Why and how to use generators for asynchronous programming

Benoit Viguier
@b_viguier

ConFoo.CA
DEVELOPER CONFERENCE

Why and how to use generators for asynchronous programming

Benoit Viguier
@b_viguier

Why and how to use generators for asynchronous programming

Benoit Viguier
@b_viguier

ConFoo.CA
DEVELOPER CONFERENCE

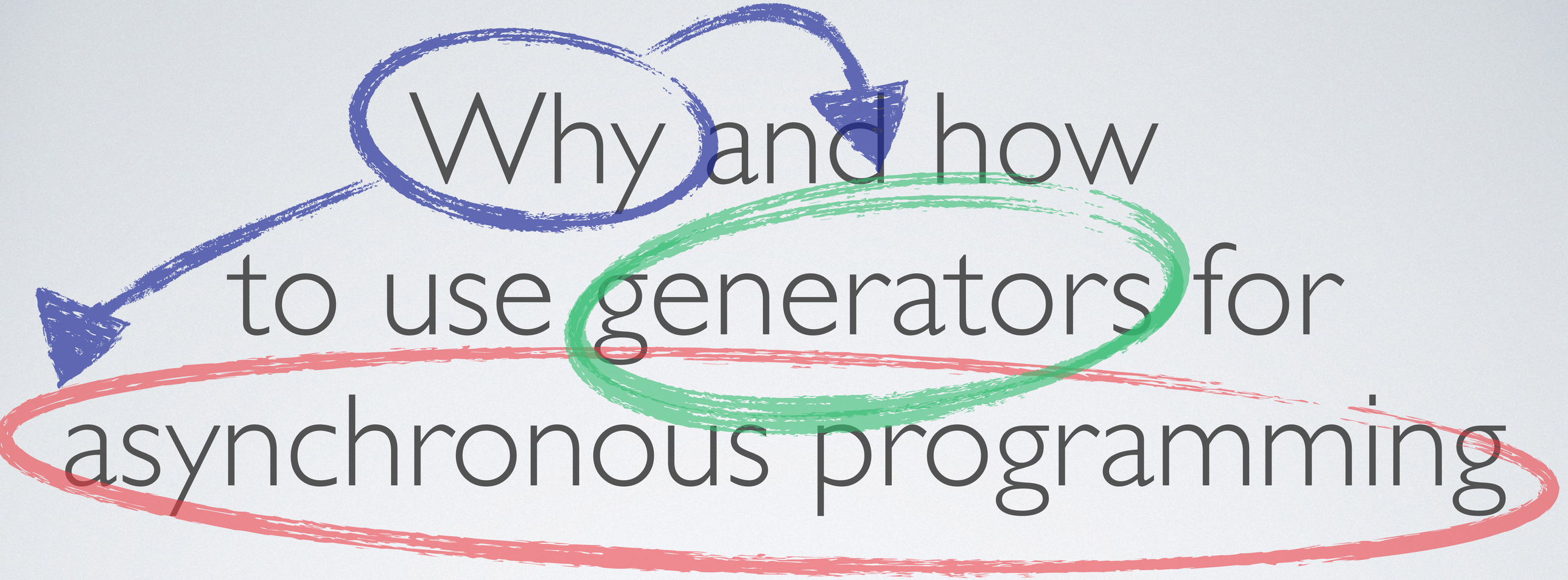
Why and how to use generators for asynchronous programming

