ekino.

Mutability killed me, Immutability saved me

La mutabilité m’a tué, l’immutabilité m’a sauvé

FORUMPHP PARIS2023

Who I am?

Sadetdin EYILI

- https://github.com/sad270
- https://twitter.com/SadEYILI

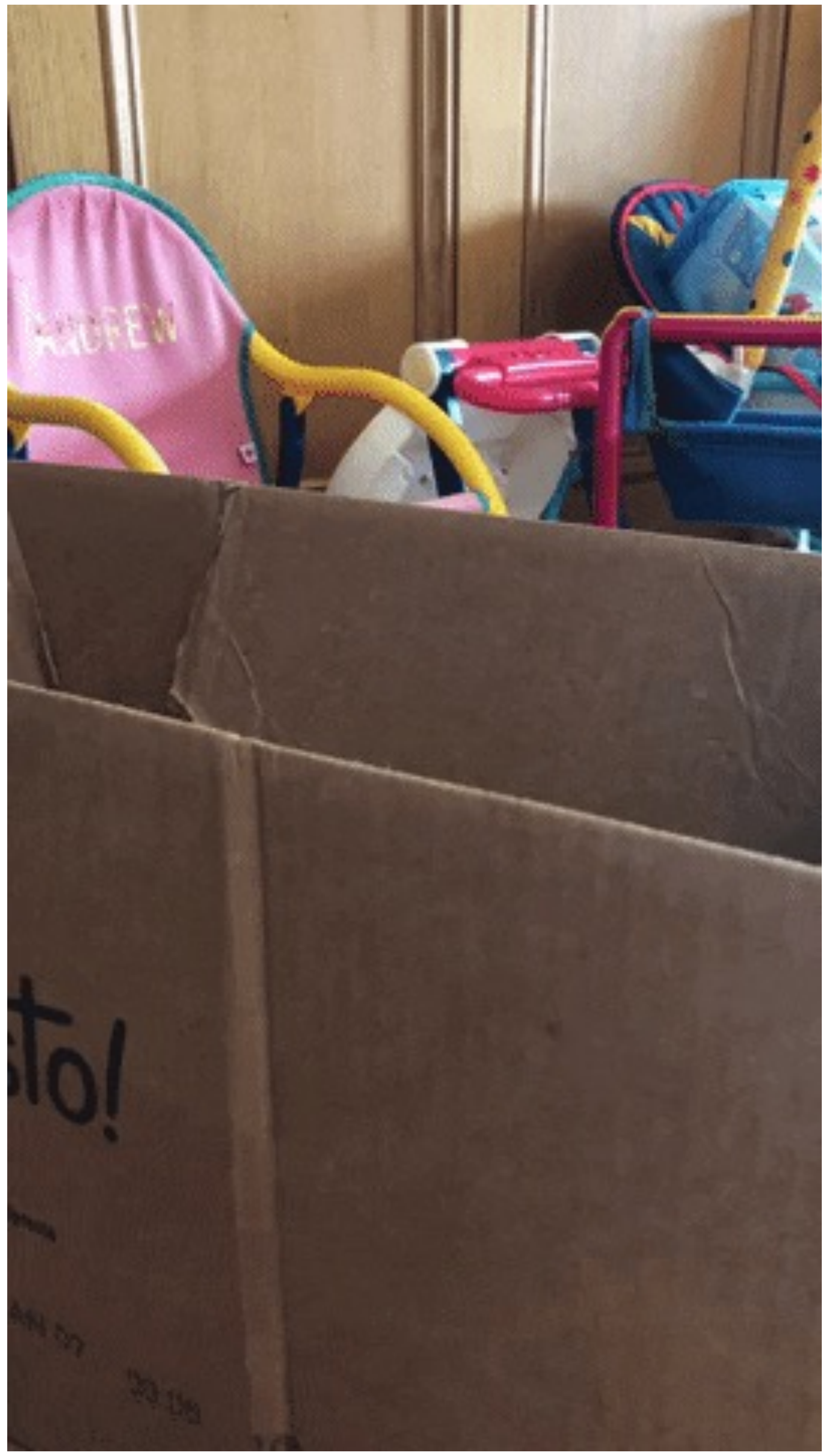
Initial commit

- Learned web development with my cousin
- Our first project: a website

Last commit

- PHP engineer at ekino
- I have 2 ducks
- I'm presenting my first conference

Join ekino if you want to listen stories about my familly, my ducks or Yvan (my car)



→ Why am I here?

Why am I here?

To share with you my experience with immutable objects.

Immuwhat?

Immutable object

In object-oriented and functional programming, an immutable object (unchangeable object) is an object whose state cannot be modified after it is created. This is in contrast to a mutable object (changeable object), which can be modified after it is created.

Wikipedia

ekino.

FORUMPHP PARIS2023

« cannot be modified after it is created »

What is the goal of this?

