Another PHP

By Max Gopey

Another PHP

- Chapter 1. Coding standards
- Chapter 2. OOP
- Chapter 3. Everything Else
- Chapter 4.

Chapter 1

Coding standards

- PSR
- Zend
- PEAR
- Wordpress
- Symphony
- Mediawiki
- FuelPHP
- CakePHP
- CodeIgniter
- Laravel
- ...
- Are
- you
- guys
- **MAD?**

Chapter 2

OOP

Why do we need objects and classes?

Chapter 3

Everything Else

Chapter 3

Gitlab Composer

Click me

```php
$fetch_refs = function($project) use ($fetch_ref, $repos) {
    $datas = array();
    try {
        foreach (array_merge($repos->branches($project['id']),
                             $repos->tags($project['id'])) as $ref) {
            foreach ($fetch_ref($project, $ref) as $version => $data) {
                $datas[$version] = $data;
            }
        }
    } catch (RuntimeException $e) {
        // The repo has no commits — skipping it.
    }
    return $datas;
};
```

Chapter 3

Array Traversing

- array_column
- array_map
- array_search
- array_reduce
- array_filter
- array_walk
- every / some

```php
$users = [
    ['id' => 123, 'first_name' => 'Max', 'last_name' => 'Gopey'],
    ['id' => 456, 'first_name' => 'Bob', 'last_name' => 'Doe'],
    ['id' => 789, 'first_name' => 'Alice', 'last_name' => 'Doe'],
];
$lastNames = array_column($users, 'last_name', 'id');
print_r($lastNames);
```

```
Array
(
    [123] => Max
    [456] => Bob
    [789] => Alice
)
```

```php
$users = [
    ['id' => 123, 'first_name' => 'Max', 'last_name' => 'Gopey'],
    ['id' => 456, 'first_name' => 'Bob', 'last_name' => 'Doe'],
    ['id' => 789, 'first_name' => 'Alice', 'last_name' => 'Doe'],
];
$fullNames = array_map(function($user) {
    return $user['first_name'] . ' ' . $user['last_name'];
}, $users);
print_r($fullNames);
```

```
Array
(
    [0] => Max Gopey
    [1] => Bob Doe
    [2] => Alice Doe
)
```

```php
$users = [
    ['id' => 123, 'first_name' => 'Max', 'last_name' => 'Gopey'],
    ['id' => 456, 'first_name' => 'Bob', 'last_name' => 'Doe'],
    ['id' => 789, 'first_name' => 'Alice', 'last_name' => 'Doe'],
];
$index = array_search('Alice Doe', array_map(function($user) {
    return $user['first_name'] . ' ' . $user['last_name'];
}, $users));
print_r($index);
```

```
2
```

```php
$users = [
    ['first_name' => 'Max', 'last_name' => 'Gopey', 'company' => 'CGI'],
    ['first_name' => 'Bob', 'last_name' => 'Doe', 'company' => 'Google'],
    ['first_name' => 'Alice', 'last_name' => 'Doe', 'company' => 'Google'],
];

$byCompany = array_reduce($users, function($result, $user) {
    @$result[$user['company']][] = $user['first_name'] . ' ' . $user['last_name'];
    return $result;
}, []);
print_r($byCompany);
```

```
Array (
    [CGI] => Array (
        [0] => Max Gopey
    )
    [Google] => Array (
        [0] => Bob Doe
        [1] => Alice Doe
    )
)
```

```php
$users = [
    ['first_name' => 'Max', 'last_name' => 'Gopey', 'company' => 'CGI'],
    ['first_name' => 'Bob', 'last_name' => 'Doe', 'company' => 'Google'],
    ['first_name' => 'Alice', 'last_name' => 'Doe', 'company' => 'Google'],
];

$CgiUsers = array_filter($users, function($user) {
    return $user['company'] === 'CGI';
});
print_r($CgiUsers);
```

```
Array (
    [0] => Array (
        [first_name] => Max
        [last_name] => Gopey
        [company] => CGI
    )
)
```

```php
$users = [
    ['first_name' => 'Max', 'last_name' => 'Gopey', 'company' => 'CGI'],
    ['first_name' => 'Bob', 'last_name' => 'Doe', 'company' => 'Google'],
    ['first_name' => 'Alice', 'last_name' => 'Doe', 'company' => 'Google'],
];
array_walk($users, function(&$user, $index) {
    unset($user['last_name'], $user['company']);
    $user['first_name'] .= ' ❤';
});
print_r($users);
```

```
Array (
    [0] => Array (
        [first_name] => Max ❤
    )
    [1] => Array (
        [first_name] => Bob ❤
    )
    [2] => Array(
        [first_name] => Alice ❤
    )
)
```

```php
function some($array, $callback) {
    foreach ($array as $item) {
        if ($callback($item)) {
            return true;
        }
    }
    return false;
}
function every($array, $callback) {
    return !some($array, function($item) use ($callback) {
        return !$callback($item);
    });
}
var_dump(every([1, 2, 3], function ($item) {return $item > 0;}));
var_dump(every([1, -2, 3], function ($item) {return $item > 0;}));
```

```
bool(true)
bool(false)
```

```php
function getBobsAndAlicesWithD($users) {
    return array_reduce(
        array_filter(
            array_map(function($user) {
                return $user['last_name'] . ', ' . $user['first_name'];
            }, $users),
            function($name) {
                return stripos($name, 'd') === 0;
            }
        ),
        function($result, $value) {
            $target = stripos($value, 'bob') !== false ? 'bobs' : 'alices';
            $result[$target][] = $value;
            return $result;
        },
        ['bobs' => [], 'alices' => []]
    );
}
```

```php
$users = [
    ['first_name' => 'Max', 'last_name' => 'Gopey', 'company' => 'CGI'],
    ['first_name' => 'Bob', 'last_name' => 'Doe', 'company' => 'Google'],
    ['first_name' => 'Alice', 'last_name' => 'Doe', 'company' => 'Google'],
];

print_r(getBobsAndAlicesWithD($users));
```

```
Array
(
    [bobs] => Array (
        [0] => Doe, Bob
    )
    [alices] => Array (
        [0] => Doe, Alice
    )
)
```