BEDROCK

Benoit Viguier

@b_viguier

ConFoo.CA
DEVELOPER CONFERENCE

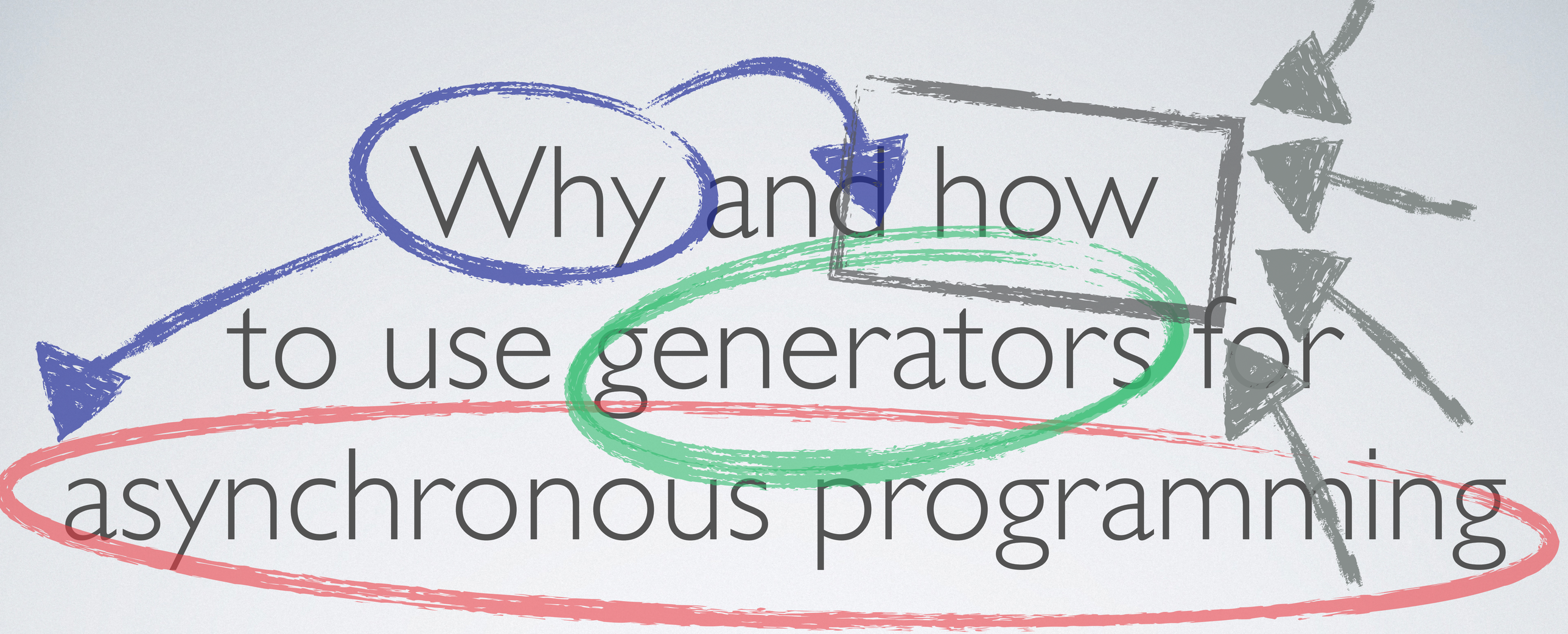


Why and how to use generators for asynchronous programming

BEDROCK

Benoit Viguier
@b_viguier

ConFoo.CA
DEVELOPER CONFERENCE



Why and how to use generators for asynchronous programming

BEDROCK

Benoit Viguier

@b_viguier

ConFoo.CA

DEVELOPER CONFERENCE

Why and how to use generators for asynchronous programming

BEDROCK

Benoit Viguier

@b_viguier

ConFoo.CA

DEVELOPER CONFERENCE


NETHERLANDS
IRELAND
Dublin
Amsterdam
Berlin
Warsaw
BELAR
London
GERMANY
POLAND
Celtic Sea
English Channel
Brussels
BELGIUM
LUX.
Prague
CZECH REP.
Paris
Luxembourg
SLOVAKIA
Bratislava
LIECH.
Vienna
Budapest
AUSTRIA
HUNGARY
FRANCE
Bern
SWITZERLAND
SLOVENIA
Chisin
ROMANIA
Ljubljana
Zagreb
CROATIA
Bay of Biscay
ITALY
Bucharest
BOS. & HER.
Belgrade
Sarajevo
SERBIA
SAN MARINO
BULGARIA
Pristina
MONT.
Sofia
KOS.
MONACO
Rome
Podgorica
Skopje
MACEDONIA
Tirana
ALB.
ANDORRA
VATICAN CITY
Madrid
PORTUGAL
SPAIN
Lisbon
Mediterranean Sea
GREECE
Athens

Lyon, France


BR
BEDROCK


BR
BEDROCK
EXIT
BACK/ RETURN
MEDIA
GUIDE
TIMER
TEXT
Panasonic
TV


GROUPE
RTL
GROUP

PRODUCTION ET ACQUISITIONS DE CONTENUS

TÉLÉVISION

- STUDIO 89 PRODUCTIONS
- C-PRODUCTIONS

DIGITAL

- GOLDEN NETWORK

CINÉMA

- SND GROUPE M6
- STUDIO
- FILMS
- SNC GROUPE M6

MÉDIAS

TÉLÉVISION EN CLAIR

- 6
- W9
- 6ter
- gulli

TÉLÉVISION PAYANTE

- PARIS PREMIERE
- téva
- serieclub
- Music
- Tiji
- canal J
- MCM TOP
- MCM
- RFM TV

TÉLÉVISION INTERNATIONALE

- gulli GIRL
- gulli بالعربي
- gulli AFRICA
- Tiji RUSSIA

DIGITAL

- gulli Replay
- 6play
- gulli MAX

RADIO

- RTL
- RTL2 LE SON POP-ROCK
- fun radio LE SON DANCEFLOOR

DIVERSIFICATIONS

- INTERACTIONS
- EDITIONS
- EVENEMENTS
- MUSIC LABEL
- gulli Parc
- Ventadis — Vente à distance - Groupe M6
- M6 boutique
- BEST of TV
- Leonis
- JOJIKA
- CREATIONS
- DIGITAL SERVICES
- Radins.com
- iGraal
- odiso
- mindbaz — your emailing service
- MINDTARGET
- TURBO
- M6météo
- passeport santé
- FOURCHETTE & BIKINI
- cuisineaz
- CROQ'KILOS
- BOTANIQUE
- déco.fr
- Panorabanques.com