Why would we want to have an object that can't be modified?

Is it a constant?

WTF?

Why?

Gne...

I don't understand...

ekino.

→ Why am I here?

Why am I here?

To share with you my experience **WITHOUT** immutable objects.

DISCLAIMER

- This conference is a reconstruction of facts that happened in a real project
- Confidentiality is preserved 🔒
- Creativity and imagination!

→ Imagine a ~~world~~ feature without immutability

Let’s code together a simple feature

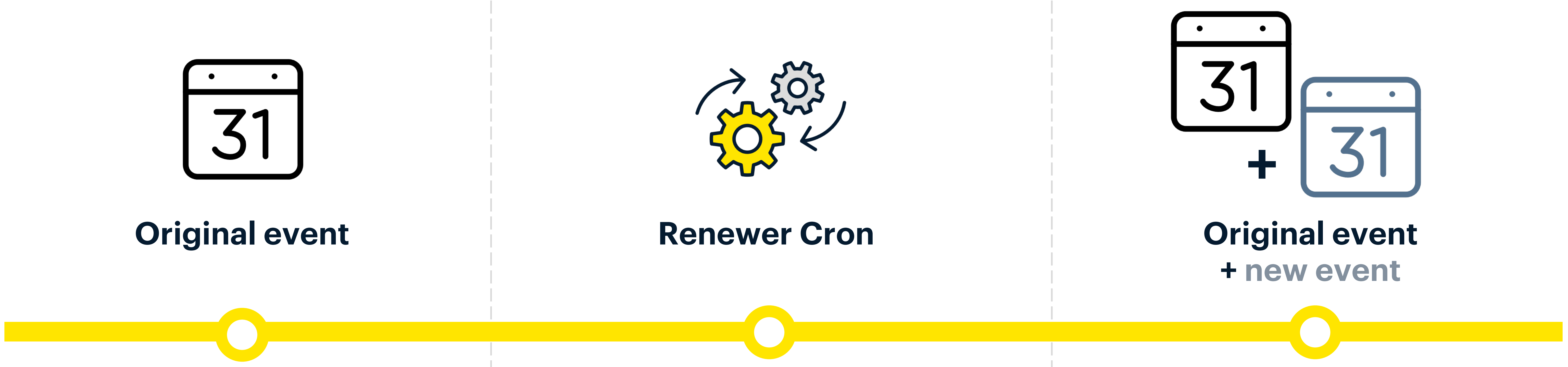
We want **Events** with:

Start date
Duration
Recurrence
Parent event (optional)

We want a **cronjob** to renew these events

We want to **renew** them without knowing when we renew them.

Let’s code together a simple feature



Let’s code together a simple feature

Original event

Renewer Cron

Original event
+ new event

Original event
Start at: 01/01/2023
Duration: 1 day
Recurrence: 3 months
Parent: null (initial event)

Original event
Start at: 01/01/2023
Duration: 1 day
Recurrence: 3 months
Parent: null (initial event)

New event
Start at: 01/04/2023
Duration: 1 day
Recurrence: 3 months
Parent: Original event

```php
class RecurrentEvent
{
    public function __construct(
        public \DateTime $startAt,
        public \DateInterval $duration,
        public \DateInterval $recurrenceInterval,
        public ?RecurrentEvent $parentEvent = null,
    ) {
    }
}
```

```php
$event = new RecurrentEvent(
    new \DateTime(),
    new \DateInterval('P1D'),
    new \DateInterval('P3M'),
);
```

Constructor property promotion – php.net

```php
class CronCommand {
   public function renewEvents(array $events) {
      foreach ($events as $parentEvent) {
         $newEvent = $this->renew($parentEvent);
         $this->save($newEvent);
      }
   }

   private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
      return new RecurrentEvent(
         $parentEvent->startAt->add($parentEvent->recurrenceInterval),
         $parentEvent->duration,
         $parentEvent->recurrenceInterval,
         $parentEvent,
      );
   }
}
```

- ☑ Check the new event dates are OK!
- ☑ Deploy!
- ☑ Client said it works
- ☑ Client is happy
- ☑ I am happy
- ☑ We are happy!

☑ Check the new event dates are OK!

☑ Deploy!

☑ Client said it works

☑ Client is happy

☑ I am happy

☑ We are happy!

... Few days later ...

Client is not happy!

Client's old events dates have changed!

WHAT? But how?