Chapter 3

Generators

```
Traversable                    (Interface)
├── Iterator                   (Interface)
│   └── Generator              (Class)
└── IteratorAggregate          (Interface)
```

```php
function garbageGenerator() {
    $n = rand(1, 10);
    while ($n--) {
        yield md5(rand());
    }
}
$garbage = garbageGenerator();
foreach ($garbage as $trash) {
    echo $trash, PHP_EOL;
}
```

```
6e620c902c7088ace3ebf6c96f5dedd5
1340dcc6f3e0e39b4c48f480f5a92d52
c264962d537032be6c3a8a94eda811d4
0bfa2efb3909c105473a4fcaa71b697b
```

```php
function readFileLines($path) {
    $handle = fopen($path, 'r');
    while ($line = fgets($handle)) {
        yield $line;
    }
    fclose($handle);
}

$lines = readFileLines(__FILE__);
foreach($lines as $line) {
    echo $line;
};
```

Symfony\Component\Process\InputStream

Click me

```php
function writer(InputStream $stream) {
    $stream->write('Message 1');
    $stream->write('Message 2');
    yield '2 messages written';
    $stream->write('Message 3');
    $stream->write('Message 4');
    yield '2 messages written';
    $stream->write('Message 5');
    $stream->write('Message 6');
    yield '2 messages written';
}

function reader(InputStream $stream) {
    foreach ($stream as $line) {
        if (strlen($line)) {
            yield $line;
        } else {
            $stream->close();
        }
    }
}
```

```php
$stream = new InputStream();
$queue[] = writer($stream);
$queue[] = reader($stream);

while (true) {
    $continue = array_reduce(
        $queue,
        function($result, Iterator $queueItem) {
            if ($valid = $queueItem->valid()) {
                echo $queueItem->current(), "\n";
                $queueItem->next();
            }
            return $result || $valid;
        },
        false);
    if (!$continue) {
        break;
    }
}
```

```
2 messages written
Message 1
2 messages written
Message 2
2 messages written
Message 3
Message 4
Message 5
Message 6
```

Chapter 3

Functional programming

```php
function getBobsAndAlicesWithD($users) {
    return array_reduce(
        array_filter(
            array_map(function($user) {
                return $user['last_name'] . ', ' . $user['first_name'];
            }, $users),
            function($name) {
                return stripos($name, 'd') === 0;
            }
        ),
        function($result, $value) {
            $target = stripos($value, 'bob') !== false ? 'bobs' : 'alices';
            $result[$target][] = $value;
            return $result;
        },
        ['bobs' => [], 'alices' => []]
    );
}
```