RÉGIE

- CTZAR {za:(r)}
- PUBLICITÉ
- UNLIMITED

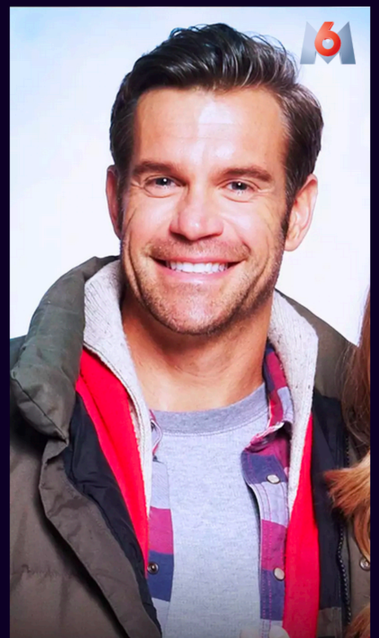

PRODUCTION ET ACQUISITIONS DE CONTENUS
TÉLÉVISION
STUDIO 89 PRODUCTIONS
C-PRODUCTIONS
DIGITAL
GOLDEN NETWORK
CINÉMA
SND GROUPE M6
STUDIO
FILMS
SNC GROUPE M6
MÉDIAS
TÉLÉVISION EN CLAIR
6
W9
6ter
gulli
TÉLÉVISION PAYANTE
PARIS PREMIÈRE
téva
serieclub
Music
TiJi
canal J
MCM TOP
MCM
RFM TV
TÉLÉVISION INTERNATIONALE
gulli GIRL
gulli بالعربي
gulli AFRICA
TiJi RUSSIA
DIGITAL
gulli Replay
6play
gulli MAX
RADIO
RTL
RTL2
LE SON POP-ROCK
fun radio
LE SON DANCEFLOOR
DIVERSIFICATIONS
INTERACTIONS
EDITIONS
EVENEMENTS
MUSIC LABEL
gulli Parc
Ventadis
Vente à distance - Groupe M6
BEST of TV
Leonis
JOJIKA
CREATIONS
DIGITAL SERVICES
Radins.com
iGraal
odiso
mindbaz
MINDTARGET
TURBO
M6météo
passeport santé
FOURCHETTE & BIKINI
cuisineaz
CROQ'KILOS
BOTANIQUE
déco.fr
Panorabanques
RÉGIE
CTZAR
PUBLICITÉ
UNLIMITED

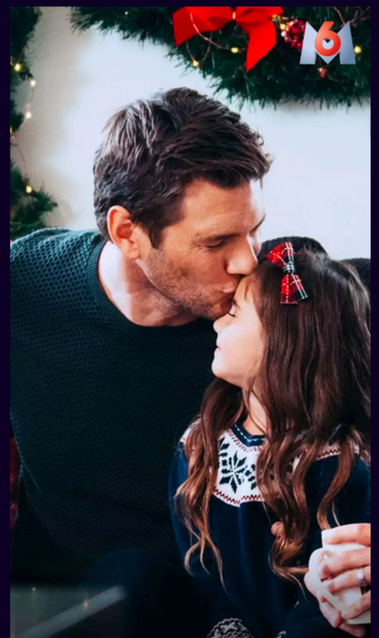

6play
M6
W9
6ter
funradio
etc.
RTL play
À VOUS DE VOIR
RTL TVI
club RTL
PLUG RTL
etc.
RTL most
RTL KLUB
RTL
etc.
RTL PLAY
RTL
2 RTL
Kockica
DIGITAL CHAPERS
etc.

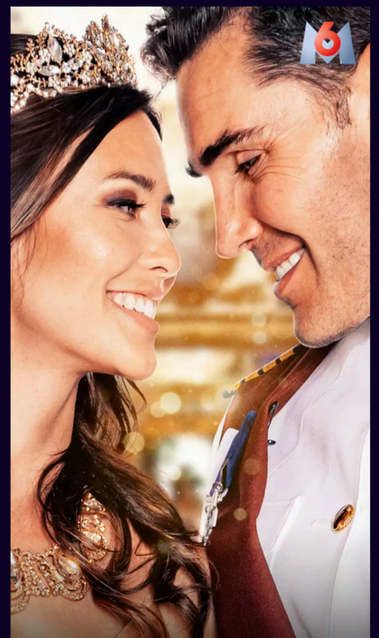

10 Million unique
users per month

Programmes par genre

Le Groupe M6 vous souhaite une bonne année !







