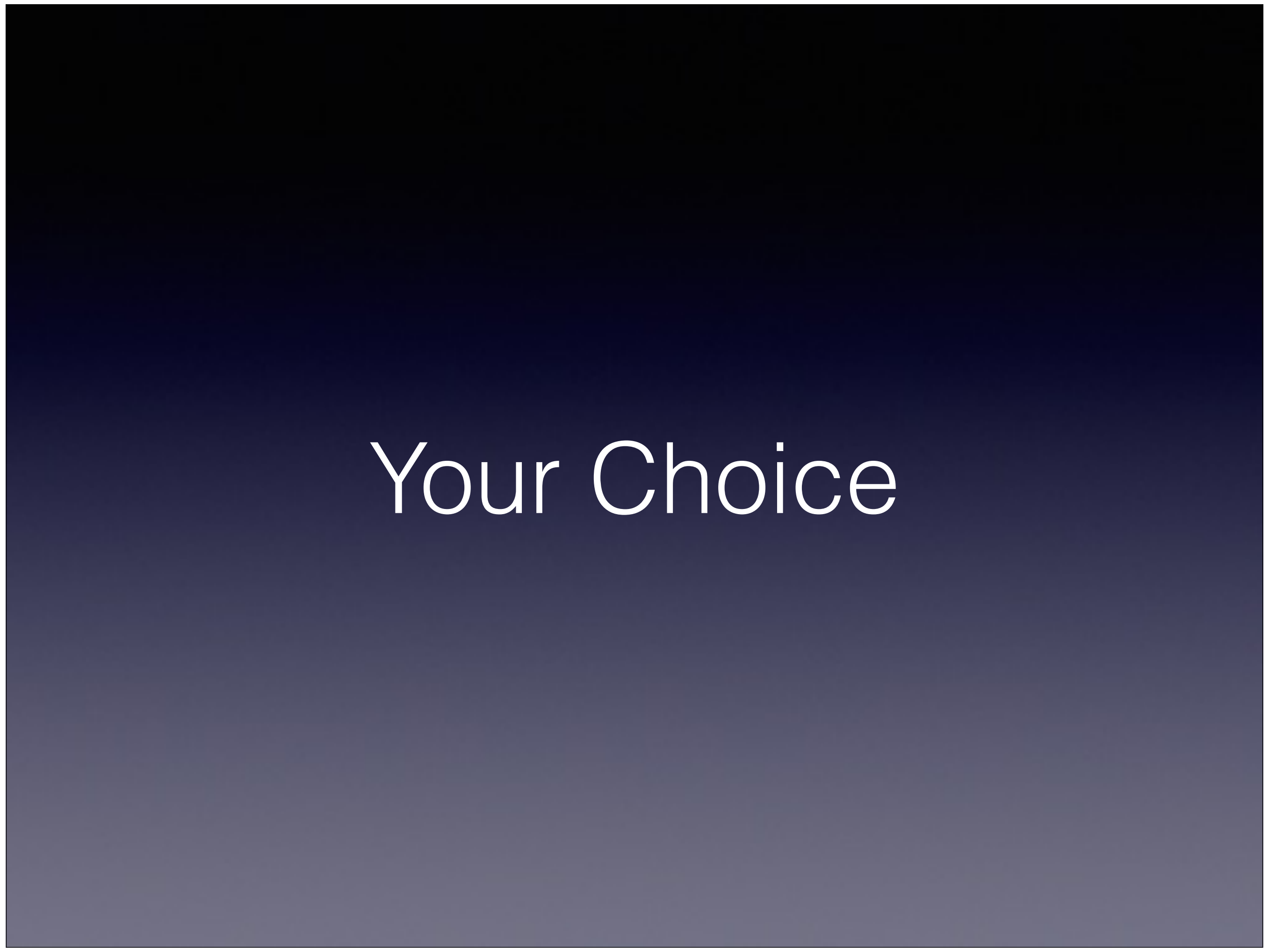
PHP Misses You

Or how I learned to stop hating the language

This is a brief talk about your ex - PHP.

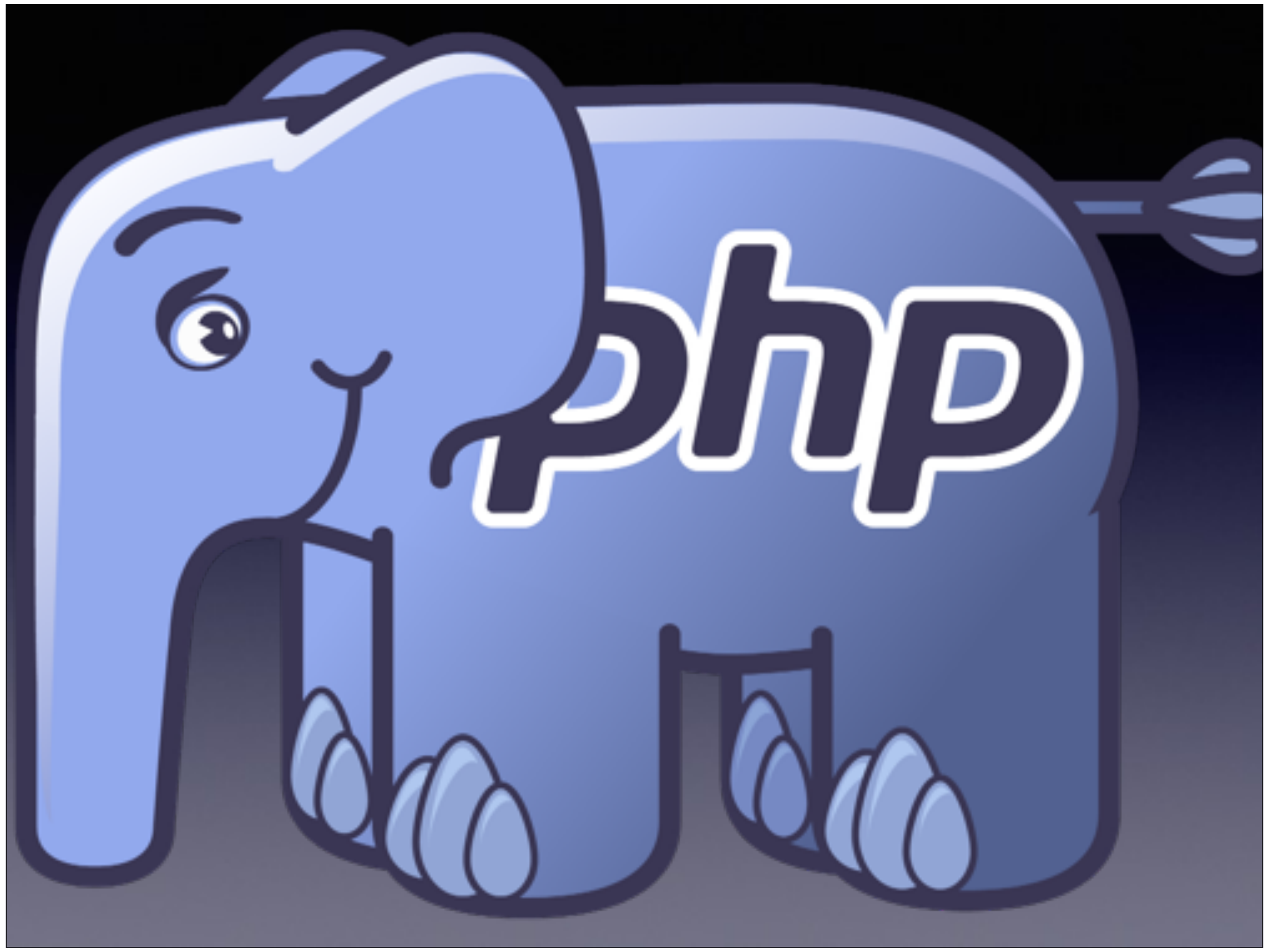


Your Choice

First and foremost - I’m not trying to preach. I’m just trying to inform the public that it’s not that terrible and has a place amongst other web languages.

You’re at the Bar

You’re at the bar with a few friends. Ruby, Python, even your weird friends Scala and Haskell showed up. They’re all here - and in walks PHP.



“Seriously - what is this clown doing here,” you hear someone ask.

You tentatively nod in agreement - but it’s been a while since you two were serious. What even drove you two apart in the first place?

Ternary Statements?

```php
<?php

$test = 1;
echo ($test === 1) ? "one" :
     ($test === 2) ? "two" : "three", "\n";

```

two

empty()

```php
<?php

function test()
{
    return true;
}

var_dump(empty(test()));

```

```
Fatal error: Can't use function return value
in write context in php shell code on line #
```

Naming?

- strcmp()
- strcasecmp()
- is_object()
- isset()
- base64_encode()
- urlencode()

The list goes on…

This list doesn't even start to touch the problems with PHP and naming of functions. Underscores vs no underscores. 'i' in the name to denote case insensitivity, and seemingly at random sometimes a whole new function name will be used to denote the same difference.