Let’s debug a simple feature

Original event

Original event
Start at: 01/01/2023
Duration: 1 day
Recurrence: 3 months
Parent: null (initial event)

Renewer Cron

Original event
+ new event

~~**Original event**~~
~~**Start at:** 01/01/2023~~
~~**Duration:** 1 day~~
~~**Recurrence:** 3 months~~
~~**Parent:** null (initial event)~~

New event
Start at: 01/04/2023
Duration: 1 day
Recurrence: 3 months
Parent: Original event

Original event
Start at: 01/04/2023
Duration: 1 day
Recurrence: 3 months
Parent: null (initial event)

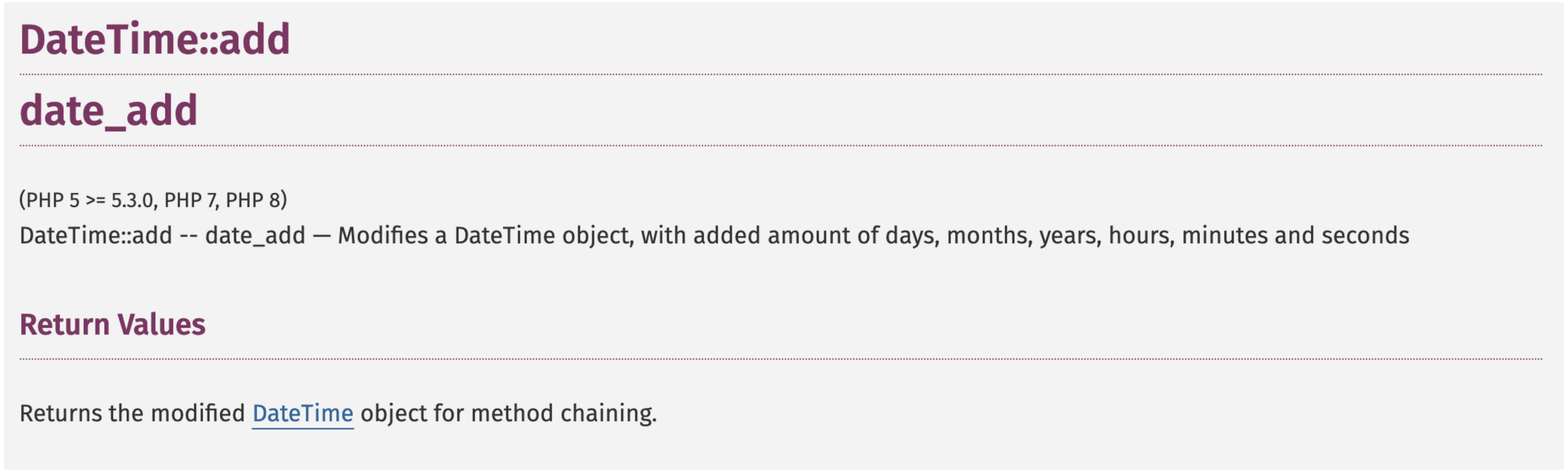
« Objects are passed by reference by default » *

* It’s not completely true! But it’s not the subject! php.net

```
class CronCommand {
   public function renewEvents(array $events) {
      foreach ($events as $parentEvent) {
         $newEvent = $this->renew($parentEvent);
         $this->save($newEvent);
      }
   }

   private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
      return new RecurrentEvent(
         $parentEvent->startAt->add($parentEvent->recurrenceInterval),
         $parentEvent->duration,
         $parentEvent->recurrenceInterval,
         $parentEvent,
      );
   }
}
```

1 Send parent event to ”renew” method

2 Parent event is “passed by reference”



DateTime::add

date_add

(PHP 5 >= 5.3.0, PHP 7, PHP 8)

DateTime::add -- date_add — Modifies a DateTime object, with added amount of days, months, years, hours, minutes and seconds

Return Values

Returns the modified DateTime object for method chaining.

php.net

ekino.

FORUMPHP PARIS2023

```
class CronCommand {
    public function renewEvents(array $events) {
        foreach ($events as $parentEvent) {
            $newEvent = $this->renew($parentEvent);
            $this->save($newEvent);
        }
    }

    private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
        return new RecurrentEvent(
            $parentEvent->startAt->add($parentEvent->recurrenceInterval),
            $parentEvent->duration,
            $parentEvent->recurrenceInterval,
            $parentEvent,
        );
    }
}
```

