

What's Happening in PHP?

Eli White

Vice President — One for All Events

@EliW





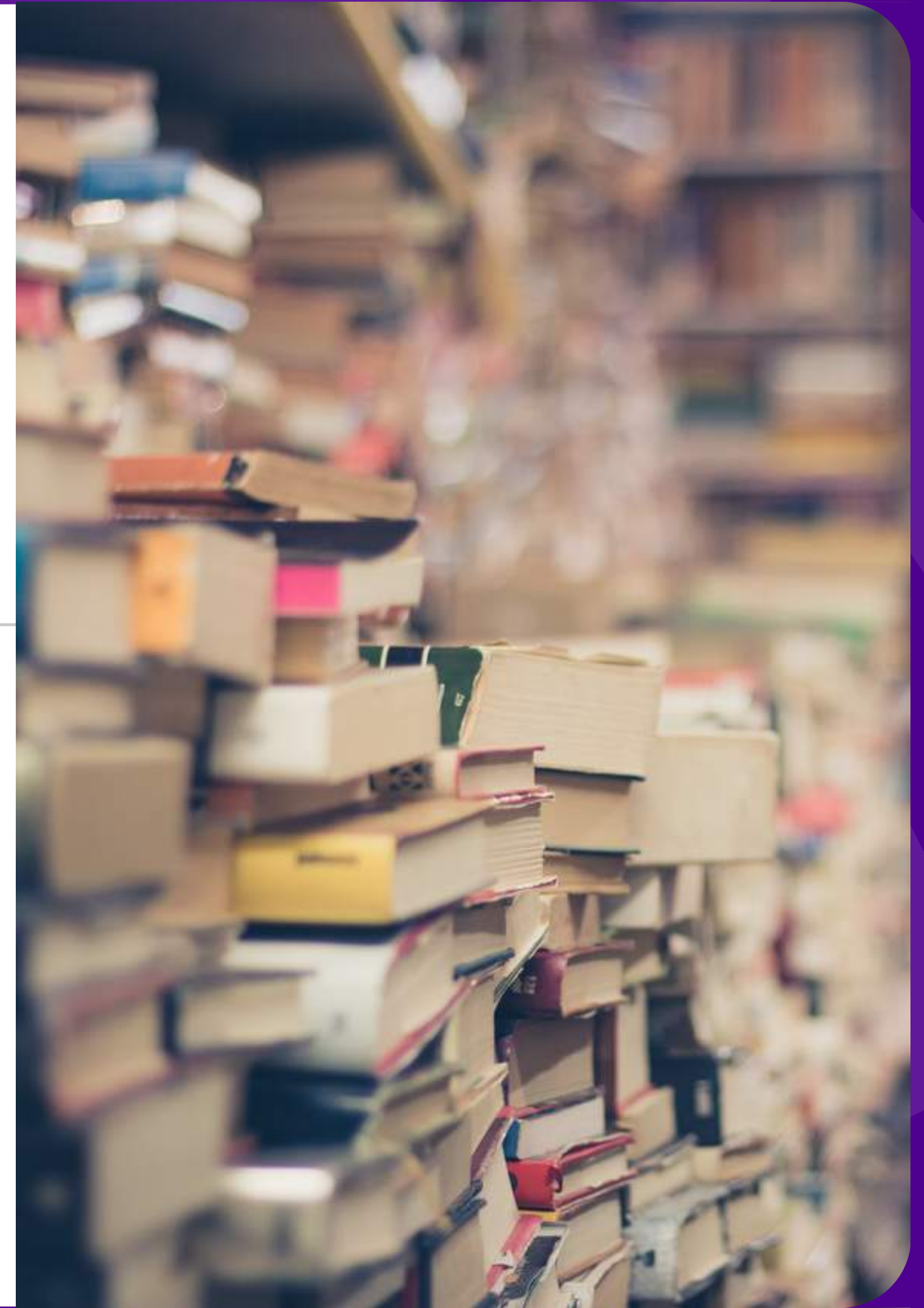
Not your Grandpa's PHP



Sampling of Changes

Namespaces

PHP 5.3 — 2009



What are Namespaces?

What are Namespaces?

```
class DateTime {
    private $dt;

    public __construct($input) {
        $this->dt = $input;
    }

    public format() {
        return date("Y-m-d", $this->dt);
    }
}
```

```
class DateTime {
    public static rfc2822($time) {
        return date(DATE_RFC2822, $time);
    }

    public static rss($time) {
        return date(DATE_ATOM, $time);
    }
}
```



Solve Proliferation of Prefixes

Zend_

wp_

drupal_

Solve Proliferation of Prefixes

sf

DateTime

Defining Namespaces

Namespaces let you have multiple libraries with identically named classes & functions.

```
<?php
namespace WorkLibrary;

class Database {
    public static connect() { /* ... */ }
}
```

Define with the **namespace** keyword to include all code that follows.

```
<?php
namespace MyLibrary;

class Database {
    public static connect() { /* ... */ }
}
```

Sub-namespaces

Use a backslash `\` to create these

```
<?php
namespace MyLibrary\Model;

class Comments {
    public __construct() { /* ... */ }
}
```

```
<?php
namespace Treb\Framework\Utility;

class Cache {
    public __construct() { /* ... */ }
}
```

Using Namespaces

Use the fully qualified name

```
$db = MyProject\Database::connect();  
$model = new MyProject\Model\Comments();
```

Import via the **use** keyword

```
use MyProject\Database;  
$db = Database::connect();
```

Alias when importing via **as** keyword

```
use MyProject\Model\Comments as MyCo;  
$model = new MyCo();
```

Reference builtin classes with top level ****

```
$images = new \DateTime();
```

Closures

PHP 5.3 — 2009





Closure vs Lambda vs Anonymous Function

Creating Anonymous Functions

Declare using
function & store
in variable

Many built-in
functions now
support them

```
$hal = function($what) {  
    echo "Opened the {$what}\n";  
};  
$hal('bay doors'); // Displays: Opened the bay doors
```

```
$turtles = [  
    'Leonardo' => ['color' => 'blue'],  
    'Raphael' => ['color' => 'red'],  
    'Donatello' => ['color' => 'purple'],  
    'Michelangelo' => ['color' => 'orange'],  
];
```

```
$color_sort = function($a, $b) {  
    return strcmp($a['color'], $b['color']);  
};  
uasort($turtles, $color_sort);
```

Creating a Closure

PHP defines the keyword **use** for pulling variables into scope

Variables must be explicitly included

```
$tax = 0.05;
$total = 0;
$cart = ['book' => 9.99, 'toy' => 4.95, 'milk' => 2.50];

$add = function($amount) use ($tax, &$total) {
    $total += $amount + $amount * $tax;
};

foreach ($cart as $price) {
    $add($price);
}

setlocale(LC_MONETARY, 'en_US');
echo money_format('%n', $total), "\n";
// Displays $18.31
```

Traits

PHP 5.4 — 2012



Traits

Enable horizontal
code reuse

Create code and
inject it into
different classes

Contains actual
implementation

```
// Define a simple, albeit silly, trait.
trait Counter
{
    protected $_counter;

    public function increment() {
        ++$this->_counter;
    }

    public function decrement() {
        --$this->_counter;
    }

    public function getCount() {
        return $this->_counter;
    }
}
```