Time for a change

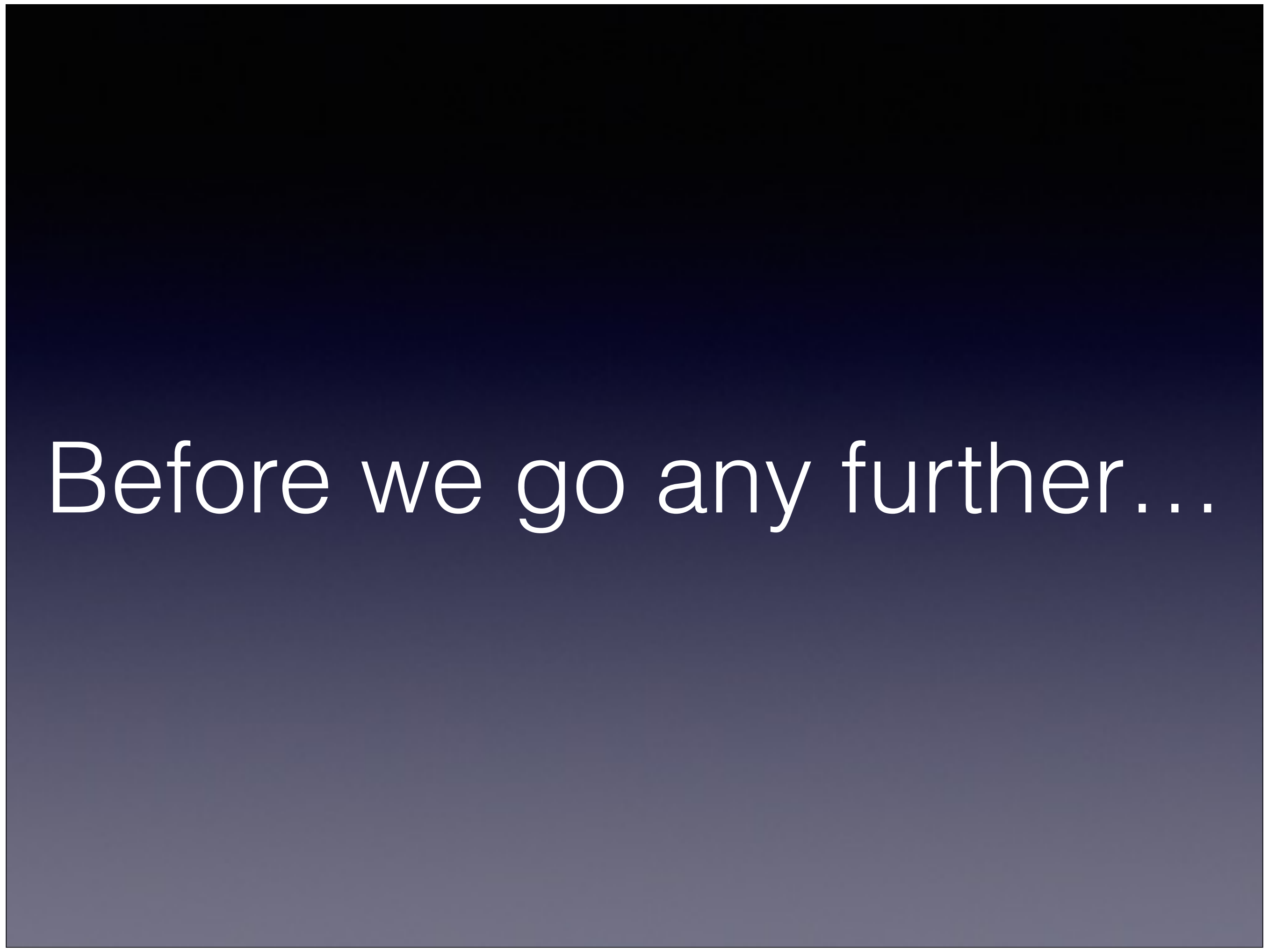
Regardless of why you split - you needed a change. You needed something more sane, something more testable, something less painful.

Who is at fault?

All parties are at fault. PHP has some terrible habits. Some have died, some will never die, and some will be refuted as faults at all. But just like an ex, if you keep talking to PHP and bringing up the past, that’s all you’ll ever have.

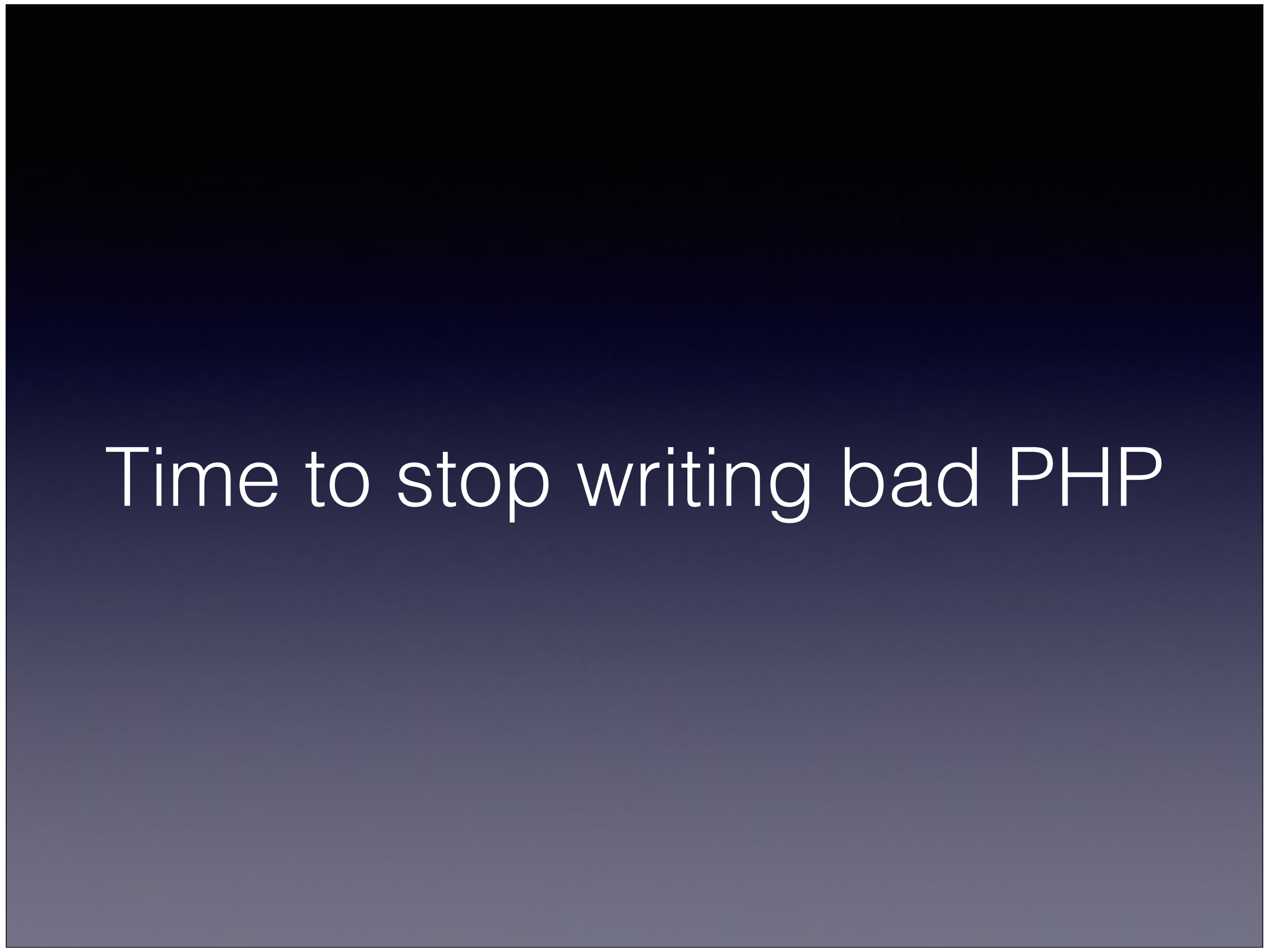
You decide to talk

You've got a new project you're working on and it needs a simple web component. Maybe it's just an API front-end for a database. Maybe you just need some data scraped and returned via json. PHP is all ears.



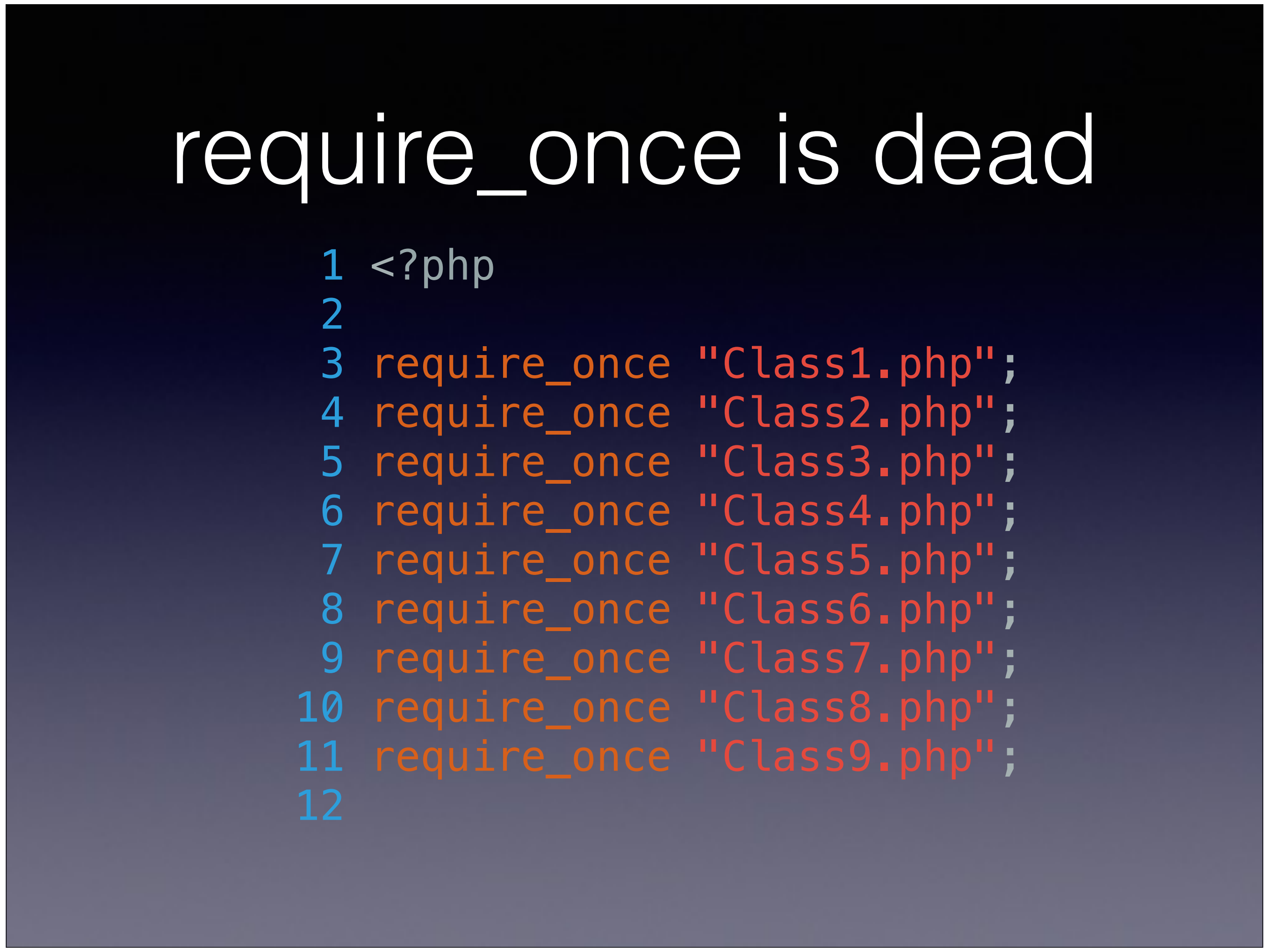
Before we go any further…

You need to check yourself. You don’t want to fall into old habits so quickly. You need to make sure PHP is ready for an adult relationship (open?).