1 Send parent event to "renew" method

2 Parent event is "passed by reference"

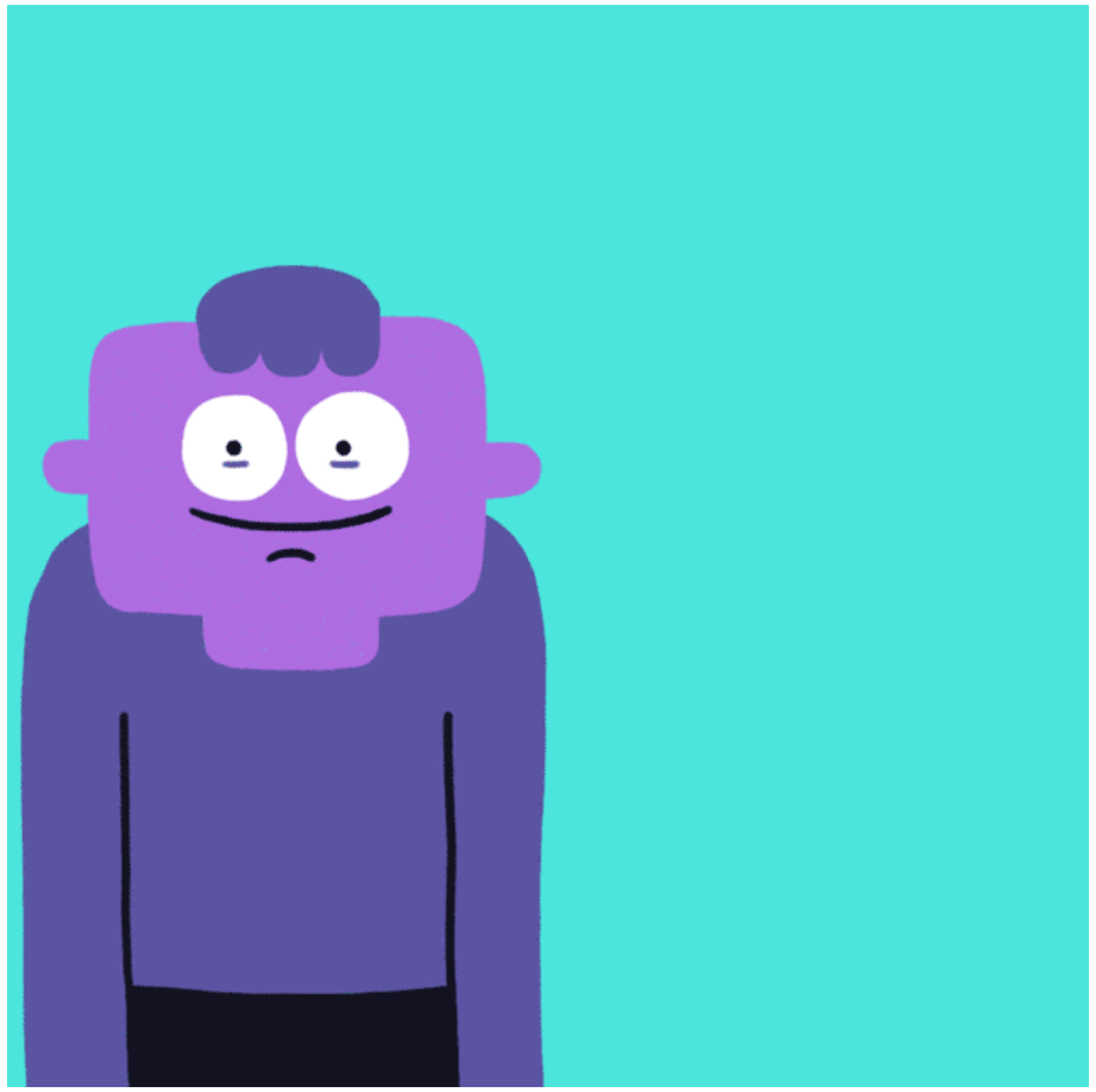
3 We modify the parent event startAt and set the new event startAt, at the same time.

ekino.

→ Imagine a ~~world~~ bugfix without immutability

How to fix it fastly?

One word “clone“



```php
class CronCommand {
    public function renewEvents(array $events) {
        foreach ($events as $parentEvent) {
            $newEvent = $this->renew($parentEvent);
            $this->save($newEvent);
        }
    }

    private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
        $clonedParentStartAt = clone $parentEvent->startAt;

        return new RecurrentEvent(
            $clonedParentStartAt->add($parentEvent->recurrenceInterval),
            $parentEvent->duration,
            $parentEvent->recurrenceInterval,
            $parentEvent,
        );
    }
}
```

1 Send parent event to "renew" method

2 Parent event is "passed by reference"

3 Clone

4 Modify the clone

How to fix it fastly?

One word “clone“

- Easy and fast
- But not the real solution

Why it's not the real solution?

- Display event’s end date?

ekino.

FORUMPHP PARIS2023

Why it's not the real solution?

- Display event’s end date?

```
echo $event->startAt->add($event->duration)->format("Y-m-d H:i:s");
```

Why it's not the real solution?

- Display event’s end date? Clone

```
echo $event->startAt->add($event->duration)->format("Y-m-d H:i:s");
```

```
$clonedStartAt = clone $event->startAt;
echo $clonedStartAt->add($event->recurrenceInterval)->format("Y-m-d H:i:s");
```

Why it's not the real solution?

- Display event’s end date? Clone
- Display next event end date, without renewing it? **Clone**
- Check if event A start in following 3 days after event B? **CLONE**

ekino.

FORUMPHP PARIS2023

Why it's not the real solution?