One year of asynchronous PHP in production

ConFoo.CA
DEVELOPER CONFERENCE

Tomorrow, 13:00

Asynchronous Programming

BEDROCK

Benoit Viguier

@b_viguier

ConFoo.CA

DEVELOPER CONFERENCE

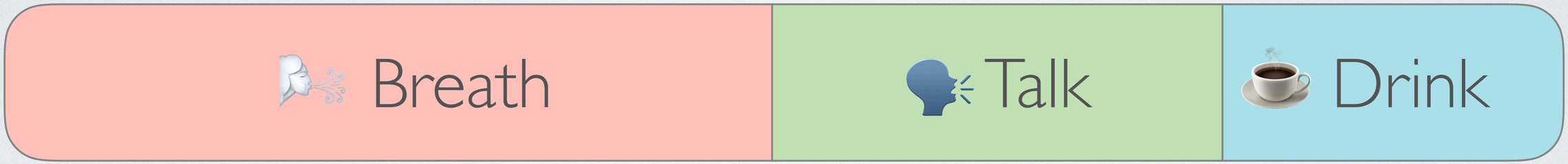
Tea for Two

An illustrated example

Tasks

-  **Drink** Tea
-  **Talk** with a friend
-  **Breath** just… to live

Synchronous

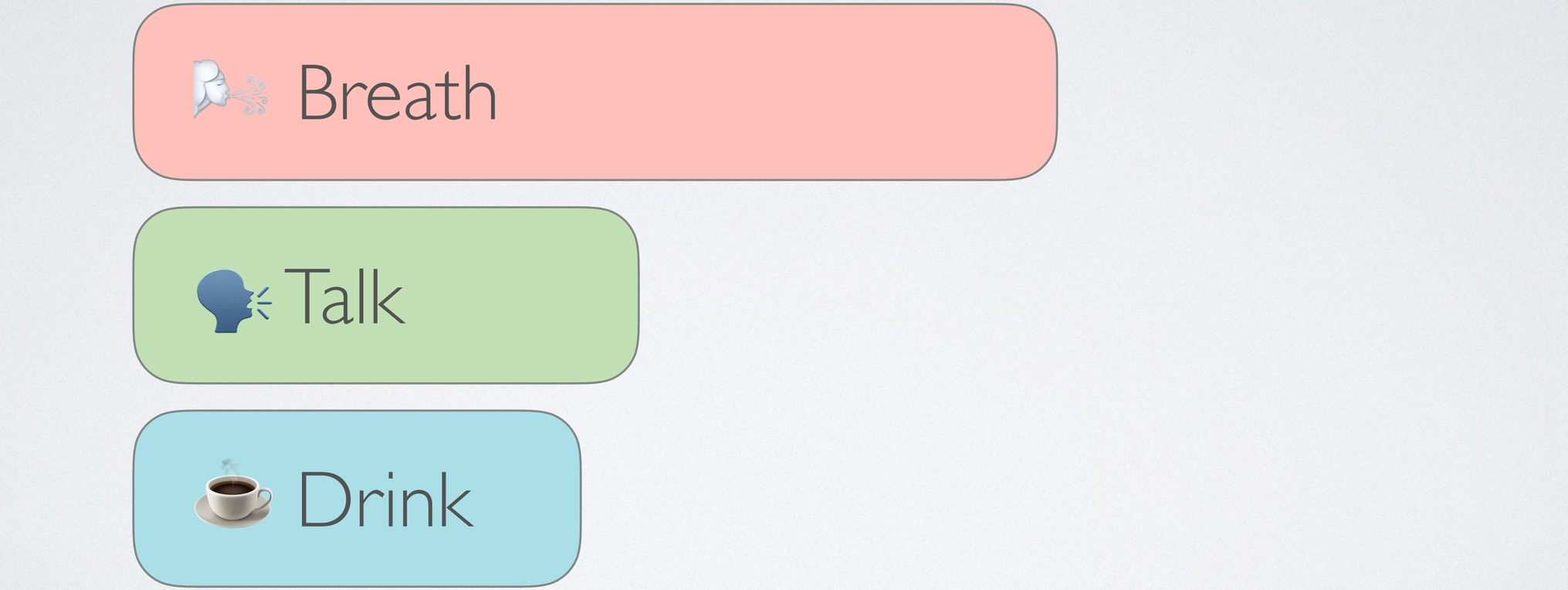


Time

Concurrency

Starting next task before the current one ends

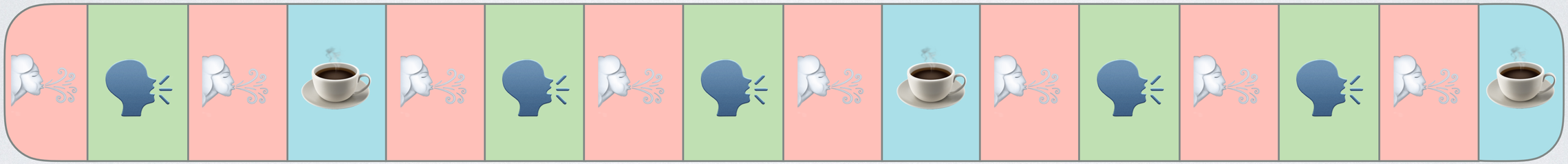
Parallel