Time to stop writing bad PHP

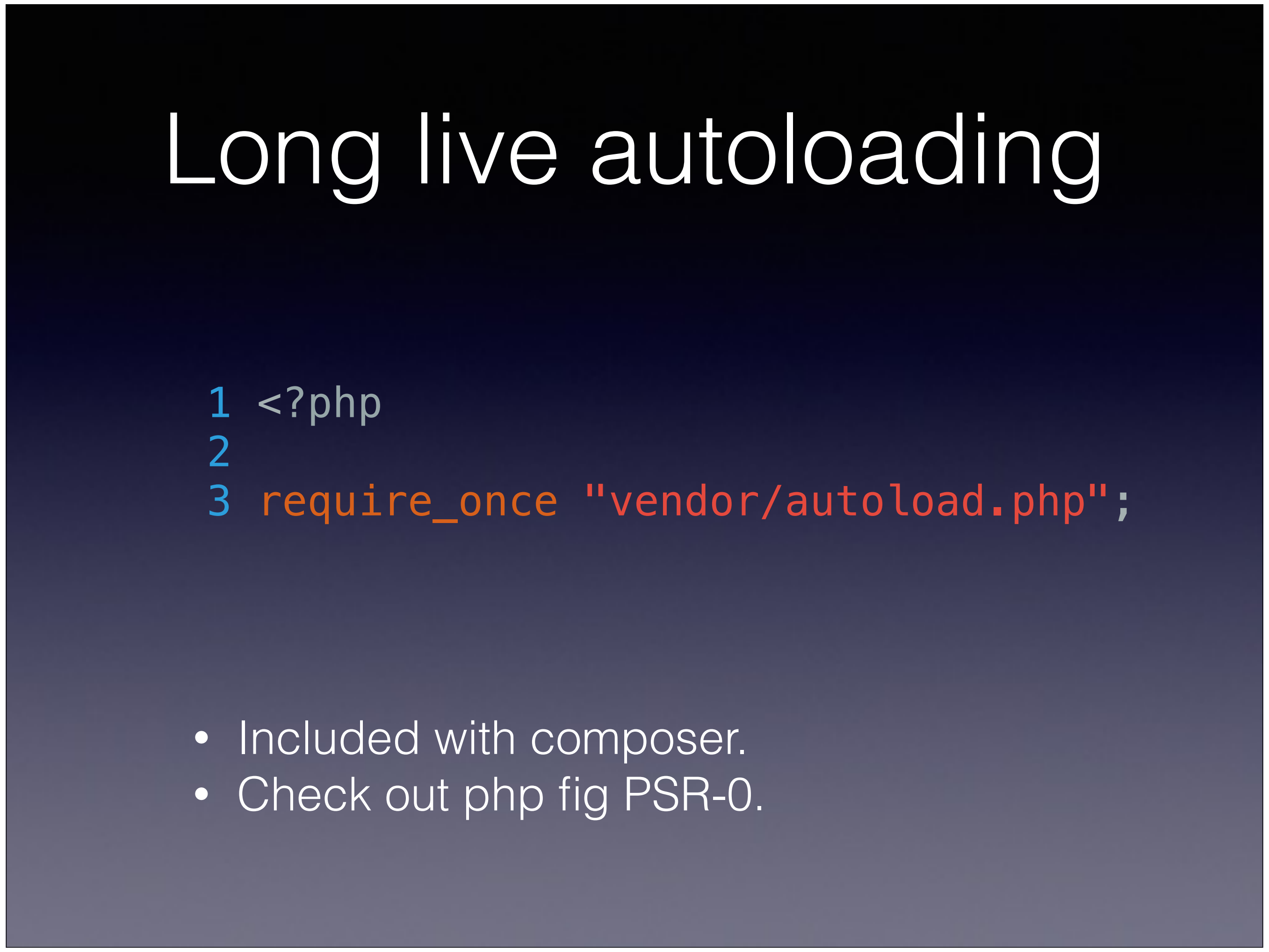
Laughable right? "All PHP is bad PHP…" But like I said, if you only talk to PHP like a child, it'll treat you like a child in return.



require_once is dead

```php
<?php

require_once "Class1.php";
require_once "Class2.php";
require_once "Class3.php";
require_once "Class4.php";
require_once "Class5.php";
require_once "Class6.php";
require_once "Class7.php";
require_once "Class8.php";
require_once "Class9.php";

```

Enough already.

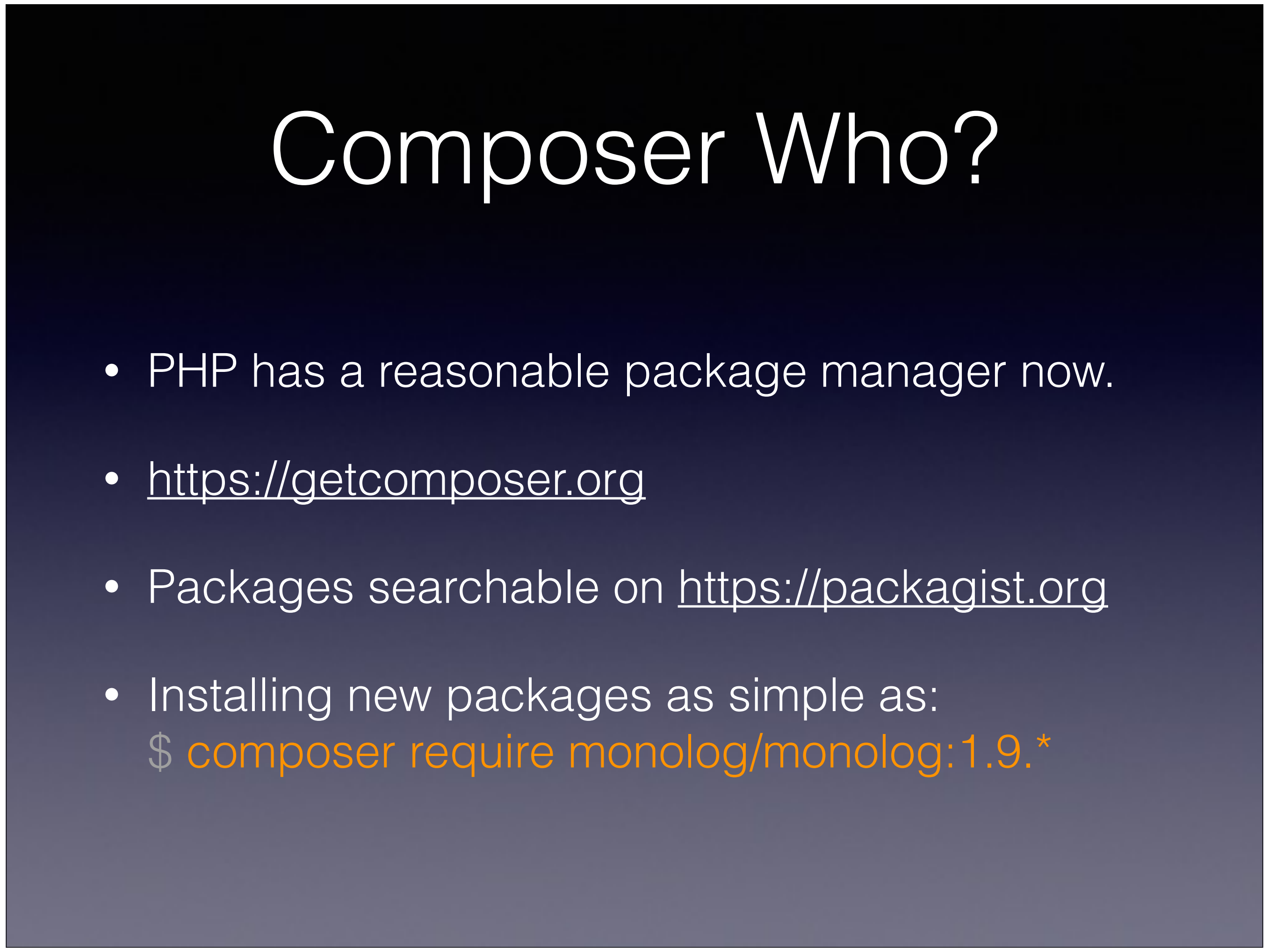


Long live autoloading

```php
<?php

require_once "vendor/autoload.php";
```

- Included with composer.
- Check out php fig PSR-0.

If you install any packages with composer, it'll automatically create and update an autoloader for you. Still uses require_once - but classes will be lazy-loaded as necessary when using PSR-0 style autoloading.