- Display event’s end date? Clone
- Display next event end date, without renewing it? **Clone**
- Check if event A start in following 3 days after event B? **CLONE**
- *Lorem?* CLONE
- *Ipsum?* CLONE
- *Foo?* CLONE
- *Bar?* CLONE

Side effects ~~CLONE!~~

What is the real solution?

What is the real issue?

When I renew my events, it modifies the start date of the original event

What is the real solution?

What is the real issue?

~~When I renew my events, it modifies the start date of the original event~~

When I do some operations on my start date, it modifies my date

What is the real solution ?

One word “Immutability“

ekino.

FORUMPHP PARIS2023

But… I left the project ☹

How I met ~~your mother~~ immutability

ekino - 2021

- Internal talks
  - Learn new subject
  - Increase teammates knowledge
  - Learn to be a speaker
- List of interesting subjects
  - “Immutable object”

ekino.

FORUMPHP PARIS2023

How I learn immutability?

- Wikipedia… I didn’t understand
- Listen other talks, conferences… partially understand, but “WHY” we need it?

My immutability talk = Background task with low priority

- Immutability is complex
- Need to focus on my project

ekino.

→

How I met ~~your mother~~ immutability

Season 2

ekino - 2021

- Internal talks
  - Learn new subject
  - Increase teammates knowledge
  - Learn to be a speaker
- List of interesting subject
  - "Immutable object"

ekino - 2022

- Looking for a subject for an article
  - *"You can write on Immutability for example"*
  - What? How did they know? Why is this subject haunting me?
  - Immutability is too complex!

ekino.

ekino - 2021

- Internal talks
  - Learn new subject
  - Increase teammate knowledge
  - Learn to be a speaker
- List of interesting subject
  - “Immutable object”
- Listen to talks, watch conferences, read articles
- Talking with colleagues (they are really cool !)
- Following ekino’s best practices
- And practice a lot
- I learned immutability, but I didn’t know I learned

ekino - 2022

- Looking for a subject for an article

Imagine a ~~world~~ feature with immutable date

Let's code together a simple feature

Original event

Renewer Cron

Original event
+ new event

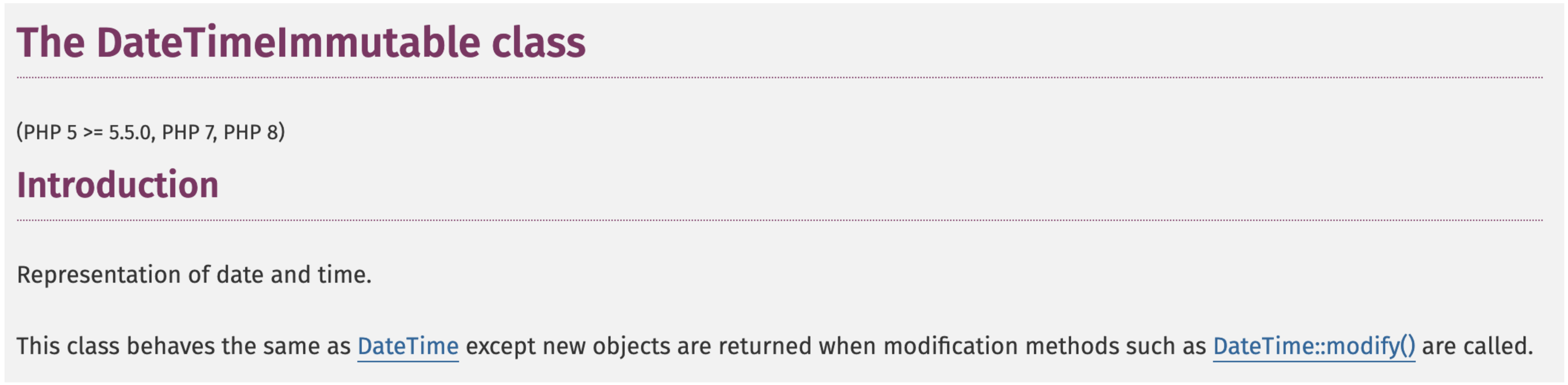
Original event
Start at: 01/01/2023
Duration: 1 day
Recurrence: 3 months
Parent: null (initial event)

Original event
Start at: 01/01/2023
Duration: 1 day
Recurrence: 3 months
Parent: null (initial event)

New event
Start at: 01/04/2023
Duration: 1 day
Recurrence: 3 months
Parent: Original event