Most languages are single threaded

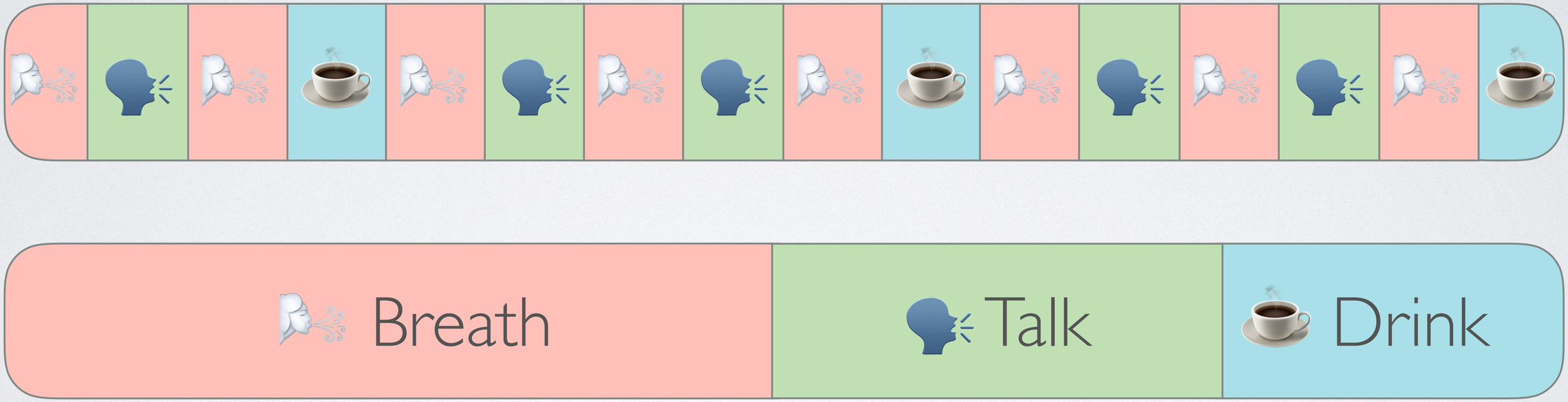
… by default

Asynchronous



Time

But… 🤔



Analyzing tasks

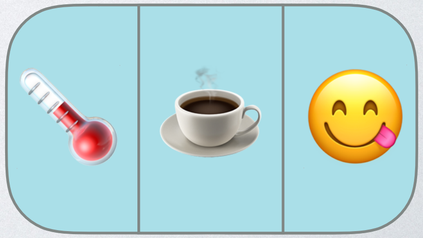
Internal/External operations

Drinking Tea

Waiting for the right temperature

Drinking (actually)

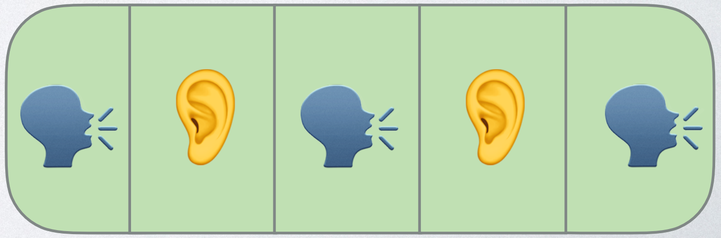
Enjoying



🗣️ **Talking** with a friend

🗣️ Speaking, asking something

👂 Listening

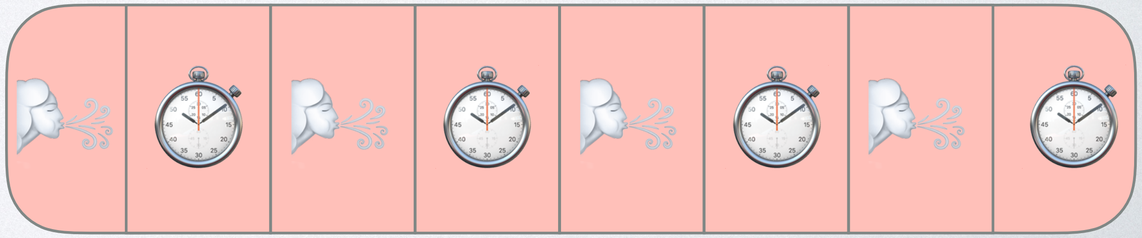


Breathing

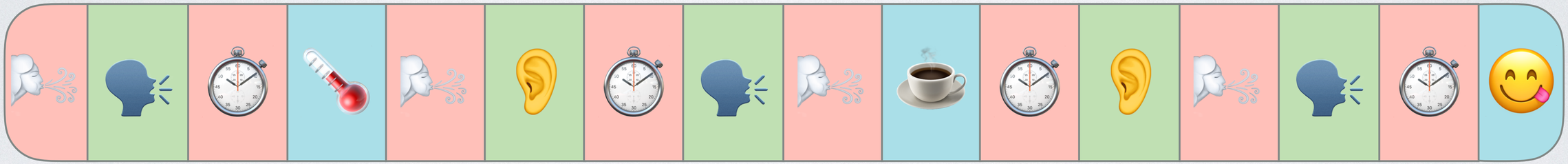
Breathing…

Waiting (a little)