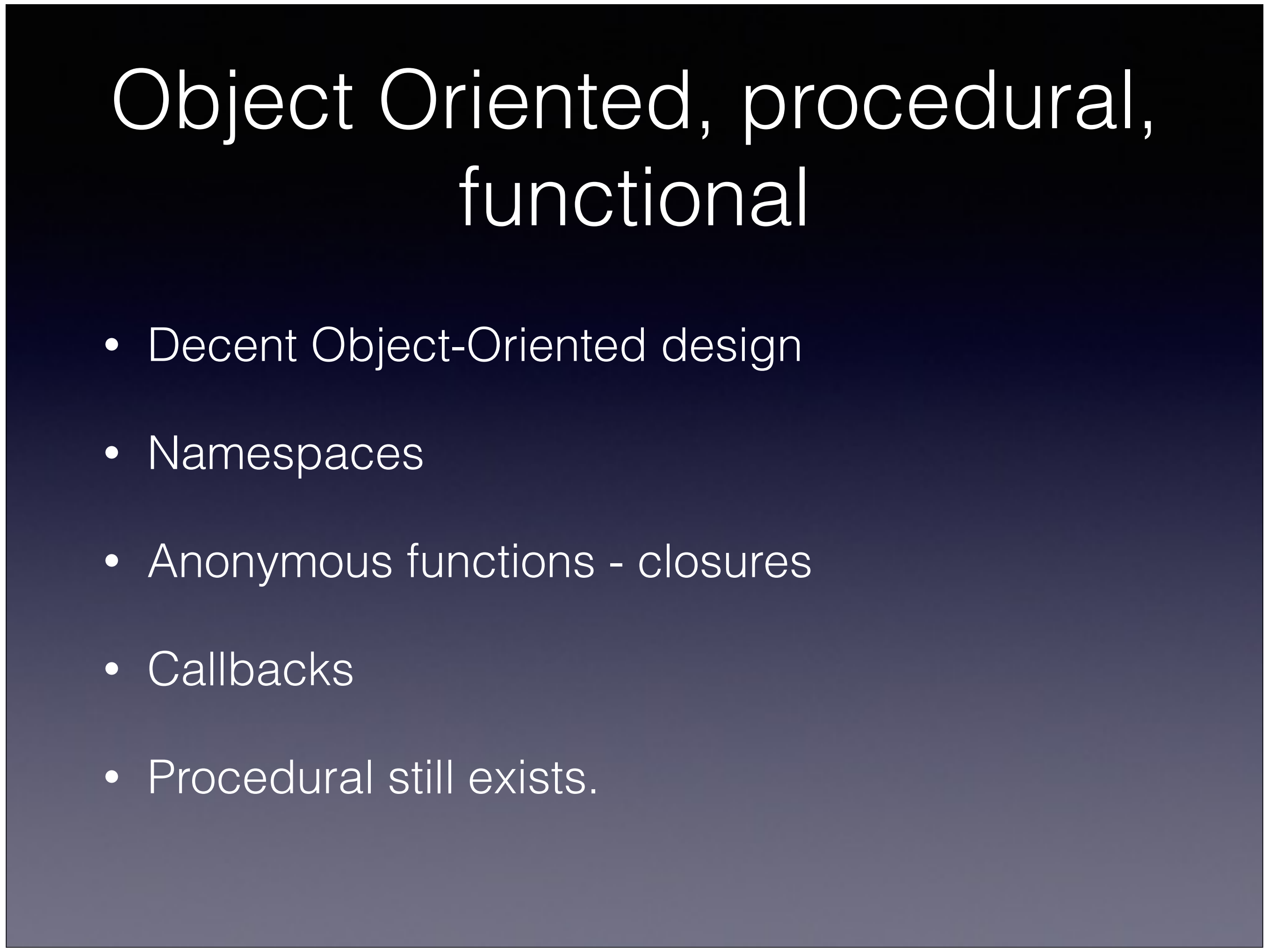

Composer Who?

- PHP has a reasonable package manager now.
- https://getcomposer.org
- Packages searchable on https://packagist.org
- Installing new packages as simple as:
  ```
  $ composer require monolog/monolog:1.9.*
  ```

Composer is a little late to the game - but better late then never. PHP has badly needed a reasonable packaging system, and I think composer does this job well.

Composer supports repositories hosted on github, bitbucket, or private repositories.

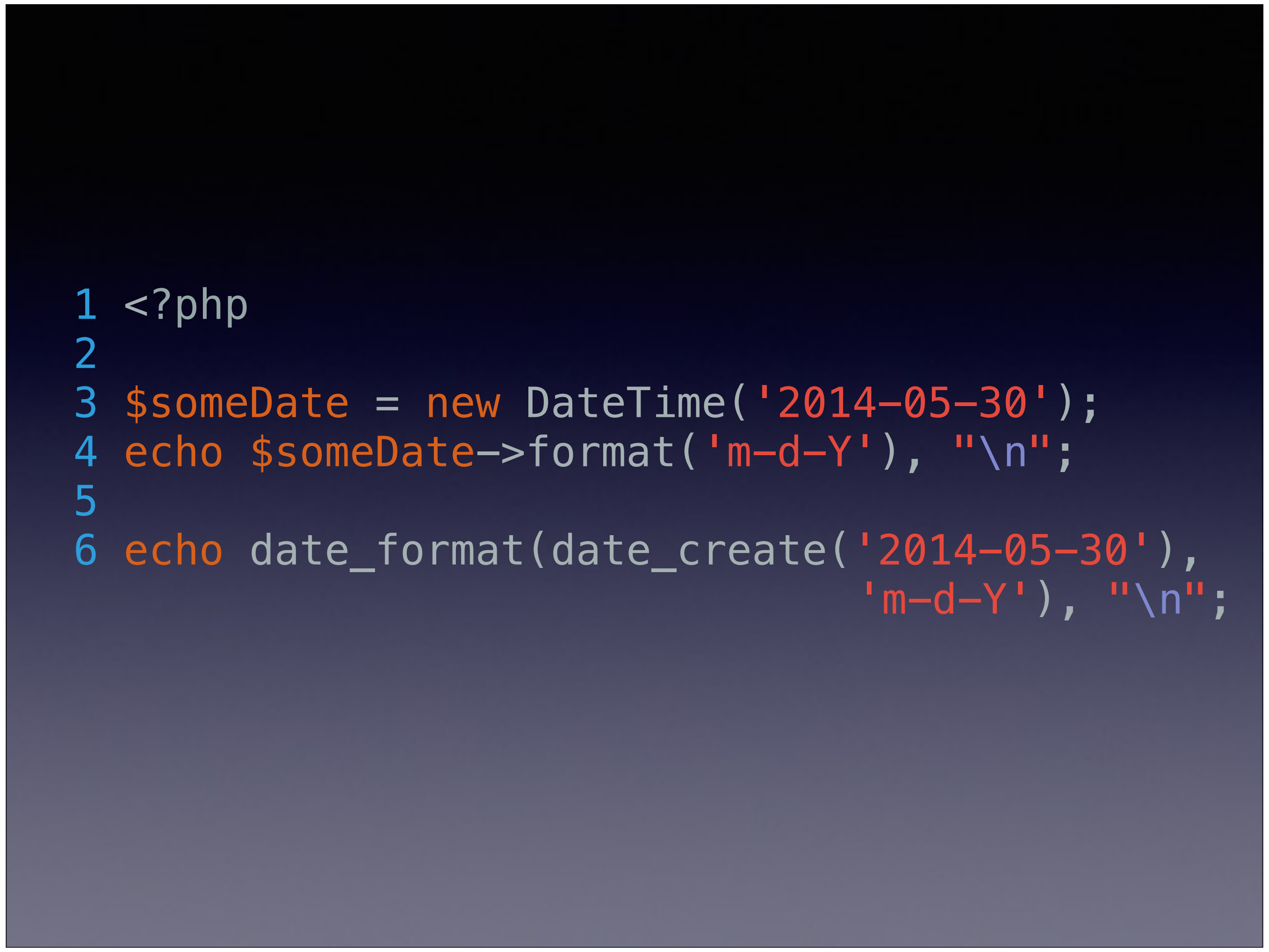
Composer supports "require" versioning via tags or branches and installing from source or compressed "distribution" releases.



Object Oriented, procedural, functional

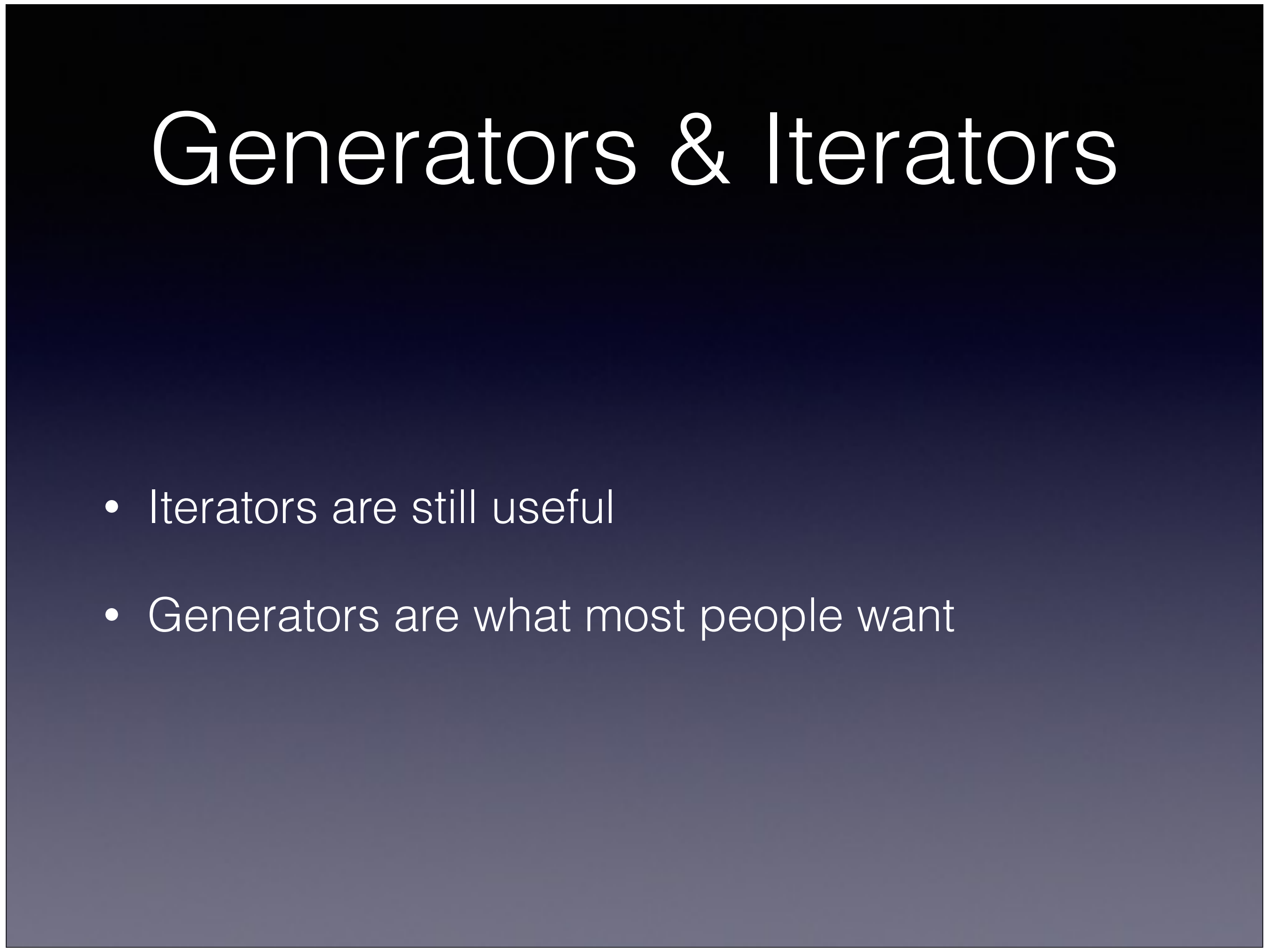
- Decent Object-Oriented design
- Namespaces
- Anonymous functions - closures
- Callbacks
- Procedural still exists.