Same code, only one change

Replace DateTime by DateTimeImmutable


The DateTimeImmutable class

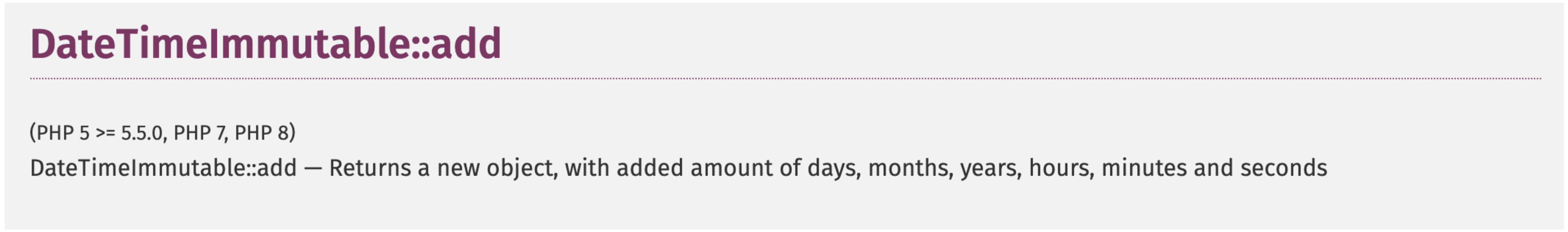
(PHP 5 >= 5.5.0, PHP 7, PHP 8)

Introduction

Representation of date and time.

This class behaves the same as DateTime except new objects are returned when modification methods such as DateTime::modify() are called.

php.net


DateTimeImmutable::add

(PHP 5 >= 5.5.0, PHP 7, PHP 8)

DateTimeImmutable::add — Returns a new object, with added amount of days, months, years, hours, minutes and seconds

php.net

```php
class RecurrentEvent
{
    public function __construct(
        public \DateTimeImmutable $startAt,
        public \DateInterval $duration,
        public \DateInterval $recurrenceInterval,
        public ?RecurrentEvent $parentEvent = null,
    ) {
    }
}
```

```php
$event = new RecurrentEvent(
    new \DateTimeImmutable(),
    new \DateInterval('P1D'),
    new \DateInterval('P3M'),
);
```

```php
class CronCommand {
    public function renewEvents(array $events) {
        foreach ($events as $parentEvent) {
            $newEvent = $this->renew($parentEvent);
            $this->save($newEvent);
        }
    }

    private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
        return new RecurrentEvent(
            $parentEvent->startAt->add($parentEvent->recurrenceInterval),
            $parentEvent->duration,
            $parentEvent->recurrenceInterval,
            $parentEvent,
        );
    }
}
```

ekino.

FORUMPHP PARIS2023

Is it the real solution?

- Display event’s end date?

```
echo $event->startAt->add($event->duration)->format("Y-m-d H:i:s");
```

ekino.

→ Imagine a ~~world~~ feature with immutable object

DateTime === object

DateTime Behaviour === object behaviour

Side effects on DateTime === side effects with objects

We need immutable DateTime === we need immutable objects

Lets upgrade our project and add more features

Lets upgrade our project and add more features

- We need paid events!
- Events are international, we need to manage different currencies

We will use a value object

```php
class Price {
    public function __construct(
        public int $value,
        public Currency $currency, // enum
        public int $precision = 6,
    ) {
    }
}
```

```php
class RecurrentEvent
{
    public function __construct(
        public \DateTimeImmutable $startAt,
        public \DateInterval $duration,
        public \DateInterval $recurrenceInterval,
        public Price $price,
        public ?RecurrentEvent $parentEvent = null,
    ) {
    }
}
```

```php
$event = new RecurrentEvent(
    new \DateTimeImmutable(),
    new \DateInterval('P1D'),
    new \DateInterval('P3M'),
    new Price(12, Currency::Euro),
);
```

```
class CronCommand {
   public function renewEvents(array $events) {
      foreach ($events as $parentEvent) {
         $newEvent = $this->renew($parentEvent);
         $this->save($newEvent);
      }
   }

   private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
      return new RecurrentEvent(
         $parentEvent->startAt->add($parentEvent->recurrenceInterval),
         $parentEvent->duration,
         $parentEvent->recurrenceInterval,
         $parentEvent->price,
         $parentEvent,
      );
   }
}
```

ekino.

FORUMPHP PARIS2023

I said MORE money!

Lets upgrade our project and add MORE money!

- We need paid events!
- Events are international, we need to manage different currencies
- Increase prices when renewing an event

```
class CronCommand {
    public function renewEvents(array $events) {
        foreach ($events as $parentEvent) {
            $newEvent = $this->renew($parentEvent);
            $newEvent->price->value++;
            $this->save($newEvent);
        }
    }

    private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
        return new RecurrentEvent(
            $parentEvent->startAt->add($parentEvent->recurrenceInterval),
            $parentEvent->duration,
            $parentEvent->recurrenceInterval,
            $parentEvent->price,
            $parentEvent,
        );
    }
}
```

wasted

ekino.

FORUMPHP PARIS2023

```php
class CronCommand {
  public function renewEvents(array $events) {
    foreach ($events as $parentEvent) {
      $newEvent = $this->renew($parentEvent);
      $newEvent->price->value++;
      $this->save($newEvent);
    }
  }

  private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
    return new RecurrentEvent(
      $parentEvent->startAt->add($parentEvent->recurrenceInterval),
      $parentEvent->duration,
      $parentEvent->recurrenceInterval,
      $parentEvent->price,
      $parentEvent,
    );
  }
}
```

2 Increment new event price AND parent price

1 Send parent price to __construct method “by reference”

How to fix it?