Using Traits

Inject into a class
with the **use**
keyword

A class can include
multiple **traits**

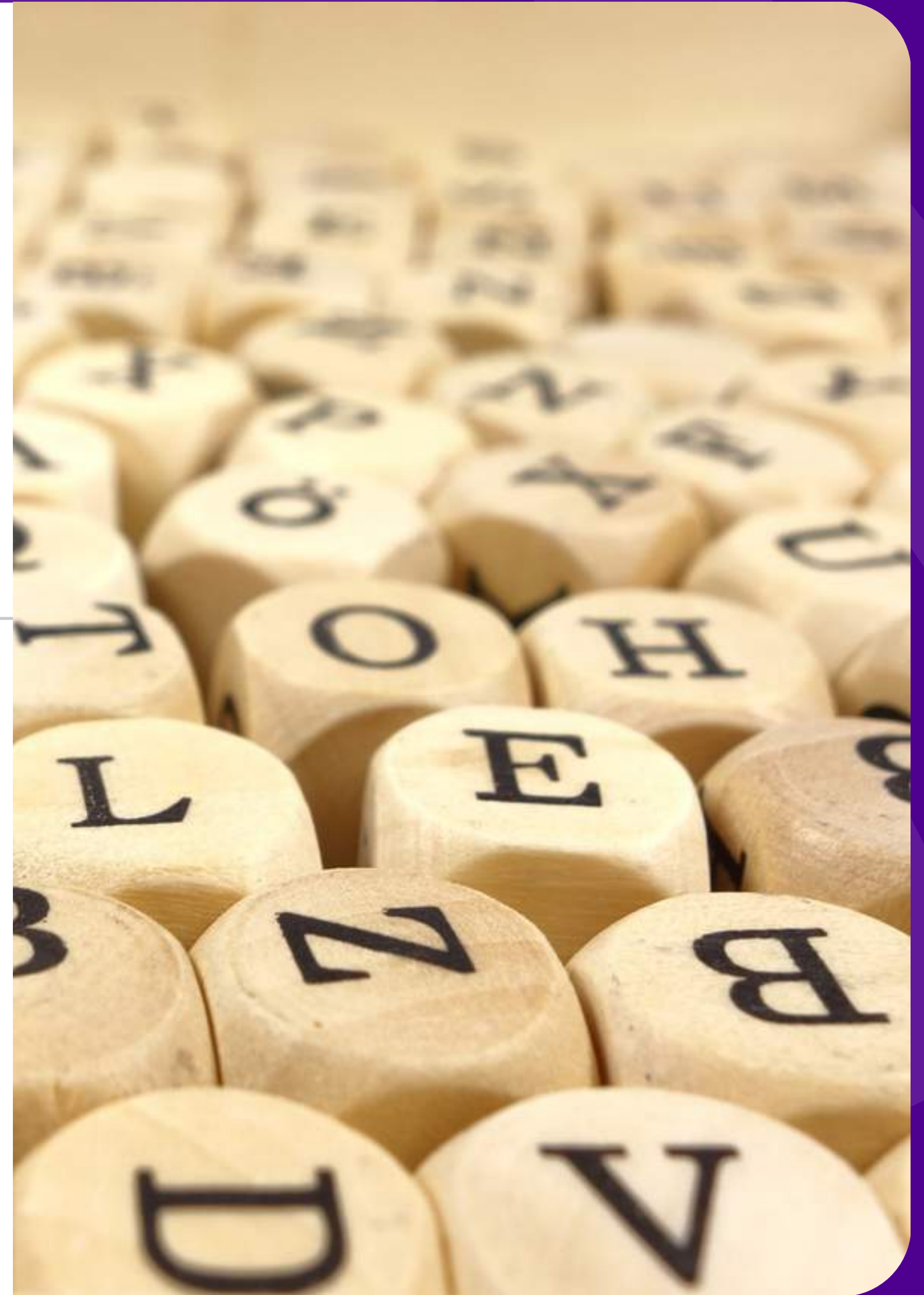
```
class MyCounter
{
    use Counter;

    /* ... */
}

$counter = new MyCounter();
$counter->increment();
echo $counter->getCount(); // 1
```

Strict Typing

PHP 7.0 — 2015



So what's the big deal?



Object & Array Hints

Declare object names
on parameters

Also could
require an array

PHP 5:
Fatal error

PHP 7:
TypeError exception

```
function access(DateTime $time, PDO $db, User $user) {  
    $stmt = $db->prepare(  
        "UPDATE login  
        SET lastAccess = ?  
        WHERE uid = ?");  
    $stmt->execute([  
        $time->format("Y-m-d H:i:s"), $user->uid]);  
}
```

```
function array_mult(array $in) {  
    $result = 1;  
    foreach ($in as $value) {  
        $result *= $value;  
    }  
    return $result;  
}
```

Scalar Hints

PHP 7 Added:
bool
float
int
string

But how should PHP
handle a mismatch
given duck-typing?

```
function greeting(string $name, bool $boss = false) {  
    if ($boss) {  
        return "Good to see you {$name}\n";  
    } else {  
        return "Hey there {$name}\n";  
    }  
}  
echo greeting("Milhouse");
```

```
function sum(int $x, int $y) {  
    return $x + $y;  
}  
  
echo sum(3, 4);  
echo sum(2.5, 4.5); // Returns, what?
```

Best of Both Worlds

Default result is forced conversion via duck-typing

```
<?php
function sum(int $x, int $y) {
    return $x + $y;
}

echo sum(2.5, 4.5); // Returns 6
```

Strict typing can be turned on per file, causing exceptions

```
<?php
declare(strict_types=1);

function sum(int $x, int $y) {
    return $x + $y;
}

echo sum(2.5, "4"); // Throws TypeError exception
```

Allows Programmer Decision

Since it operates on a per file basis.
You are always in charge of whether
you want strict or loose typing.

Decisions made by library
authors don't effect you.



Return Types Too

You can declare
return types as well

```
function sum(int $x, int $y): int {  
    return $x + $y;  
}
```

```
function go(): array {  
    return [ 'Instinct' => 'yellow',  
            'Mystic' => 'blue',  
            'Valor' => 'red' ];  
}
```

PHP 7.1 added
nullable types (?)

```
function connect(string $dsn): ?PDO {  
    try {  
        $db = new PDO($dsn);  
        return $db;  
    } catch (PDOException $e) {  
        return null;  
    }  
}
```