The object oriented design of php is very java-like. If you're familiar with java it should come easy - PHP even has the 'finally' keyword now. Object-oriented code is being pushed as the standard, with all the PSR's and code-sniffing rules that complain about not being a class or not belonging to a namespace. Nothing is worse than working with PHP 4 style code. Namespacing is good. You can "use" specific classes from namespaces as necessary. No more stepping on toes when using library code/packages.
Programming in a more functional manner is possible thanks to full support for anonymous functions, lambdas in python. PHP has full support for closures in any language construct that requires a callback function now.



```php
<?php

$someDate = new DateTime('2014-05-30');
echo $someDate->format('m-d-Y'), "\n";

echo date_format(date_create('2014-05-30'),
                                        'm-d-Y'), "\n";
```

Two identical lines will be printed out by this code. Many classes in the language still have procedural interfaces to them. You can use them if necessary.

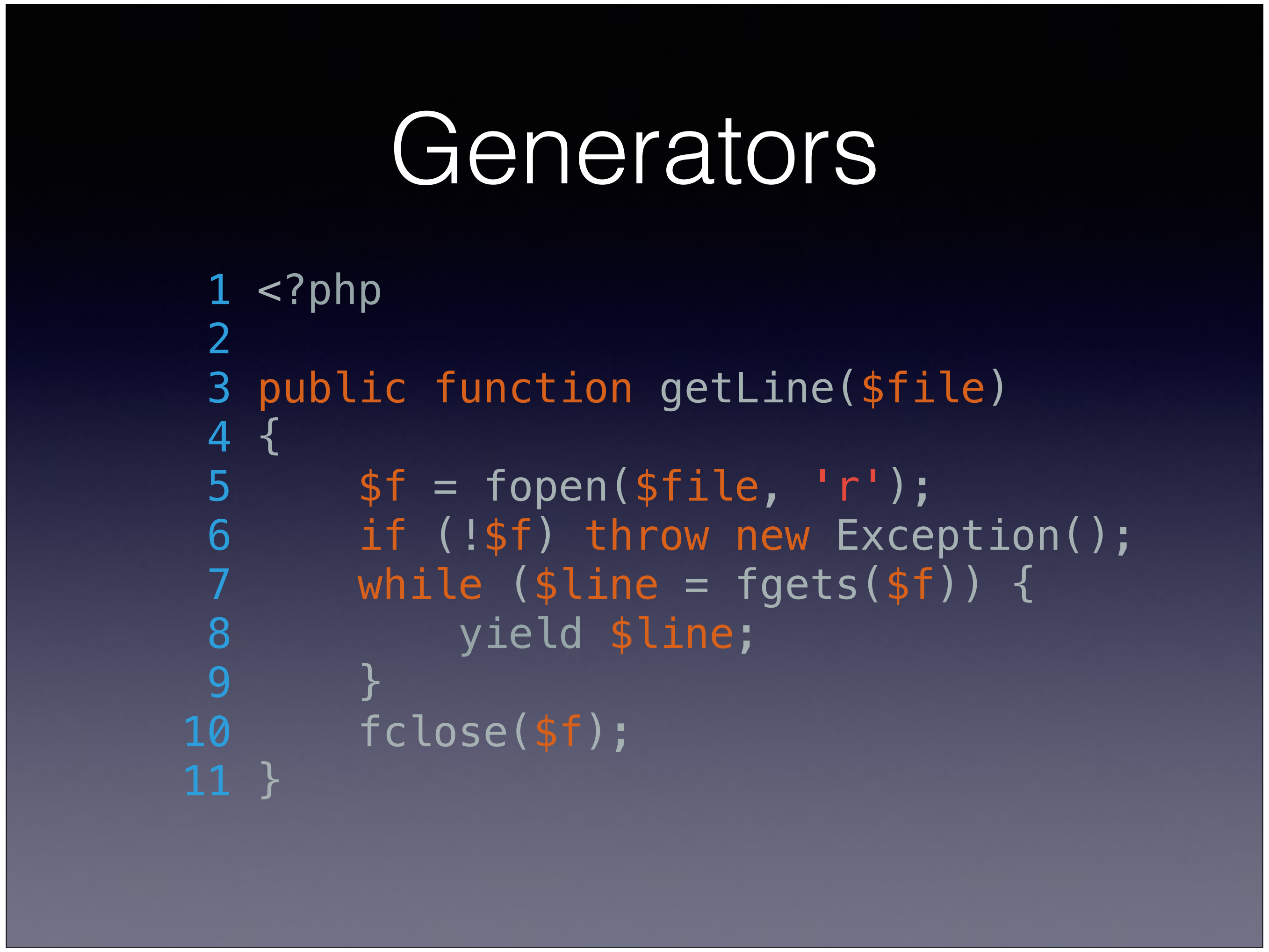


Generators & Iterators

- Iterators are still useful
- Generators are what most people want

Iterators are still a great way to proved a traversable interface to objects or data structures that otherwise would not be able to be traversed in the way that you would want them to be.

Generators take iterators a step further and automatically implement a bunch of the stateful details for you.



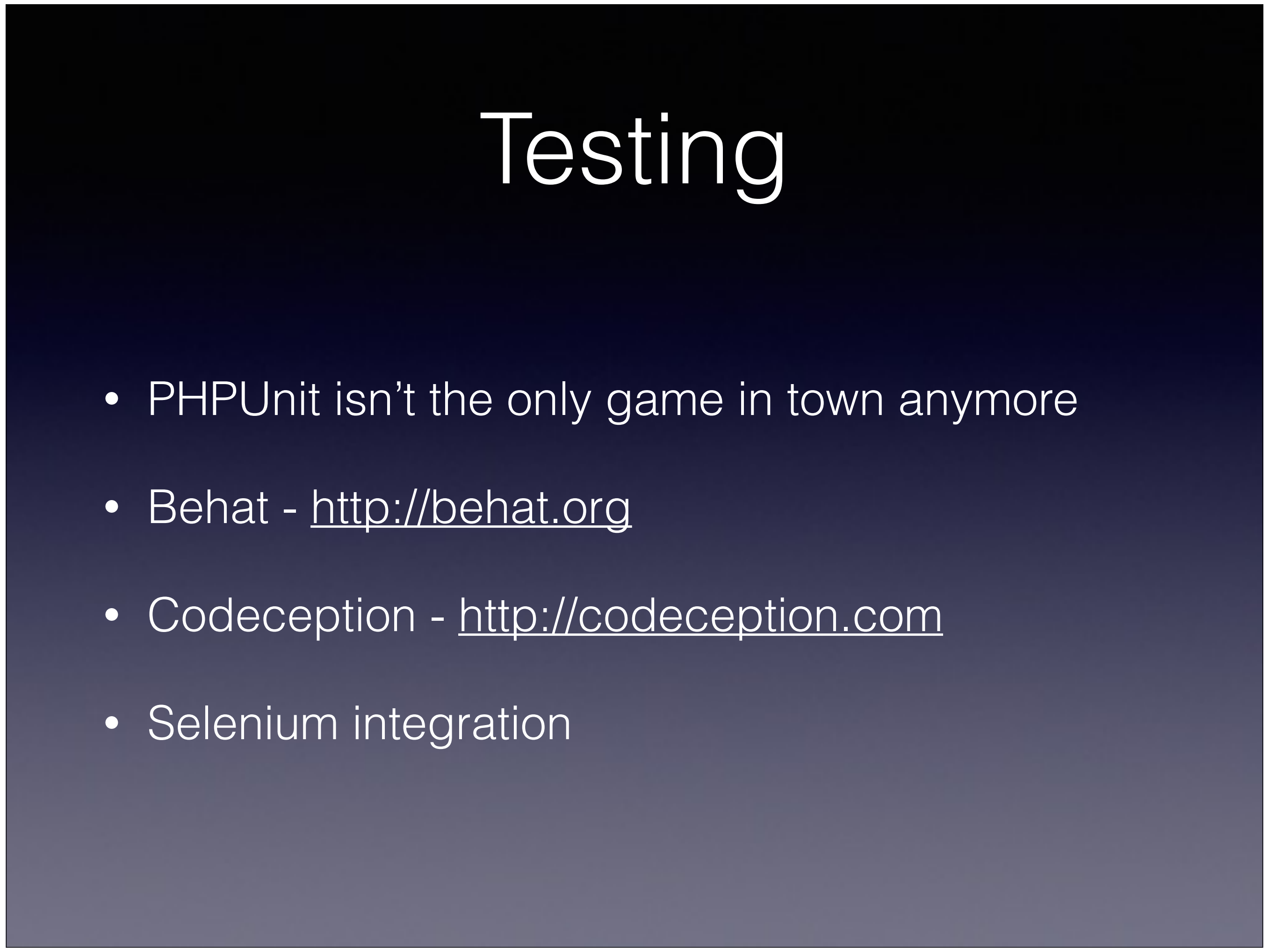
Generators

```php
<?php

public function getLine($file)
{
    $f = fopen($file, 'r');
    if (!$f) throw new Exception();
    while ($line = fgets($f)) {
        yield $line;
    }
    fclose($f);
}
```

The above is a simple generator implementation that yields the current line of the file. Calling next, current, rewind, etc., are all implemented by the generator class internally. Upon calling this function, you'll be given a generator instance and be able to use any of the generators internal methods to traverse. The biggest difference and use case for a generator, since it basically implements an iterator for you, is that it also handles your current iterator state without you having to manage it yourself if you implemented the iterator.

“What about testing?”