- Clone? ⊗
- Immutability? ✓

ekino.

→ Imagine a ~~world~~ bugfix with immutability

Immuwhat?

Immutable object

In object-oriented and functional programming, an immutable object (unchangeable object) is an object whose state cannot be modified after it is created. This is in contrast to a mutable object (changeable object), which can be modified after it is created.

Wikipedia

```
readonly class Price {
    public function __construct(
        public int $value,
        public Currency $currency, // enum
        public int $precision = 6,
    ) {
    }
}
```

Readonly !== immutable

```php
class CronCommand {
    public function renewEvents(array $events) {
        foreach ($events as $parentEvent) {
            $newEvent = $this->renew($parentEvent);
            $newEvent->price = new Price($newEvent->price->value + 1, $newEvent->price->currency, $newEvent->price->precision);
            $this->save($newEvent);
        }
    }

    private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
        return new RecurrentEvent(
            $parentEvent->startAt->add($parentEvent->recurrenceInterval),
            $parentEvent->duration,
            $parentEvent->recurrenceInterval,
            $parentEvent->price,
            $parentEvent,
        );
    }
}
```

Your solution is UGLY, where is the DX?

- “Getters”, “Setters” and now ……… “Withers”

ekino.

FORUMPHP PARIS2023

```php
readonly class Price {
    public function __construct(
        public int $value,
        public Currency $currency, // enum
        public int $precision = 6,
    ) {
    }

    public function withValue(int $value): self {
        return new self($value, $this->currency, $this->precision);
    }

    public function withCurrency(Currency $currency): self {
        return new self($this->value, $currency, $this->precision);
    }

    public function withPrecision(int $precision): self {
        return new self($this->value, $this->currency, $precision);
    }
}
```

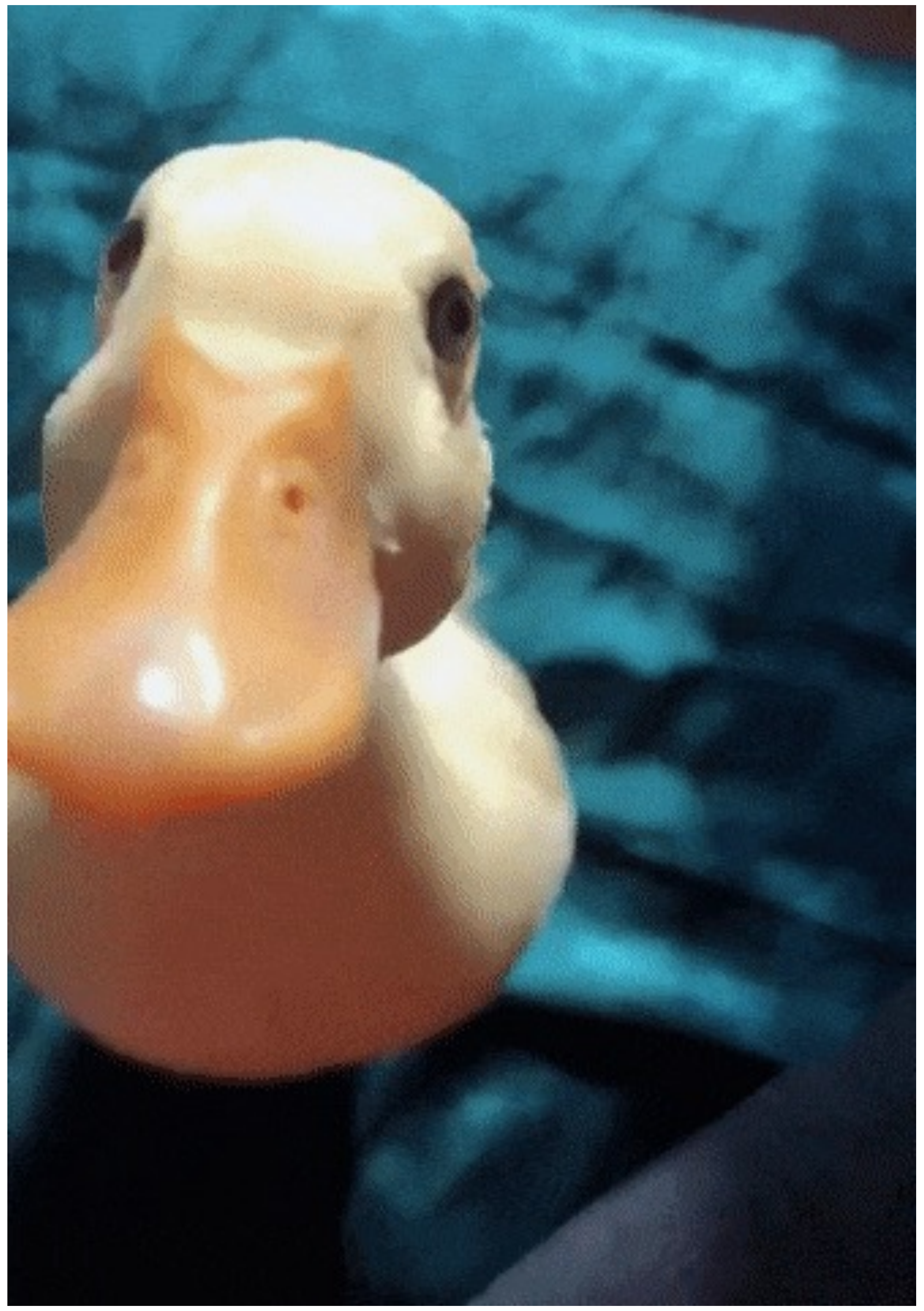
```php
class CronCommand {
   public function renewEvents(array $events) {
      foreach ($events as $parentEvent) {
         $newEvent = $this->renew($parentEvent);
         $newEvent->price = $newEvent->price->withValue($newEvent->price->value + 1);
         $this->save($newEvent);
      }
   }

   private function renew(RecurrentEvent $parentEvent): RecurrentEvent {
      return new RecurrentEvent(
         $parentEvent->startAt->add($parentEvent->recurrenceInterval),
         $parentEvent->duration,
         $parentEvent->recurrenceInterval,
         $parentEvent->price,
         $parentEvent,
      );
   }
}
```