Zend Engine 3

PHP 7.0 — 2015

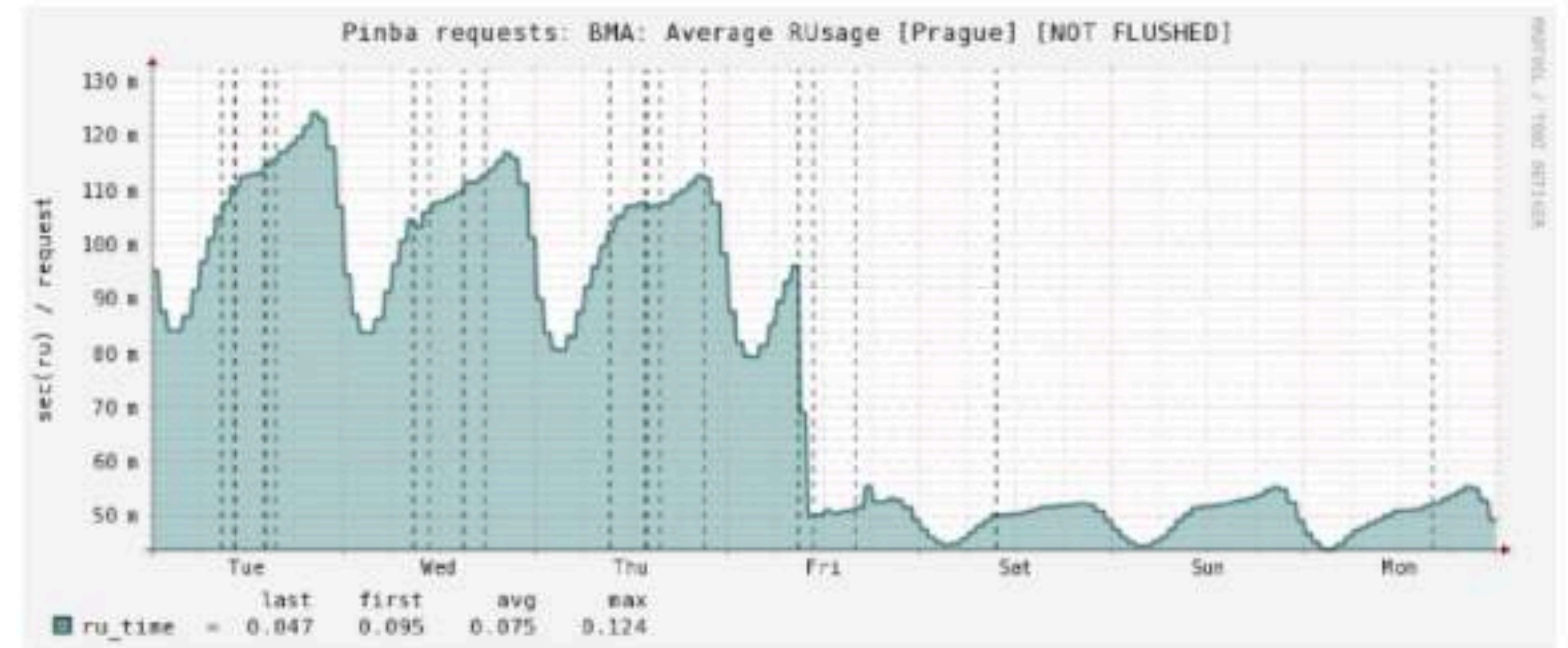




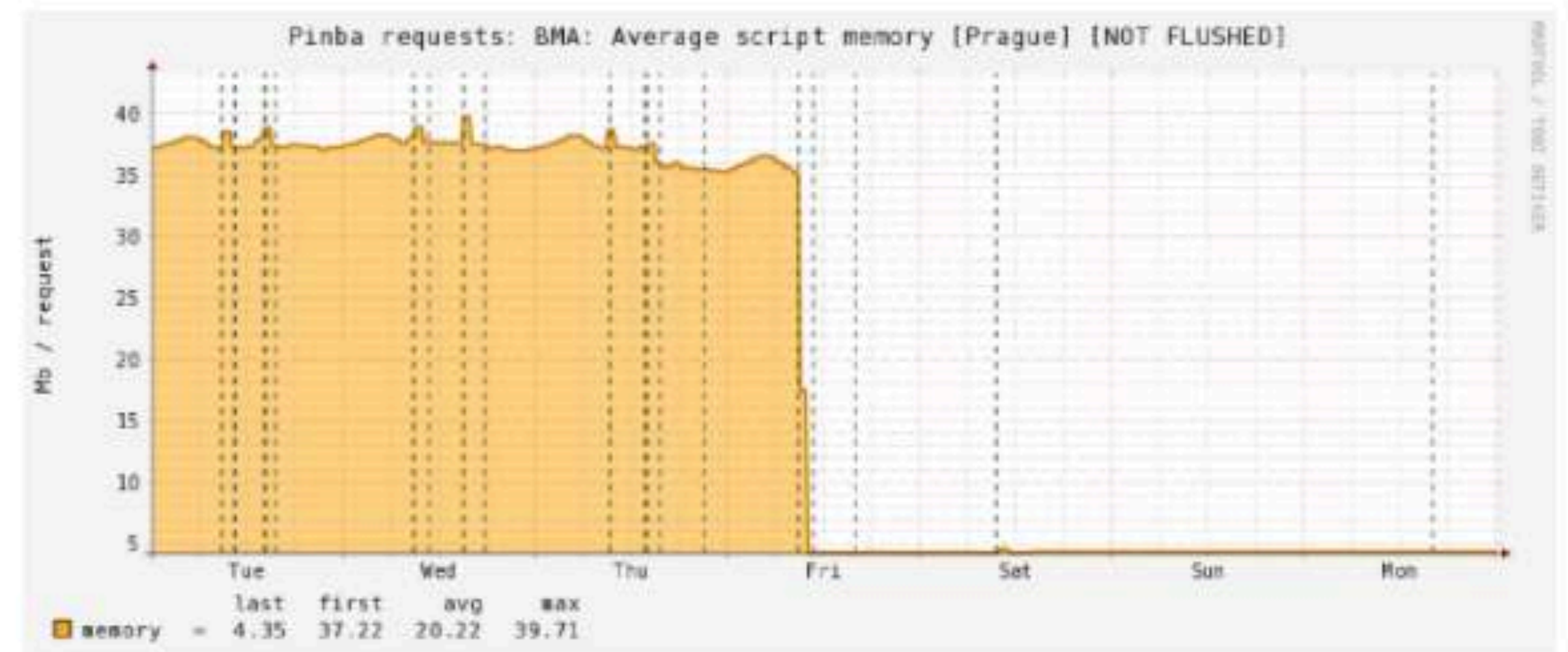
Speed & Memory Usage

Stats from Badoo upgrading to PHP 7

RUsage (CPU time):



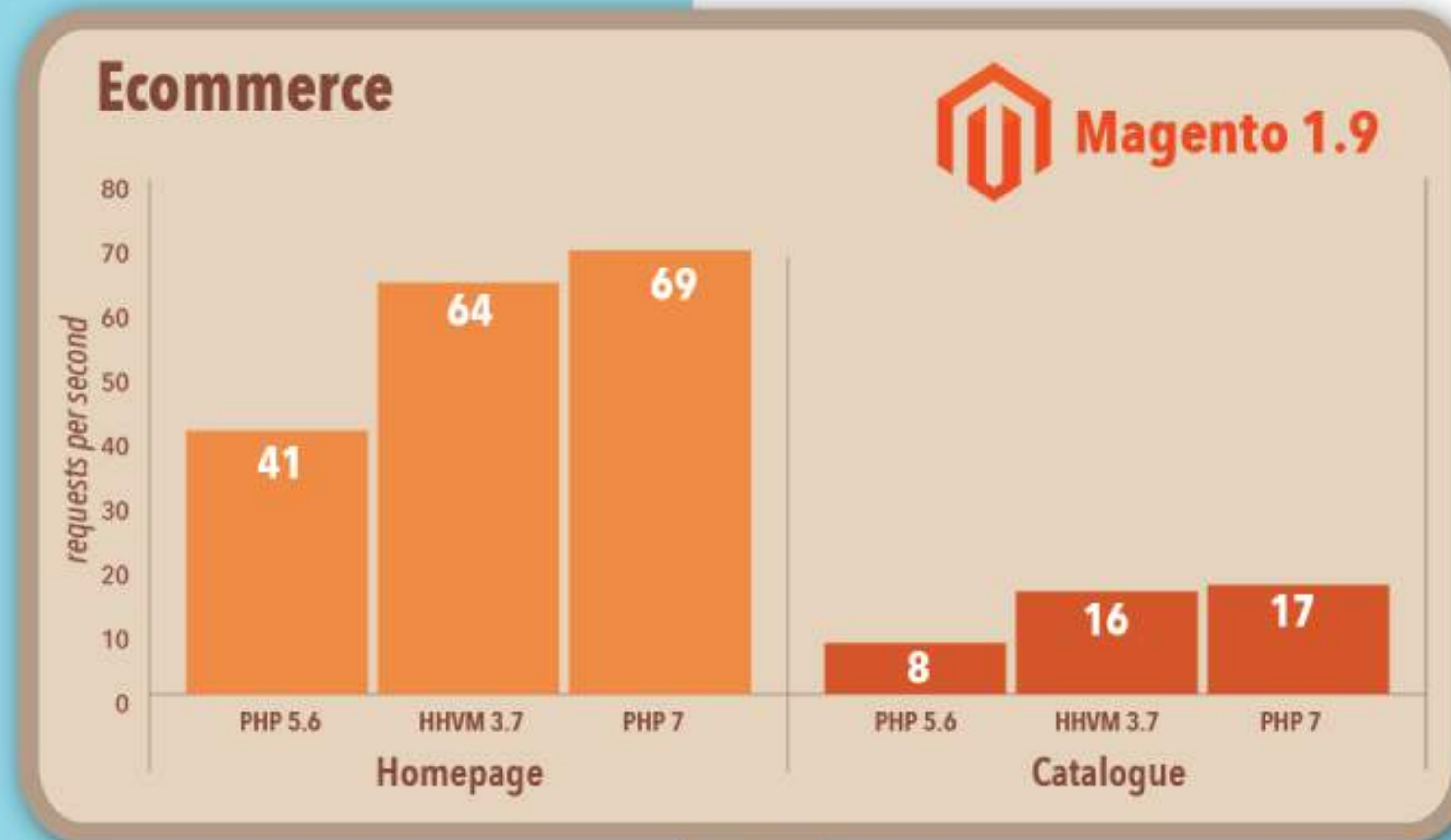
Memory usage:



Source: Badoo Tech Blog

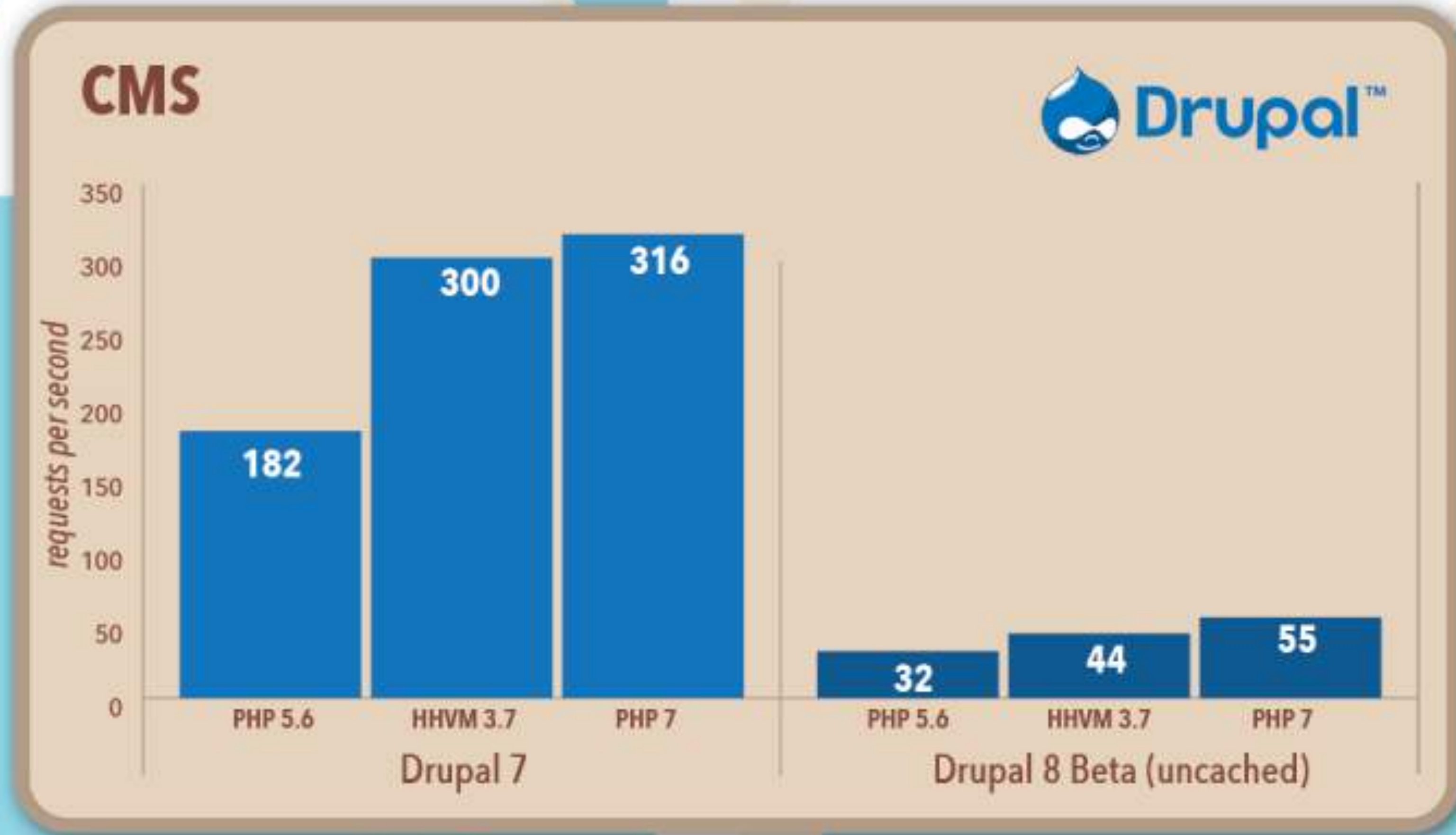
Run up to 3x Magento transactions on the same hardware

With execution time more than twice as fast compared to PHP 5.6 and 30% lower memory consumption - servers running PHP 7 will be able to serve up to 3x as many requests as those running PHP 5.6.