–Ruby



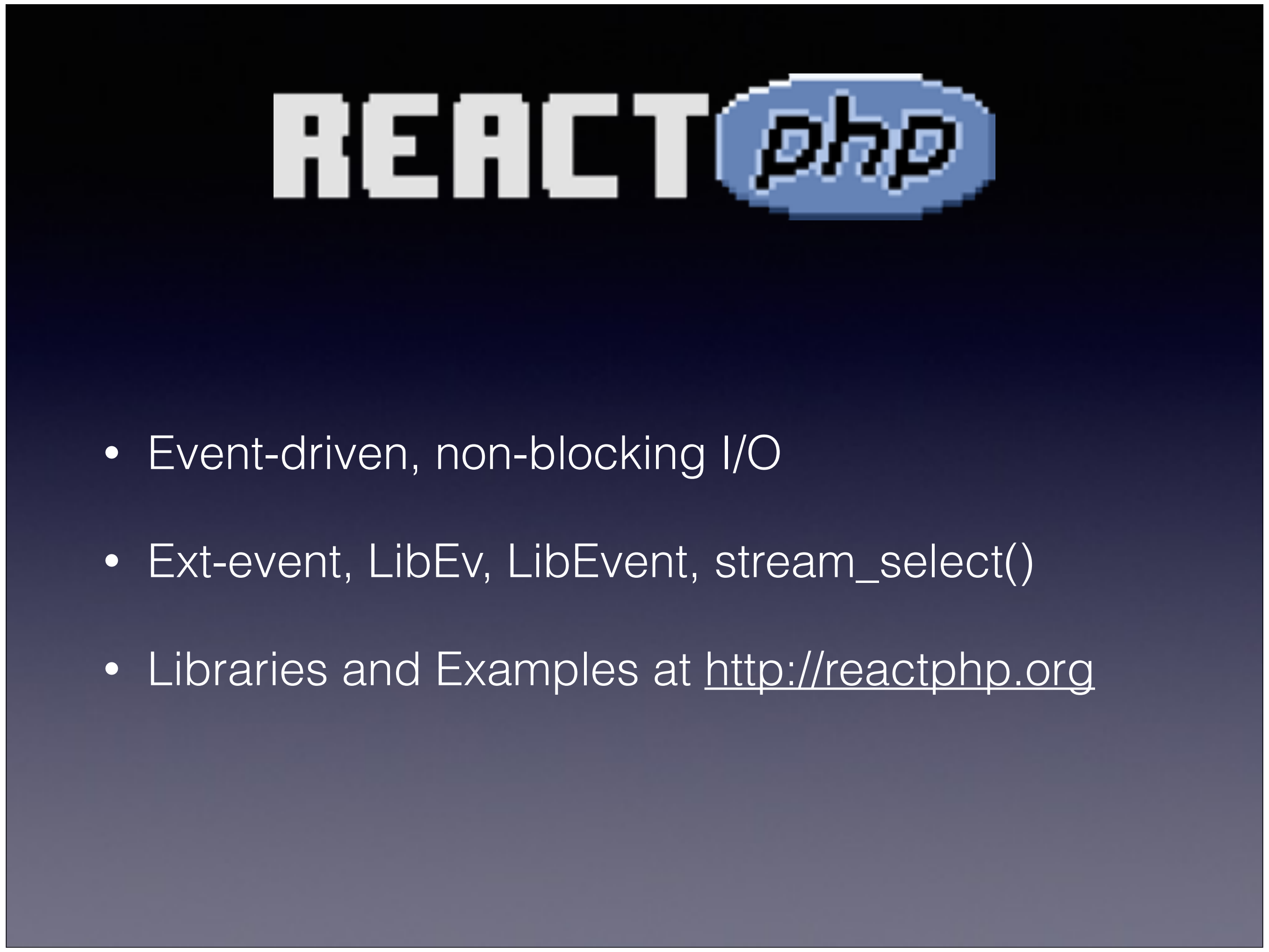
Testing

- PHPUnit isn't the only game in town anymore
- Behat - http://behat.org
- Codeception - http://codeception.com
- Selenium integration

PHPUnit isn't the only game anymore. Behat is a newish behavior driven development testing framework. Codeception is pretty excellent as it provides an easy way to get up and running with Acceptance testing via Selenium, Functional testing, and Unit tests without too much fuss setting it all up.

“Does PHP even event?”

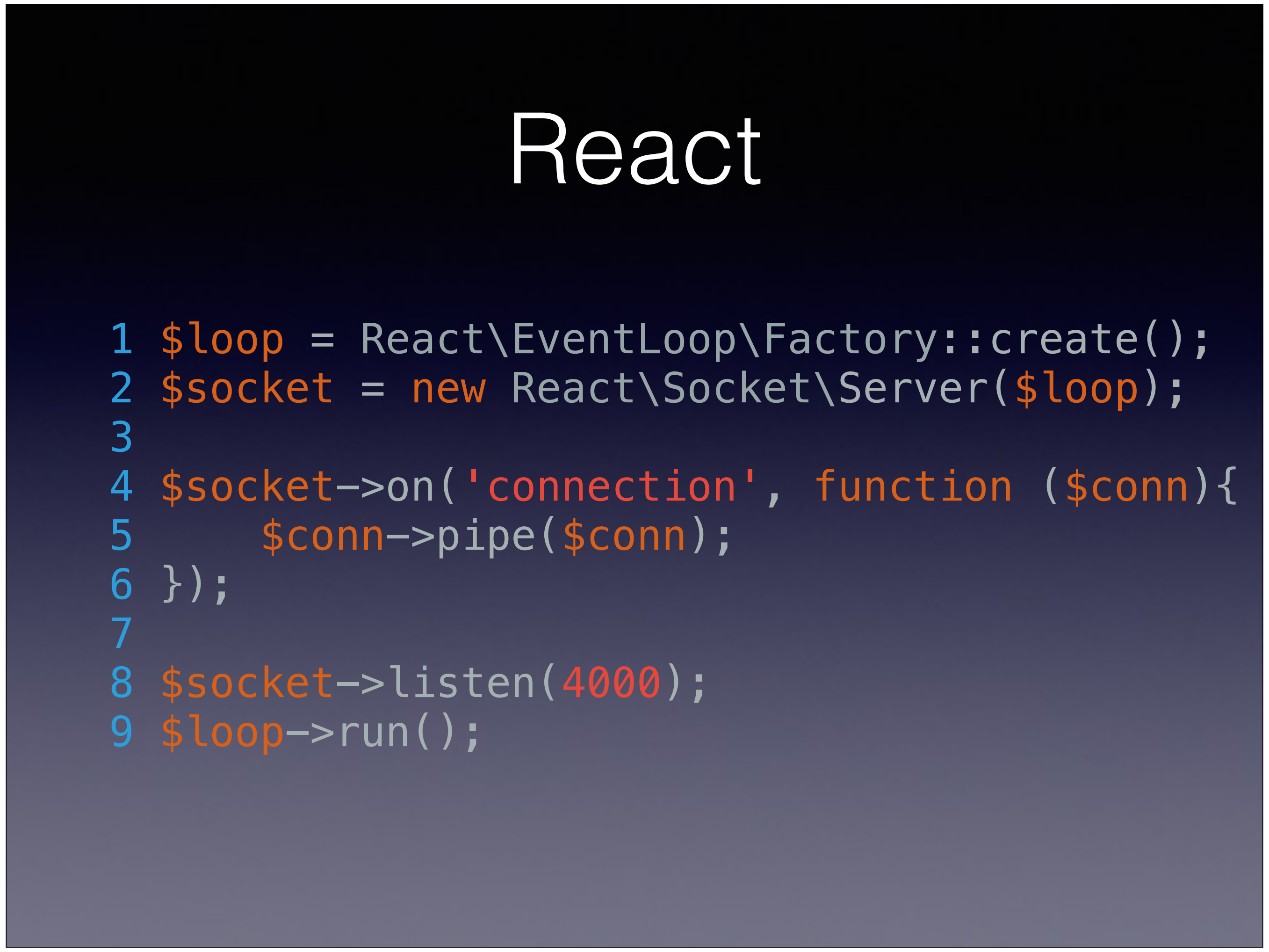
–node.js



REACT php

- Event-driven, non-blocking I/O
- Ext-event, LibEv, LibEvent, stream_select()
- Libraries and Examples at http://reactphp.org

React provides an easy way to write up an event-driven application in a language you already know. It integrates seamlessly with multiple event libraries, and can fall back to using the native PHP stream_select() call if you have to. There are a bunch of useful libraries and example applications, such as an evented non-blocking redis client, at reactphp.org.



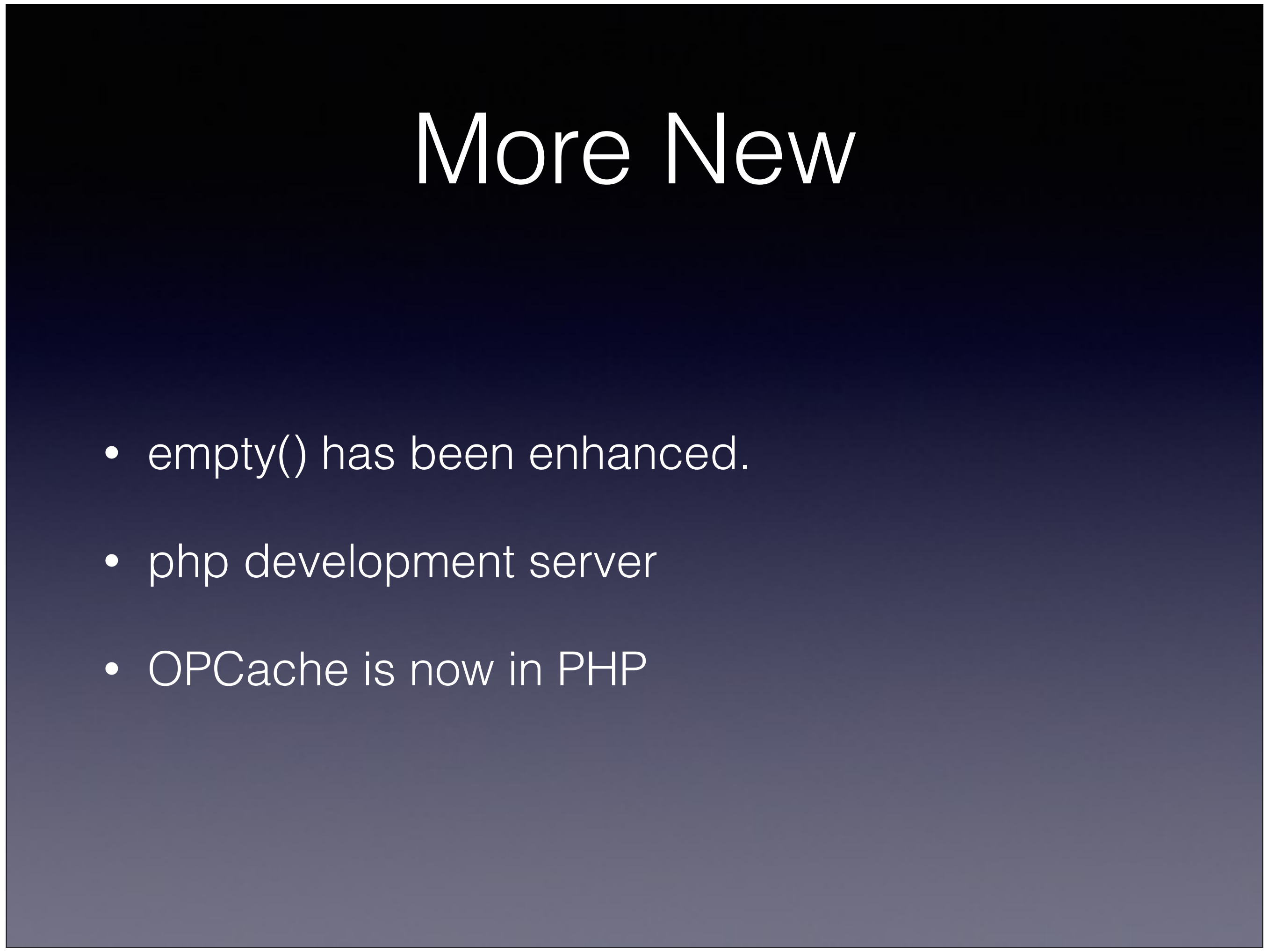
React

```php
$loop = React\EventLoop\Factory::create();
$socket = new React\Socket\Server($loop);

$socket->on('connection', function ($conn){
    $conn->pipe($conn);
});

$socket->listen(4000);
$loop->run();
```