Breathing…

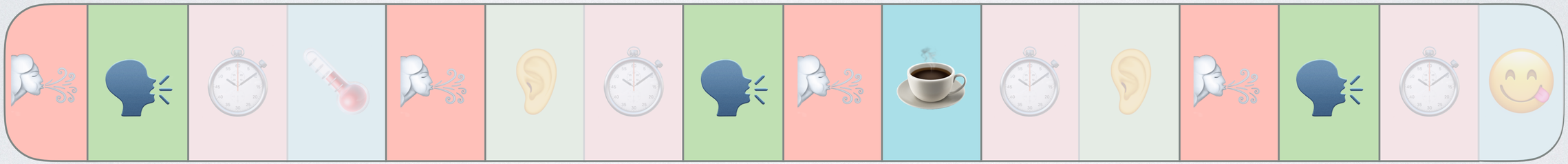


Asynchronous



Time

Internal operations



Time

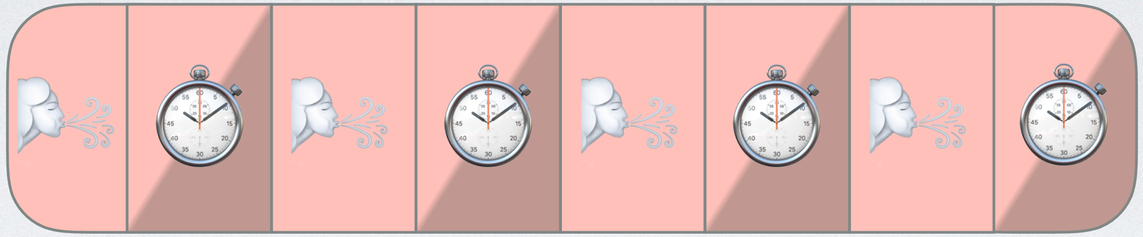
External operations

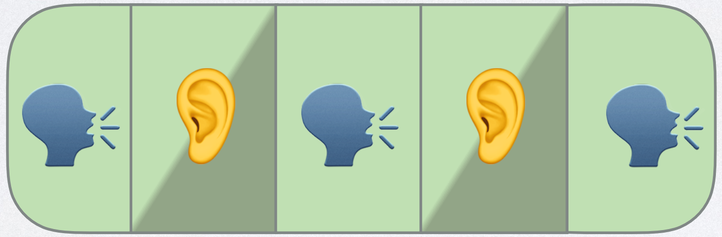
Time

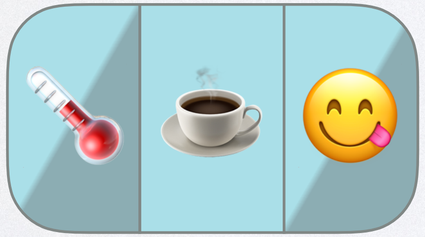