Source: Zend Technologies

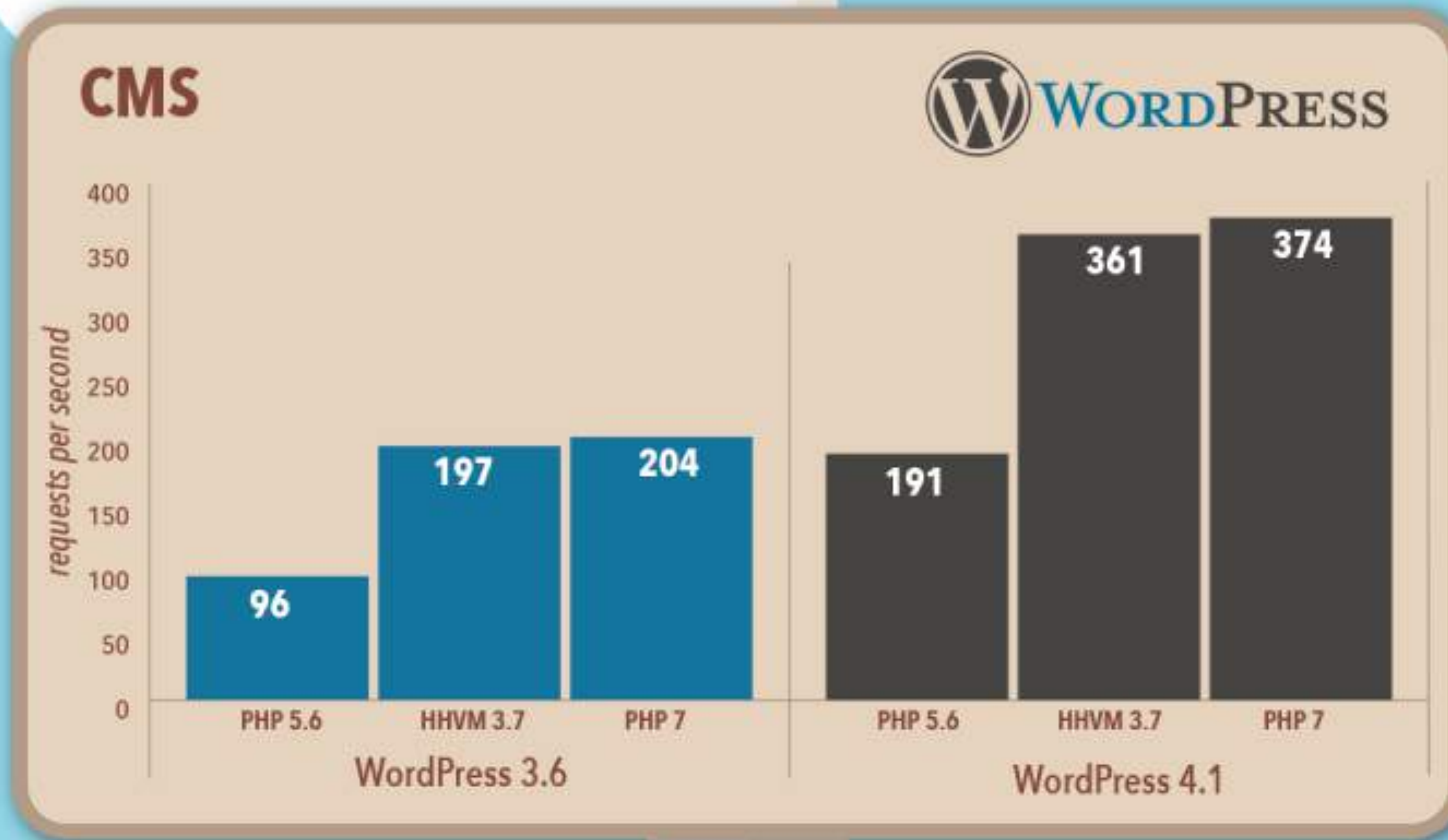
Drupal 8 runs 72% faster
with PHP 7



Source: Zend Technologies

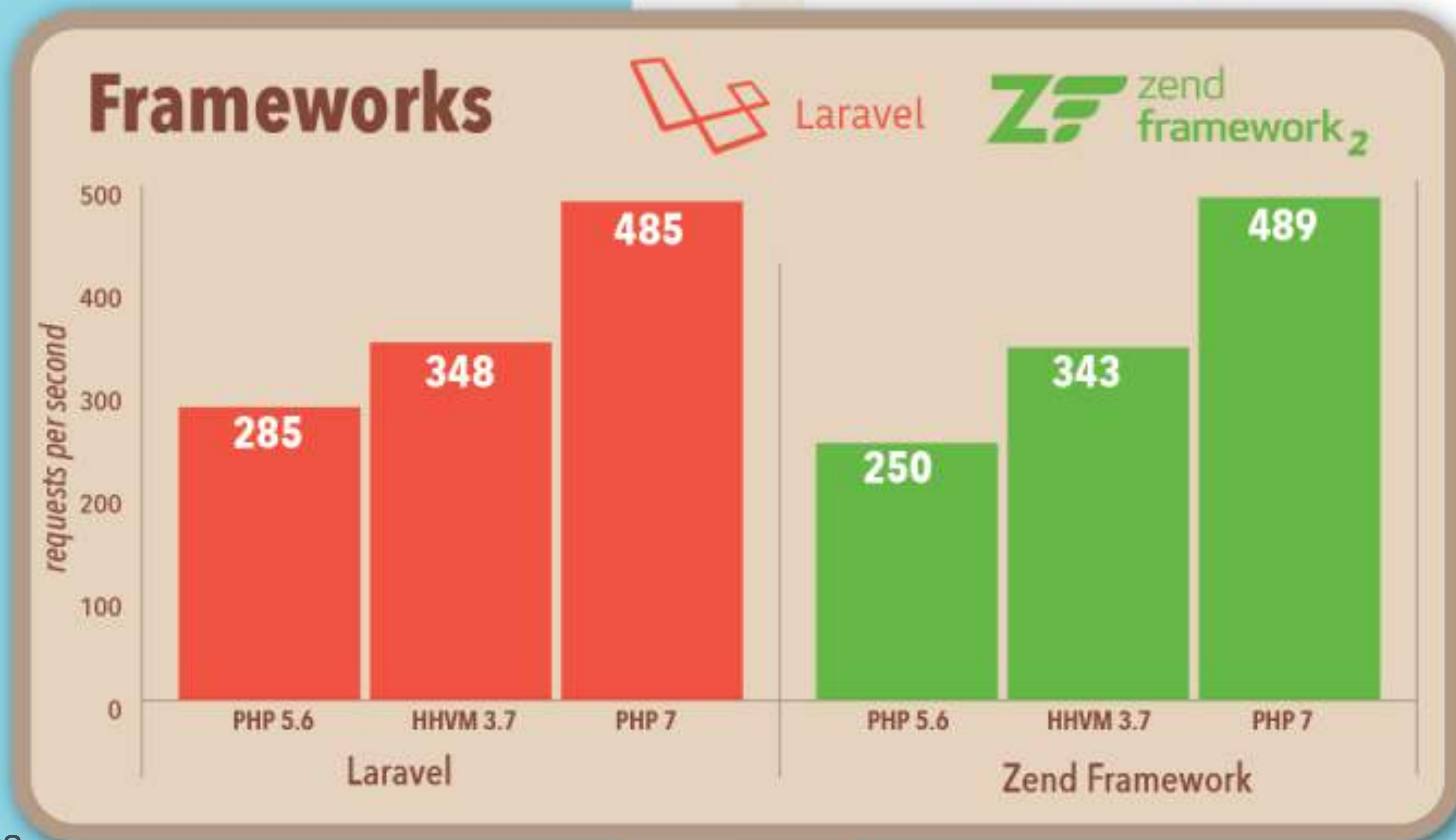
WordPress Screams on PHP 7 You'll need less servers to serve the same amount of users!

One WordPress request on PHP 5.6 executes just under 100M CPU instructions, while PHP 7 only executes 25M to do the same job.



