# PHP Namespaces in 120 Seconds


**With <3 from SymfonyCasts**

# Chapter 1: PHP Namespaces in 120 Seconds

Time to master PHP 5.3 namespaces! The good news is, namespaces are easy!

To prove it, we've challenged ourselves to explain them in 120 seconds.

Let's go!

Meet `Foo`. He's a PHP 5.2 class that does a lot of important things:

9 lines | Foo.php

```
... lines 1 - 2
class Foo
{
    public function doAwesomeThings()
    {

    }
}
```

`Foo`, say hi to the listener:

9 lines | Foo.php

```
... lines 1 - 4
    public function doAwesomeThings()
    {
        echo 'Hi listeners';
    }
```

Ok, so `Foo`'s humor is a bit old too.

Using `Foo` is easy - simply `new Foo()`:

6 lines | some-other-file.php

```
... lines 1 - 2
require 'Foo.php';

$foo = new Foo();
```

## Adding a namespace

To keep up with the times, let's put `Foo` in a brand new PHP 5.3 namespace. A namespace is like a directory and by adding `namespace`, `Foo` now lives in `Acme\Tools`:

11 lines | Foo.php

```
... lines 1 - 2
namespace Acme\Tools;

class Foo
{
... lines 7 - 10
}
```

To use `Foo`, we have to call him by his fancy new name:

6 lines | some-other-file.php

```
... lines 1 - 4
$foo = new \Acme\Tools\Foo();
```

This is just like referring to a file by its absolute path.

And that's really it! Adding a namespace to a class is like organizing files from one directory, into a bunch of sub-directories. To refer to a class, use its fully-qualified name, starting with the slash. From here, it's all gravy.

## The use Statement

Since running around with this giant name is a drag, let's add a shortcut:

```
8 lines | some-other-file.php
... lines 1 - 4
use \Acme\Tools\Foo as SomeFooClass;

$foo = new SomeFooClass();
```

The `use` statement lets us call `\Acme\Tools\Foo` class by a nickname. Heck, we can call it anything, or just let it default to `Foo`:

```
8 lines | some-other-file.php
... lines 1 - 4
use \Acme\Tools\Foo;

$foo = new Foo();
```

## The non-namespaced DateTime Class

Great? But what about old-school, non-namespaced PHP classes? For that, let's pick on `DateTime`, a handy class that's core to PHP, and got some new bells and whistles in PHP 5.3. For ever and ever, creating a new `DateTime` object looked the same: `new DateTime()`:

```
10 lines | some-other-file.php
... lines 1 - 8
$dt = new DateTime();
```

And if we're in a normal file, this still works. But in a namespaced file, PHP thinks you're talking about a class *in* the `Acme\Tools` namespace:

```
13 lines | Foo.php
... lines 1 - 6
    public function doAwesomeThings()
    {
        echo 'Hi listeners';

        $dt = new DateTime();
    }
```

You can either refer to the class by its fully-qualified name - `\DateTime`:

```
13 lines | Foo.php
... lines 1 - 10
        $dt = new \DateTime();
... lines 12 - 13
```

or add a `use` statement:

15 lines | Foo.php

```
... lines 1 - 2
namespace Acme\Tools;

use \DateTime;

class Foo
{
    public function doAwesomeThings()
    {
        echo 'Hi listeners';

        $dt = new DateTime();
    }
}
```

Yes, the `use` statement looks silly, but it tells PHP that when you say `DateTime`, you mean the non-namespaced class `DateTime`. Oh, and get rid of the beginning slash with the `use` statement - everything works completely the same with or without these, but you typically don't see them:

15 lines | Foo.php

```
... lines 1 - 4
use DateTime;
... lines 6 - 15
```

Ok bye!