Dispatching execution

Event Loop

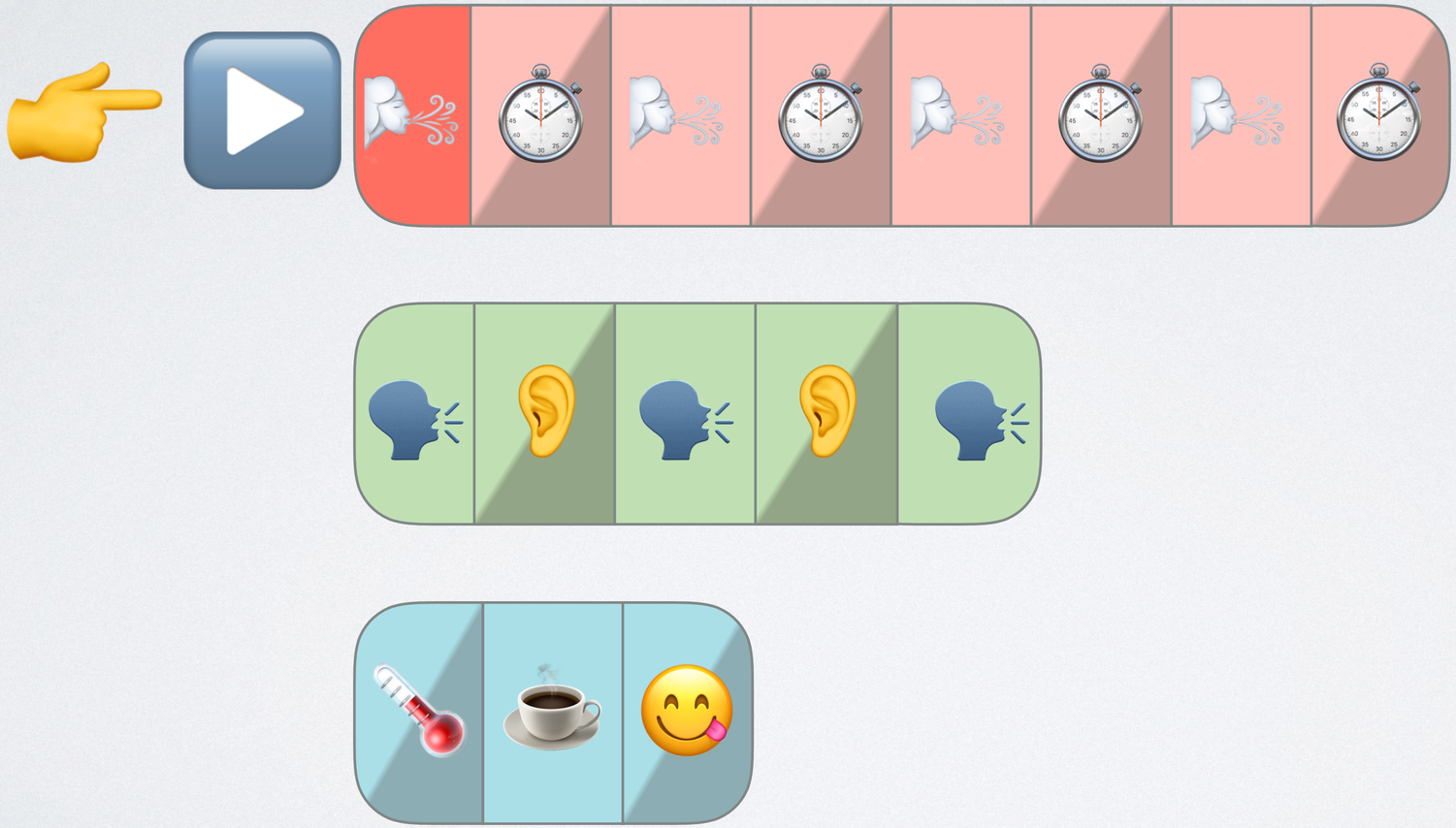
Event Loop



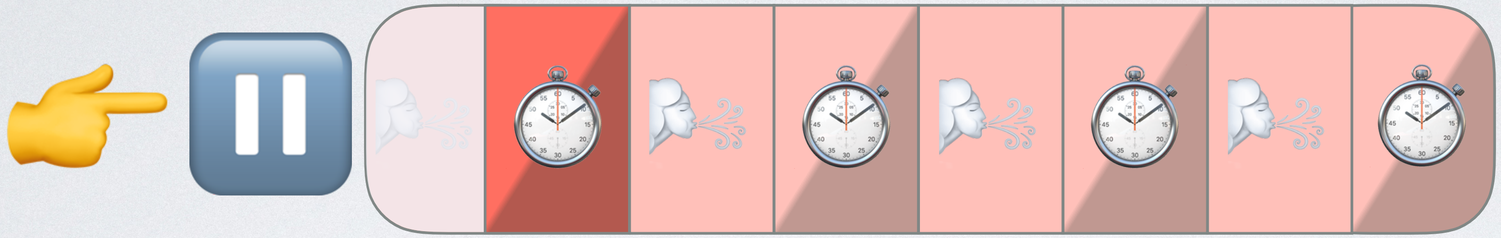


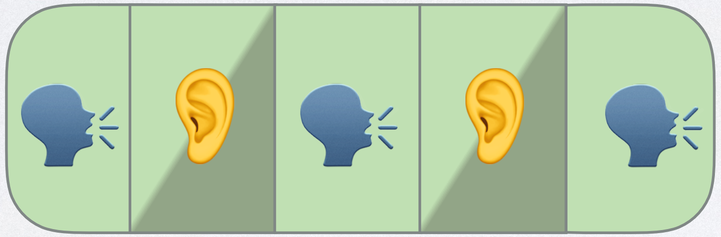


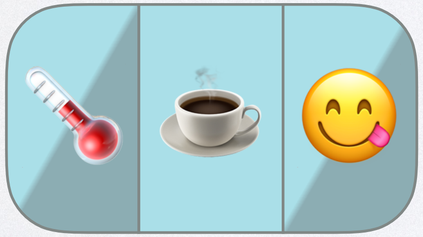
Event Loop