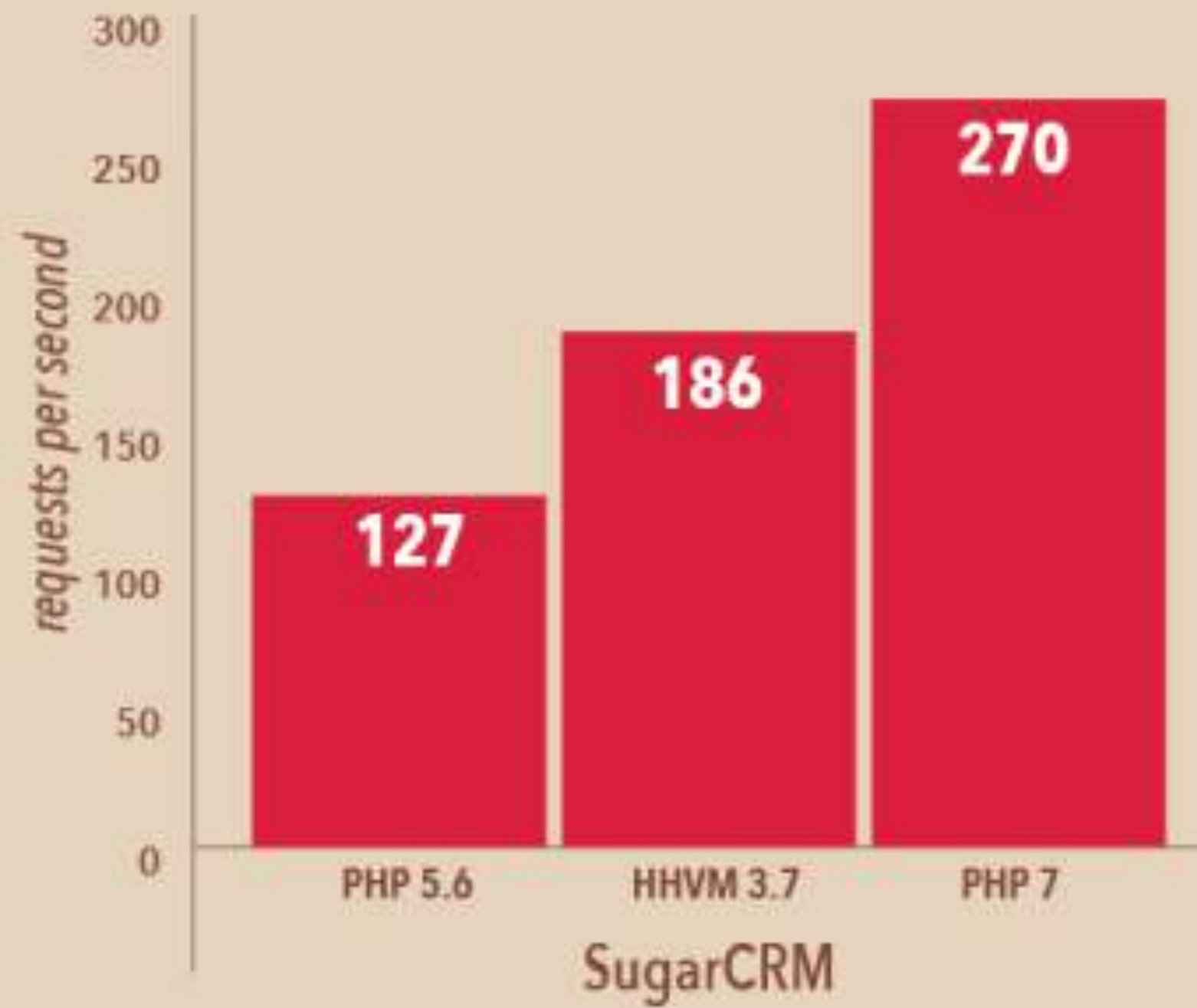
Source: Zend Technologies

We also tested how various
PHP frameworks perform
under PHP 7
SPOILER: Pretty Darn Well!



Source: Zend Technologies

CRM

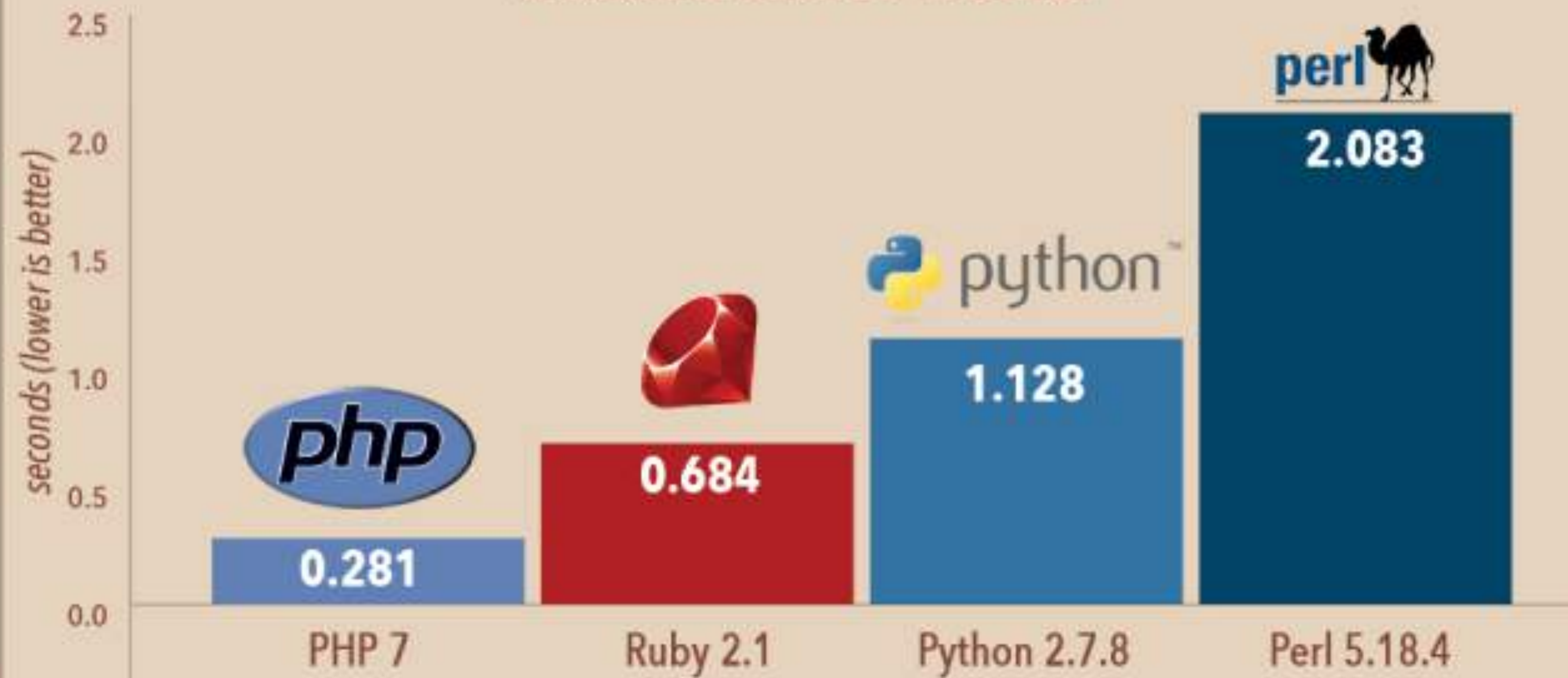


Source: Zend Technologies

Wondering how PHP compares to other dynamic languages?

It has been faster than most of them for a while and, with 7, the gap grows even bigger.