React is pretty simple to use. Create a new loop, create a socket and pass it your loop, specify what to do on an event - in this case our event is "on connection" (with a closure), and tell it to run. This is a 9 line implementation of an echo server.

So what else is new?

(A lot)



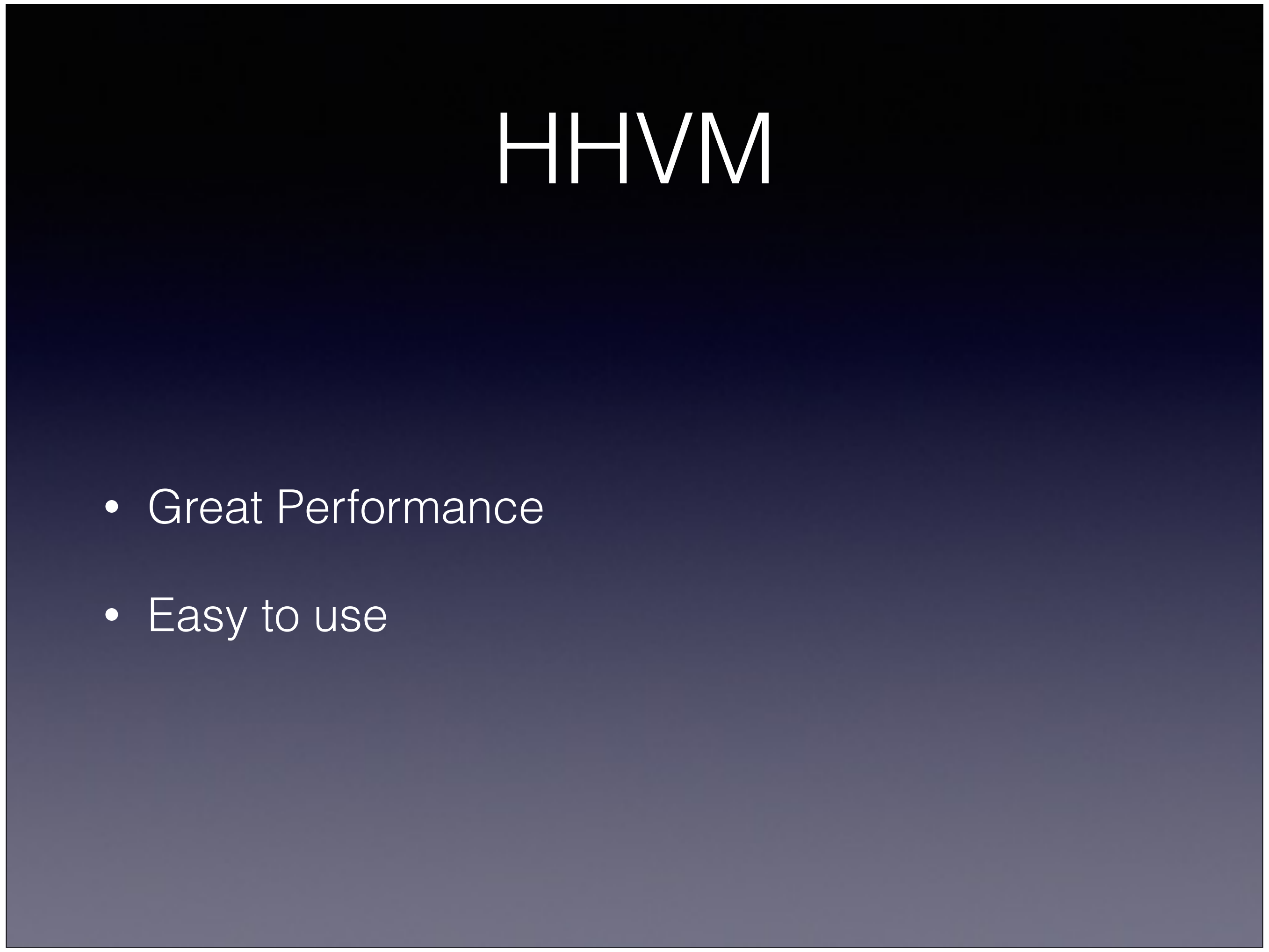
More New

- empty() has been enhanced.
- php development server
- OPCache is now in PHP

Empty can now evaluate function return values rather than just scalar values.

PHP has a development server now - php -S localhost:8888 -t ./

Zend Optimizer+ is now PHP OPCache. This became one of the silly parts of the PHP deployment story eventually. “Why does my wordpress suck?” “Did you enable opcache?” Which one do I choose? There were like 5 major competing opcaches - now we have one. It works pretty well, and there’s APCu for userland caching which the new OPCache actually lacks that most other caching implementations had.



HHVM

- Great Performance
- Easy to use

Why HHVM? Very good performance. At worst, on par with PHP, at best 150-190% runtime speed.

HHVM on Ubuntu 14.04

```
wget -O - http://dl.hhvm.com/conf/hhvm.gpg.key | sudo apt-key add -
echo deb http://dl.hhvm.com/ubuntu trusty main | sudo tee /etc/apt/sources.list.d/hhvm.list
sudo apt-get update
sudo apt-get install hhvm
```

Running HHVM

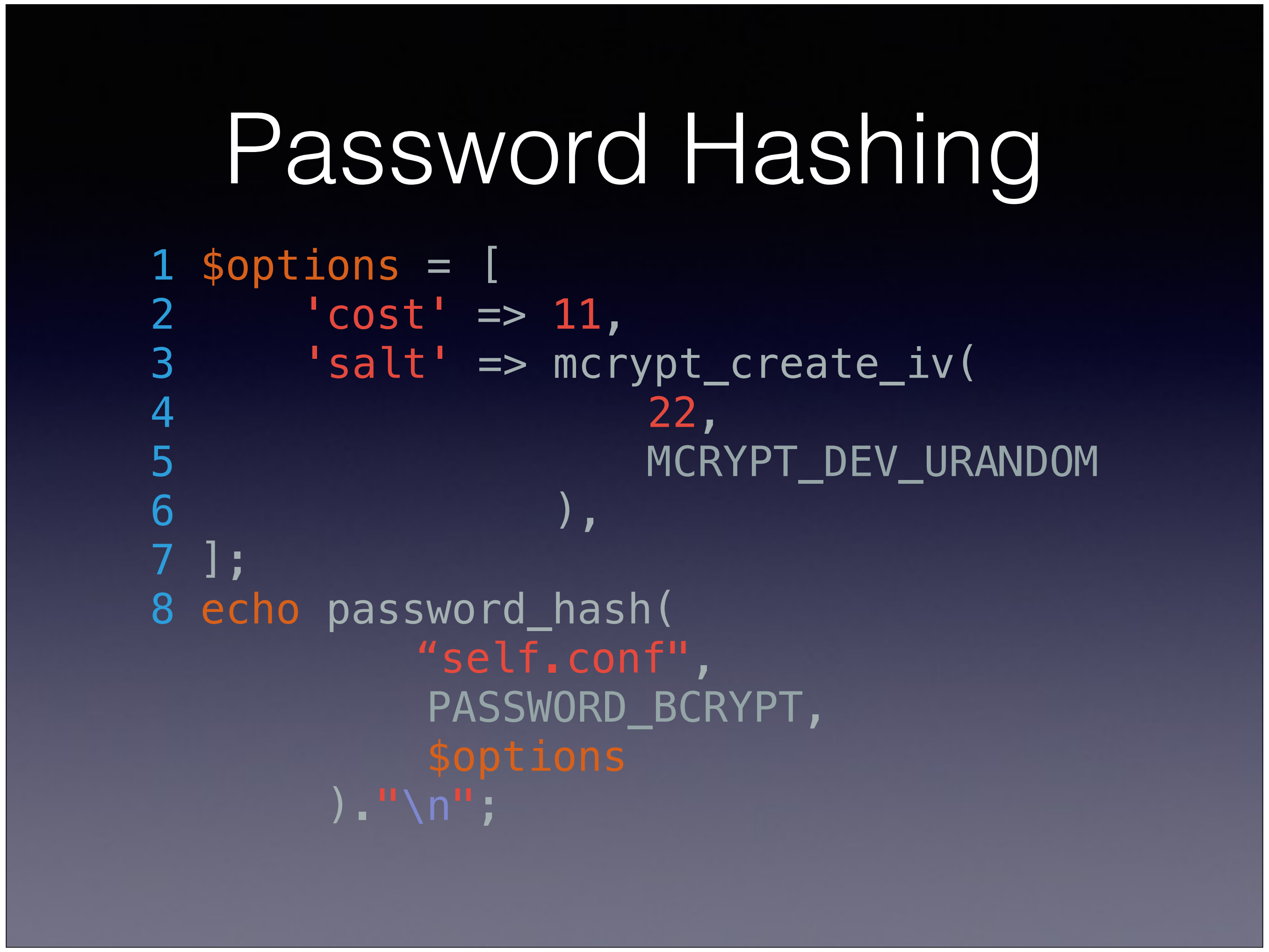
- Run a file:

```
hhvm hello.php
```

- Run a server in current directory:

```
sudo hhvm -m server
```

- Runs in daemon mode as well - see hhvm.com's "Getting Wordpress Running on HHVM"