Chapter 3

Non-standard PHP library (NSPL)

Click me

```php
$startsWith = function ($string, $substing) {
    return stripos($string, $substing) === 0;
};
$contains = function($string, $substing) {
    return stripos($string, $substing) !== false;
};
$getFullName = function ($firstName, $lastName) {
    return $lastName . ', ' . $firstName;
};

$startsWithD = f\rpartial($startsWith, 'd');
$isBob = f\rpartial($contains, 'bob');

$getFullNameFromUser = function ($user) use ($getFullName) {
    return $getFullName($user['first_name'], $user['last_name']);
};
$getStackKey = function($name) use ($isBob) {
    return $isBob($name) ? 'bobs' : 'alices';
};
$putToCorrectStack = function($stacks, $value) use ($getStackKey) {
    $stacks[$getStackKey($value)][] = $value;
    return $stacks;
};
```

```php
$getBobsAndAlicesWithD = function ($users)
    use ($startsWithD, $getFullNameFromUser, $putToCorrectStack) {
    return f\pipe(
        $users,
        f\partial(a\map, $getFullNameFromUser),
        f\partial(a\filter, $startsWithD),
        f\ppartial(a\reduce, [
            Ø => $putToCorrectStack,
            2 => ['bobs' => [], 'alices' => []]
        ])
    );
};

print_r($getBobsAndAlicesWithD($users));
```

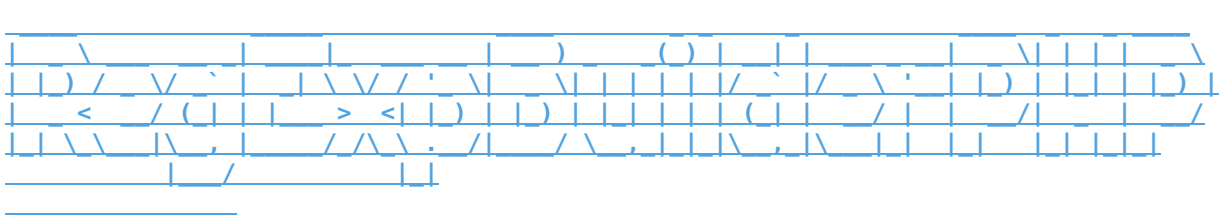
Chapter 3

Obvious logical operations

```
01.  if (anyOf([1, 3, 5])->is(5)) {
02.      // do something
03.  }
04.  if (anyOf([$name, $surname])->matches('/^\w+$/') {
05.      // do something
06.  }
07.  if (allOf([1, 3, 5])->areNot(6)) {
08.      // do something
09.  }
10.  if (either($condition1)->or($condition2)) {
11.      // do something
12.  }
13.  if (neither($x)->nor($y)) {
14.      // do something
15.  }
16.  if (the($x)->isNeither(5)->nor(10)) {
17.      // do something
18.  }
19.  if (the($x)->isGreaterThan(5)->butLessThan(10)) {
20.      // do something
21.  }
```

Chapter 3

Obvious regexp



```php
$regExp = $builder
    ->startOfInput()
    ->exactly(4)->digits()
    ->then("_")
    ->exactly(2)->digits()
    ->then("_")
    ->min(3)->max(10)->letters()
    ->then(".")
    ->anyOf(array("png", "jpg", "gif"))
    ->endOfInput()
    ->getRegExp();

//true
$regExp->matches("2020_10_hund.jpg");
$regExp->matches("2030_11_katze.png");
$regExp->matches("4000_99_maus.gif");

//false
$regExp->matches("123_00_nein.gif");
$regExp->matches("4000_0_nein.pdf");
$regExp->matches("201505_nein.jpg");
```

Useful links

Generators and Coroutines

- Хабр: Coroutines в PHP и работа с неблокирующими функциями
- Github: Asynchronous coroutines for PHP 7.
- A Curious Course on Coroutines and Concurrency
- Symfony/Component/Process/InputStream.php

Functional programming

- Github: Non-standard PHP library (NSPL) - compact functional programming oriented code

Human-readable regular expressions

- Github: RegexpBuilderPHP
- Github: PHPVerbalExpressions

Kittens

- Youtube: The funniest kitten in the world