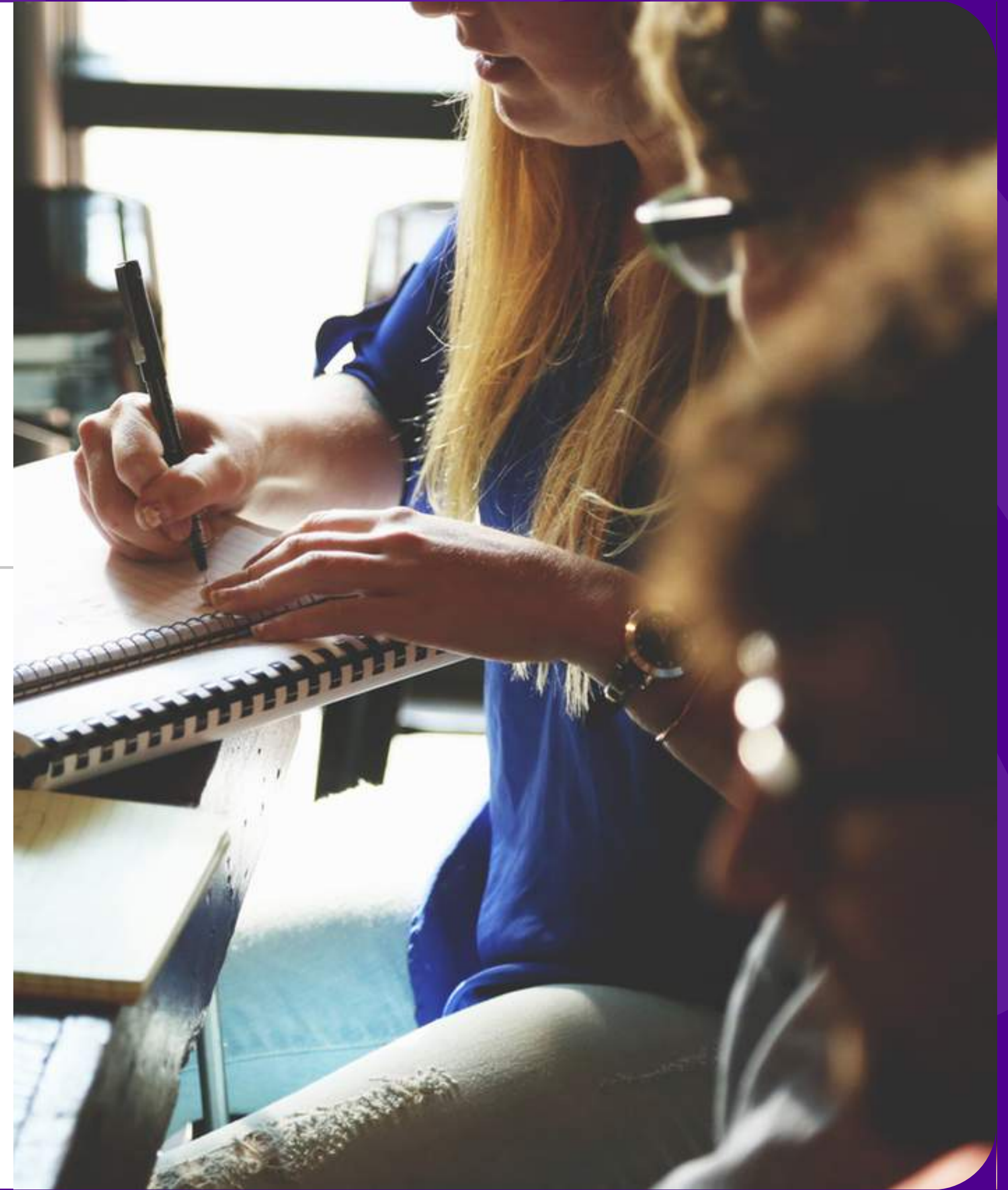
How fast dynamic languages render a Mandelbrot fractal



Source: Zend Technologies

PHP-FIG

Community Standards



History of PHP-FIG

Started organically at php[tek] 2009
5 people originally

Goal: Make PHP frameworks
work together more easily

Has grown since then,
~40 members now



Numerous Standards

- PSR-0 & PSR-4: Autoloading
- PSR-1 & PSR-2: Coding Style
- PSR-3: Logging Interface
- PSR-6: Caching Interface
- PSR-7: HTTP Message Interface
- PSR-13: Hypermedia Links



Composer

Ecosystem innovation



Dependency Manager for PHP

Allows complex systems to be built effortlessly

Adopted by most major frameworks: Symfony, Laravel, Drupal & more

Handles autoloading classes for you



getcomposer.org

JSON Configuration

Specify minimum
PHP version

List of
packages & versions

Options such as
autoload, dev tools...

```
{
  "require": {
    "php": ">=5.6.0",
    "phergie/phergie-irc-client-react": "~3.1",
    "phergie/phergie-irc-connection": "~2.0",
    "phergie/phergie-irc-event": "~1.0",
    "evenement/evenement": "~2.0",
    "monolog/monolog": "~1.6"
  },
  "require-dev": {
    "phergie/phergie-irc-bot-react-development": "~1.0"
  },
  "autoload": {
    "psr-4": {
      "Phergie\\Irc\\Bot\\React\\": "src"
    }
  }
}
```


composer up

Fetches all dependancies
and installs them

```
windlass-3:/Users/eli/Projects/ps> composer up
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 9 installs, 0 updates, 0 removals
  - Installing guzzlehttp/promises (1.3.0)
    Downloading: 100%

  - Installing psr/http-message (1.0.1)
    Downloading: 100%

  - Installing guzzlehttp/psr7 (1.3.1)
    Downloading: 100%

  - Installing guzzlehttp/guzzle (6.2.2)
    Downloading: 100%

  - Installing symfony/polyfill-mbstring (v1.3.0)
    Downloading: 100%

  - Installing symfony/dom-crawler (v3.2.0)
    Downloading: 100%

  - Installing symfony/css-selector (v3.2.0)
    Downloading: 100%

  - Installing symfony/browser-kit (v3.2.0)
    Downloading: 100%

  - Installing fabpot/goutte (v3.2.0)
    Downloading: 100%

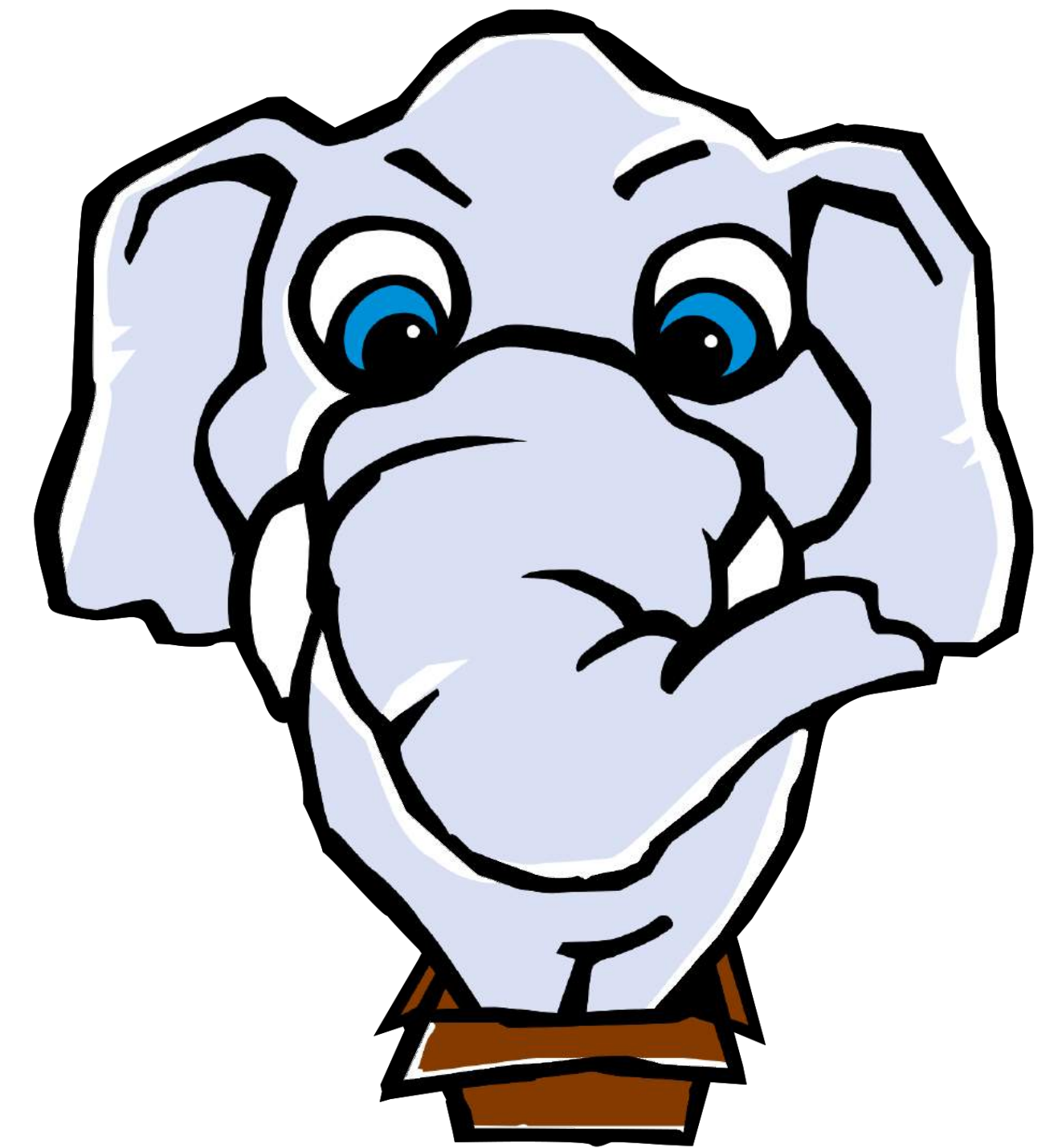
symfony/browser-kit suggests installing symfony/process ( )
Writing lock file
Generating autoload files
```

