method->willReturnCallback('str_rot13');
$method->willReturnSelf();
$method->willReturnMap($map);
```

Configuring Mock Objects

```php
$mock = $this->createMock(MyClass::class);

$mock->expects($this->once())
    ->method('update')
    ->with($this->identicalTo('something'));
```

Mock Invocation Constraints

Constraint	Description
~~`$this->any()`~~	~~Zero or more times~~
`$this->never()`	Never
`$this->atLeastOnce()`	One or more times
`$this->once()`	Exactly once
`$this->atMost($count)`	At most `$count` times
`$this->exactly($count)`	Exactly `$count` times

Running PHPUnit

```
vendor/bin/phpunit [path/to/test]
```

CLI options

Option	Description
`--generate-configuration`	Generate `phpunit.xml` file
`--list-suites`	List available test suites
`--testsuite <name>`	Only run tests from suite(s)
`--exclude-testsuite <ts>`	Exclude tests from test suite(s)
`--list-tests`	List available tests
`--covers <name>`	Only run tests that cover <name>
`--testdox`	Display TestDox output
`--coverage-html <dir>`	Write HTML code coverage report
`--help`	Display the full list of CLI options

Example phpunit.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="vendor/autoload.php"
         cacheDirectory=".phpunit.cache"
         executionOrder="depends,defects"
         beStrictAboutCoverageMetadata="true"
         beStrictAboutOutputDuringTests="true"
         failOnRisky="true"
         failOnWarning="true"
         colors="true">
    <testsuites>
        <testsuite name="default">
            <directory>tests</directory>
        </testsuite>
    </testsuites>

    <source restrictDeprecations="true"
restrictNotices="true" restrictWarnings="true">
        <include>
            <directory>src</directory>
        </include>
    </source>
</phpunit>
```

Identity Assertions

```
assertSame($expected, $actual)
assertArraysAreIdentical($expected, $actual) PHPUnit 13
assertArraysAreIdenticalIgnoringOrder(...) PHPUnit 13
assertArraysHaveIdenticalValues(...) PHPUnit 13
assertArraysHaveIdenticalValuesIgnoringOrder(...) PHPUnit 13
assertArrayIsIdenticalToArrayOnlyConsideringListOfKeys($expected, $actual, $keysToBeConsidered)
assertArrayIsIdenticalToArrayIgnoringListOfKeys($expected, $actual, $keysToBeIgnored)
```

Equality Assertions

```
assertEquals($expected, $actual)
assertEqualsCanonicalizing($expected, $actual)
assertEqualsIgnoringCase($expected, $actual)
assertEqualsWithDelta($expected, $actual, $delta)
assertObjectEquals($expected, $actual, $method)
assertFileEquals($expected, $actual)
assertArraysAreEqual($expected, $actual) PHPUnit 13
assertArraysAreEqualIgnoringOrder(...) PHPUnit 13
assertArraysHaveEqualValues(...) PHPUnit 13
assertArraysHaveEqualValuesIgnoringOrder(...) PHPUnit 13
assertArrayIsEqualToArrayOnlyConsideringListOfKeys($expected, $actual, $keysToBeConsidered)
assertArrayIsEqualToArrayIgnoringListOfKeys($expected, $actual, $keysToBeIgnored)
```

Iterable Assertions

```
assertArrayHasKey($key, $array)
assertContains($needle, $haystack)
assertContainsOnlyX($haystack)* PHPUnit 11.5
assertContainsOnlyInstancesOf($type, $haystack)
```

*: X can be Array, Bool, Callable, Float, Int, Iterable, Null, Numeric, Object, Resource, ClosedResource, Scalar, or String.

Cardinality Assertions

```
assertCount($expectedCount, $haystack)
assertSameSize($expected, $actual)
assertEmpty($actual)
assertGreaterThan($expected, $actual)
assertGreaterThanOrEqual($expected, $actual)
assertLessThan($expected, $actual)
assertLessThanOrEqual($expected, $actual)
```

Object Assertions

```
assertObjectHasProperty($propertyName, $object)
```

Type Assertions

```
assertInstanceOf($expected, $actual)
assertIsX($actual) *
assertNull($actual)
assertTrue($condition)
assertFalse($condition)
```

*: X can be Array, List, Bool, Callable, Float, Int, Iterable, Numeric, Object, Resource, Scalar, or String.

String Assertions

```
assertStringStartsWith($prefix, $string)
assertStringEndsWith($suffix, $string)
assertStringContainsString($needle, $haystack)
assertStringContainsStringIgnoringCase($needle, $haystack)
assertStringEqualsStringIgnoringLineEndings($expected, $actual)
assertMatchesRegularExpression($pattern, $string)
assertStringMatchesFormat($format, $string)
assertStringMatchesFormatFile($formatFile, $string)
assertFileMatchesFormat($format, $actualFile)
assertFileMatchesFormatFile($formatFile, $actualFile)
assertStringEqualsFile($expectedFile, $actualString)
```

JSON Assertions

```
assertJson($actual)
assertJsonFileEqualsJsonFile($expFile, $actualFile)
assertJsonStringEqualsJsonFile($expFile, $actJson)
assertJsonStringEqualsJsonString($expJson, $actJson)
```

XML Assertions

```
assertXmlFileEqualsXmlFile($expectedFile, $actualFile)
assertXmlStringEqualsXmlFile($expectedFile, $actualXml)
assertXmlStringEqualsXmlString($expectedXml, $actualXml)
```

Filesystem Assertions

```
assertDirectoryExists($directory)
assertDirectoryIsReadable($directory)
assertDirectoryIsWritable($directory)
assertFileExists($filename)
assertFileIsReadable($filename)
assertFileIsWritable($filename)
assertIsReadable($filename)
assertIsWritable($filename)
```

Math Assertions

```
assertInfinite($actual)
assertNan($actual)
```

Constraint Assertions

```
assertThat($value, $constraint)
```

Assertion Notes

All assertions accept an optional `$message` argument.

Many assertions have an `assertNot...` variant to assert the opposite, taking the same arguments.