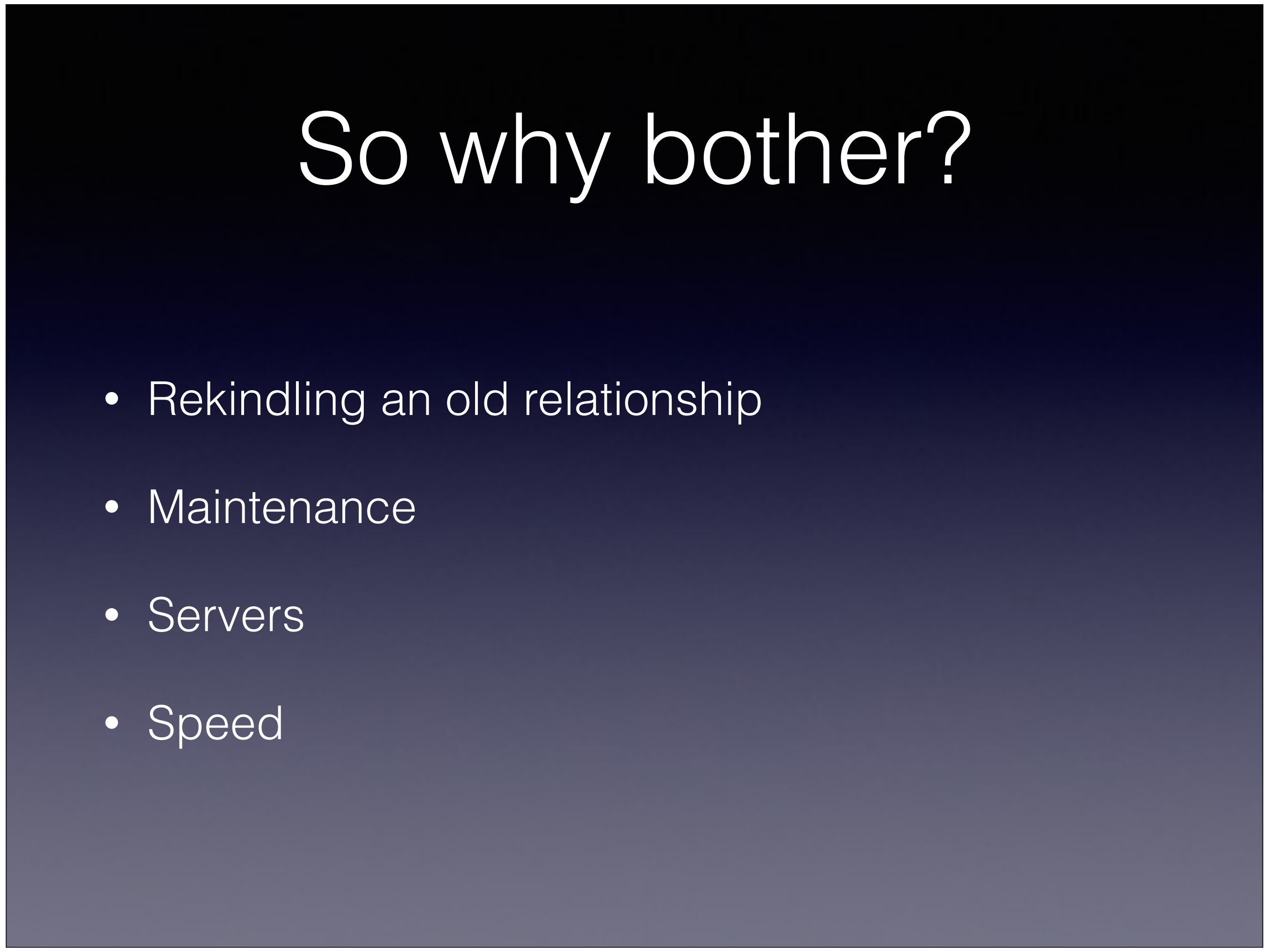
Password Hashing

```
$options = [
    'cost' => 11,
    'salt' => mcrypt_create_iv(
                  22,
                  MCRYPT_DEV_URANDOM
              ),
];
echo password_hash(
         “self.conf",
         PASSWORD_BCRYPT,
         $options
     )."\n";
```

Built-in bcrypt password hashing. Having a central password hashing function that can easily change with the times - ever increasing its difficulty or changing the algorithm as it becomes too weak is a welcome change to the language.

Much more

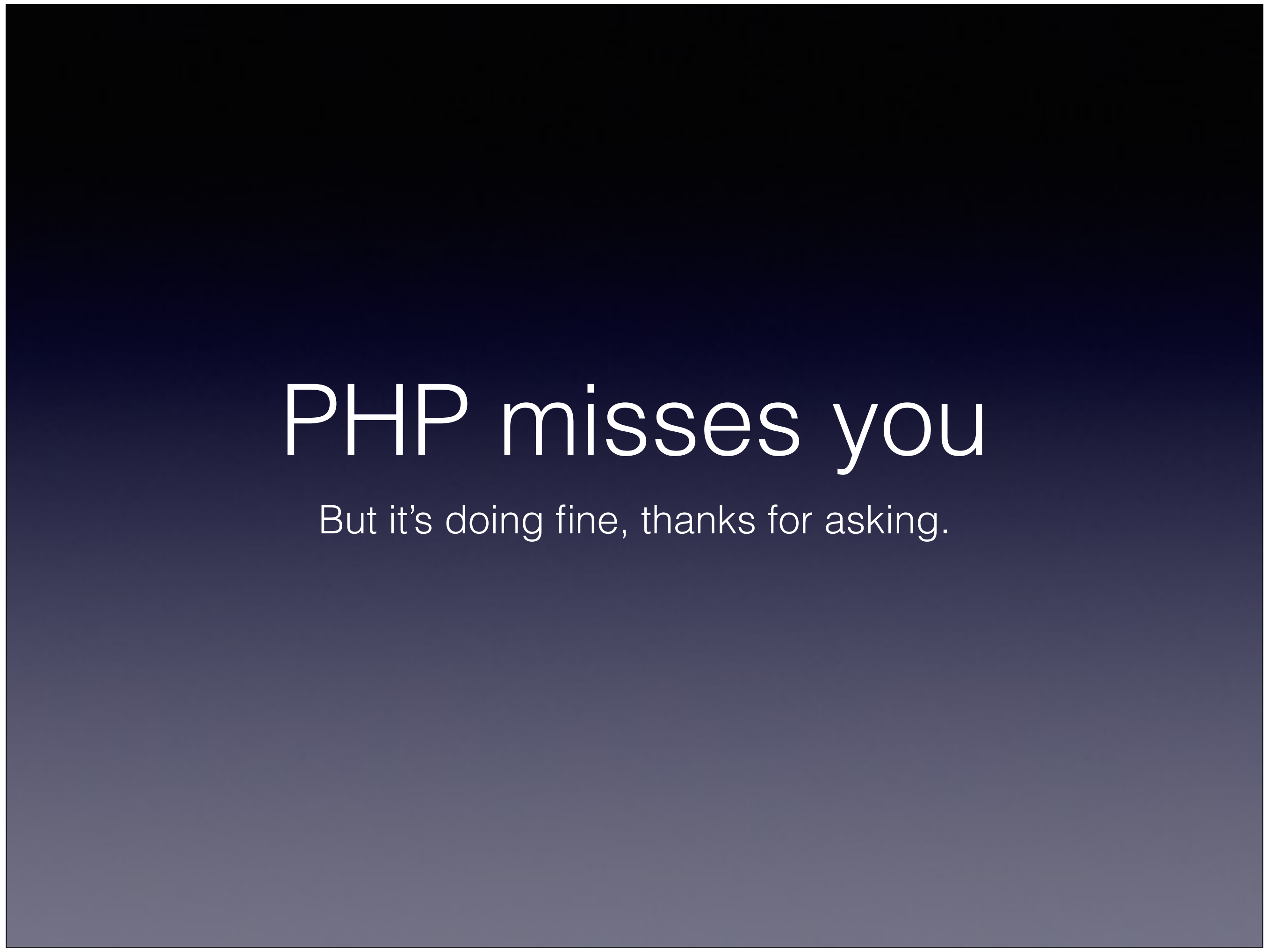
- Array And String literal dereferencing (“Yo!”[1] === “o”)
- static keyword for late static bindings
- foreach supports non-scalar keys
- etc



So why bother?

- Rekindling an old relationship
- Maintenance
- Servers
- Speed

Most of the things people run to other languages for are available in PHP now. Rekindling an old friendship can make for a more comfortable experience. There's a chance you're already running PHP for something, why not use that server for other tasks rather than introducing an unknown in production - or something with a much more ridiculous deployment story for scaling. With Hack becoming available to fix many of PHP's largest problems with type safety and performance, HHNI being available in HHVM to write native "extensions" in PHP and present them to the VM as an "extension", and PHP learning new tricks it's becoming harder to ignore.
PHP still powers over 240MM sites - there's a good chance if you walk into any server you'll run into it, so while you're there chatting, you may as well have an adult conversation and be friends.



PHP misses you

But it's doing fine, thanks for asking.

If time permits:

Frameworks - Yii & Laravel.

Yii - conventions over configuration. Rails-like. Lazy-loading classes. Very speedy.

Laravel - awesome community. Lots of example code and tutorials/guides.

Other options available for writing extensions these days: http://zephir-lang.com - PHP-esque code compiles to C extension. Syntax is a bit different - declared variables and types. Very fast execution.

Thank you!