Package Repository

Community run
main repository

Packages actually live elsewhere

Can be loaded via
Git, SVN, Zip file, ...



packagist.org

Same JSON File

Add descriptors to
composer.json

Config becomes
usable package

Submit to Packagist

```
{
  "name": "laravel/framework",
  "description": "The Laravel Framework.",
  "keywords": ["framework", "laravel"],
  "license": "MIT",
  "homepage": "https://laravel.com",
  "support": {
    "issues": "https://github.com/laravel/framework/issues",
    "source": "https://github.com/laravel/framework"
  },
  "authors": [
    {
      "name": "Taylor Otwell",
      "email": "taylor@laravel.com"
    }
  ],
  "require": {
    "php": ">=5.6.4",
    "ext-mbstring": "*",
    "ext-openssl": "*",
    "classpreloader/classpreloader": "~3.0",
    "doctrine/inflector": "~1.0",
    "jeremeamia/superclosure": "~2.2",
```

Smaller Stuff

Late static binding

Goto support

Short array syntax

Built in Web Server

finally{}

Generators

Built in Opcache

Variadic functions

Constant scalar expressions

Null coalescing operator (??)

UFO operator (<=>)

Secure password hashing

Pardon a brief commercial interruption



PHP[TEK] 2017

The Premier PHP Conference
12th Annual Edition

MAY 24-26 — ATLANTA

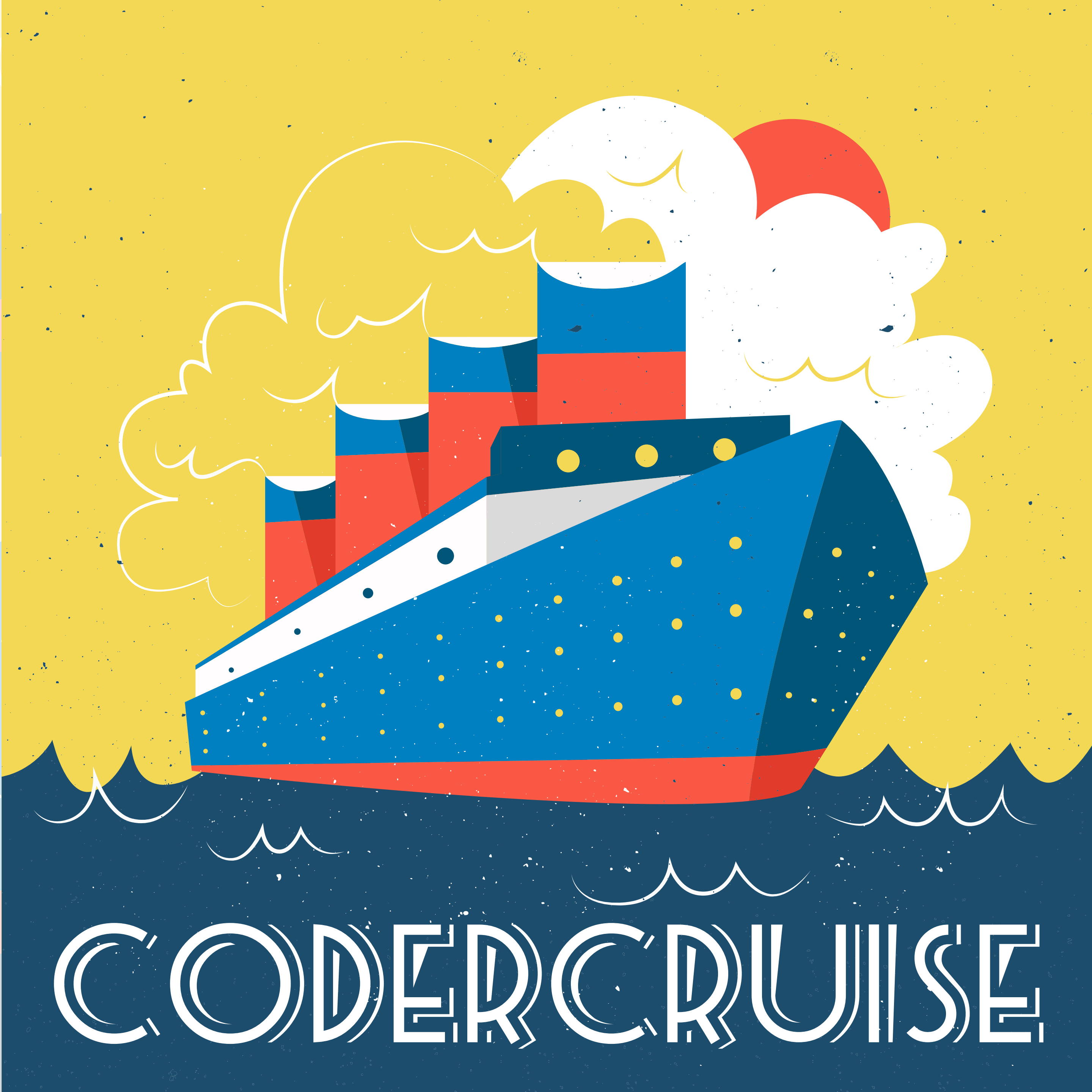
tek.phparch.com

CODERCruise.COM

JULY 16-23, 2017

PORT OF NEW ORLEANS

***Montego Bay
Grand Cayman
Cozumel***



CODERCruise

For this presentation & more:
eliw.com

Twitter: @EliW

One for All Events:
www.oneforall.events



*One for All
Events*