ekino.

FORUMPHP PARIS2023

Important things to know about immutable objects

- Immutable objects must have immutable properties
- DateTimeImmutable is not Immutable



Immutable objects must have immutable properties

```php
readonly class User{
    public function __construct(
        public string $email,
        public \DateTime $createdAt,
    ) {
    }
}

$user = new User('user@ekino.com', new \DateTime());
$user->email = 'haker@ekino.com'; // <= this will not work
$user->createdAt = (new \DateTime())->modify('+1 day'); // <= this will not work
$user->createdAt->modify('+1 day'); // <= this will work
```

Immutable objects must have immutable properties

```php
readonly class UserImmutable{
    public function __construct(
        public string $email,
        public \DateTimeImmutable $createdAt,
    ) {
    }
}

$user = new UserImmutable('user@ekino.com', new \DateTimeImmutable ());
$user->email = 'haker@ekino.com'; // <= this will not work
$user->createdAt = (new \DateTimeImmutable ())->modify('+1 day'); // <= this will not work
$user->createdAt->modify('+1 day'); // <= this will work, but it will not modify user’s createdAt
```

DateTimeImmutable is not Immutable

```php
$date = new \DateTimeImmutable('2022-01-01');
$date->__construct('2023-01-01');
echo $date->format('Y-m-d H:i:s'); // 2023-01-01 00:00:00
```

ekino.

FORUMPHP PARIS2023

DateTimeImmutable is not Immutable

```php
$date = new \DateTimeImmutable('2022-01-01');
$date->__construct('2023-01-01');
echo $date->format('Y-m-d H:i:s'); // 2023-01-01 00:00:00
```

```php
class UserImmutable{
    private string $email;
    private function __construct(string $email) {
        $this->email = $email;
    }
    public static function create(string $email): self {
        return new self($email);
    }
    // public function getEmail(): string
    // public function withEmail(string $email): self
}
```

~~$user = new UserImmutable('user@ekino.com');~~

```php
$user = UserImmutable::create('user@ekino.com');
```

When should I convert an object to an immutable object?

~~When should I convert an object to an immutable object?~~
When should I convert an immutable object to a mutable object?

- Defensive programming!

When should I convert an immutable object to a mutable object?

Most of time, our code tries to “modelize” real life

In real life:

When a user “John Doe” changes his email

=> He is still “John Doe”

=> User must mutate

When you change the value of a price

=> It’s not the same price

=> Price must be immutable

Look at “Domain Driven Design”

End of my conference?

- The beginning of your curiosity!

ekino.

FORUMPHP PARIS2023

Thank you. **Merci.** 謝謝. dhany

Go further 🍍

- https://en.wikipedia.org/wiki/Immutable_object
- « Objects are passed by references by default » https://www.php.net/manual/en/language.oop5.references.php
- https://en.wikipedia.org/wiki/Side_effect_(computer_science)
- https://en.wikipedia.org/wiki/Value_object
- https://en.wikipedia.org/wiki/Defensive_programming
- https://en.wikipedia.org/wiki/Domain-driven_design
- My article https://medium.com/ekino-france/introduction-aux-objets-immutables-f7b7d0b00f36 [FRENCH]