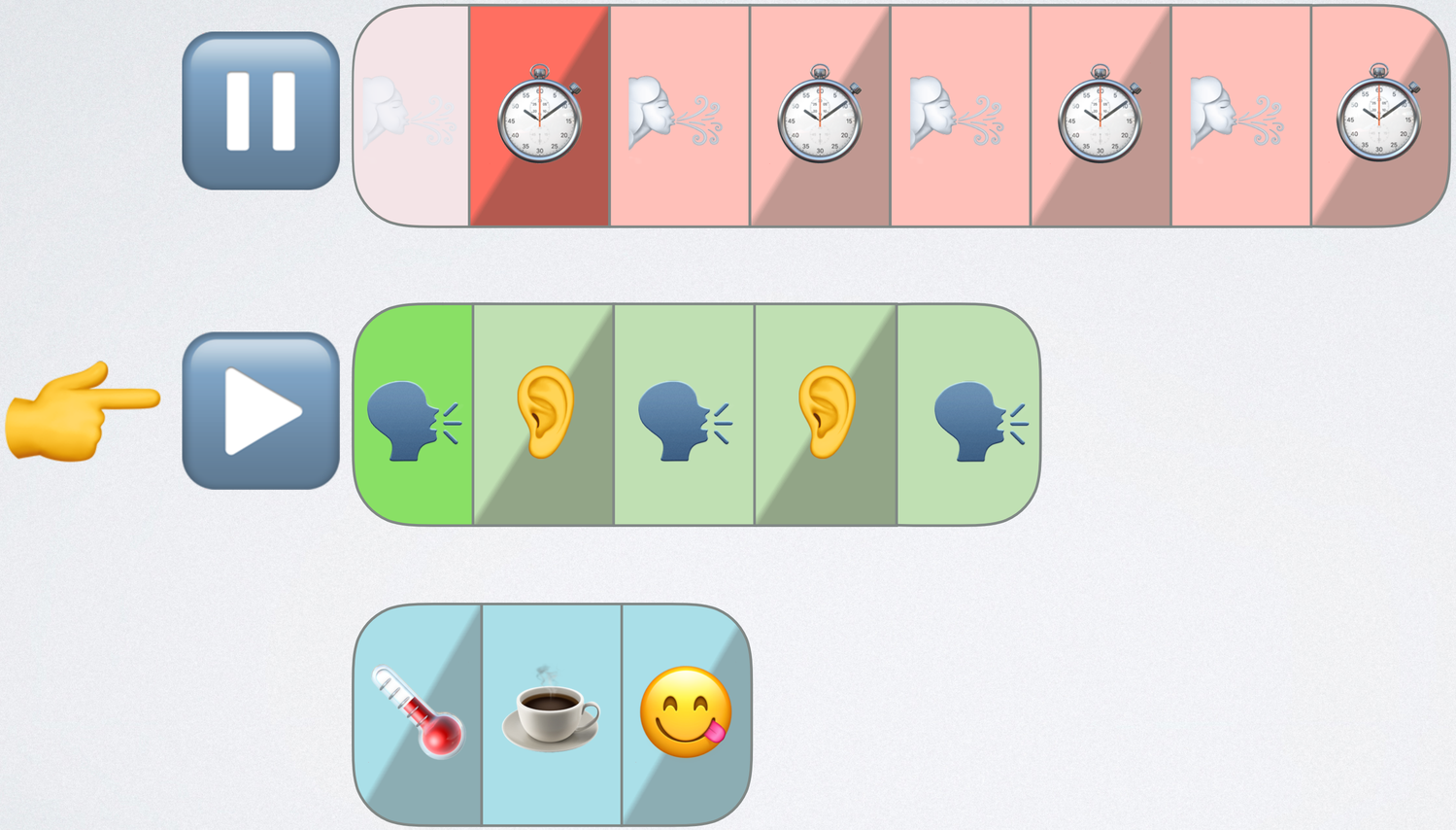
Event Loop







Event Loop

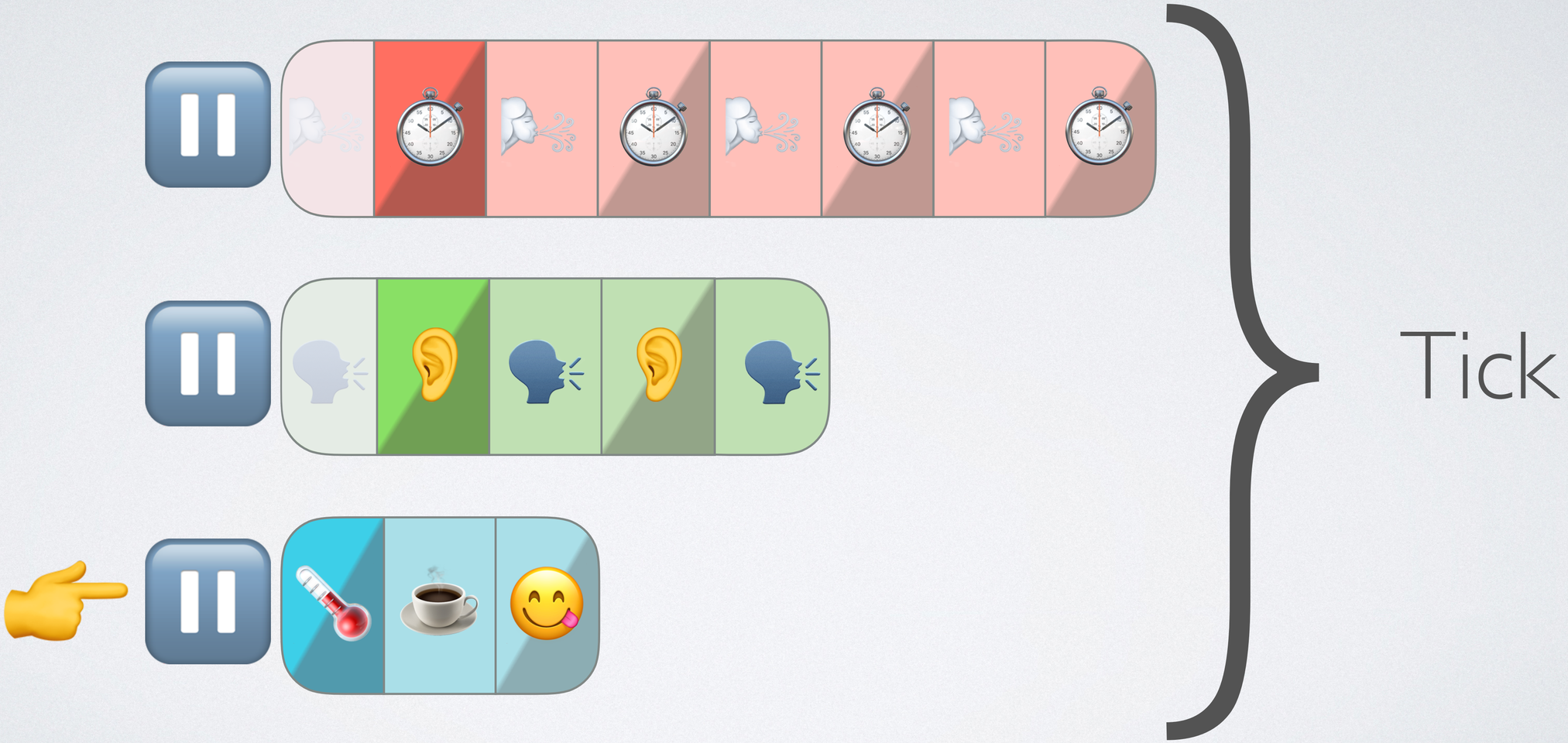


Event Loop

Event Loop

Event Loop

Event Loop



Event **Loop**

Event Loop

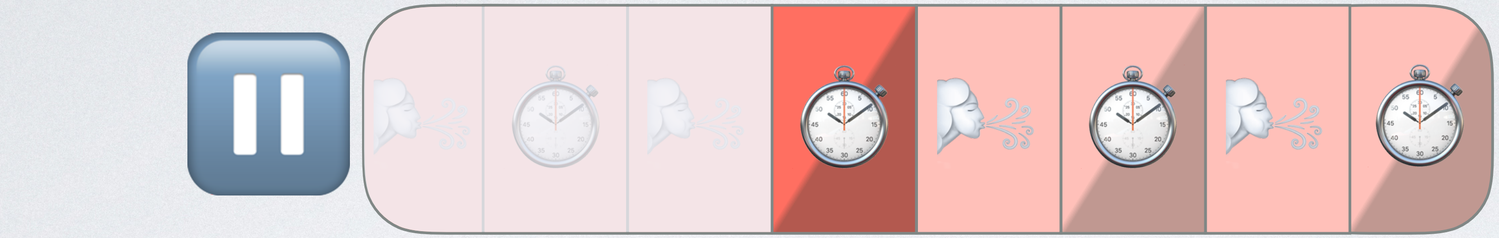
Event Loop

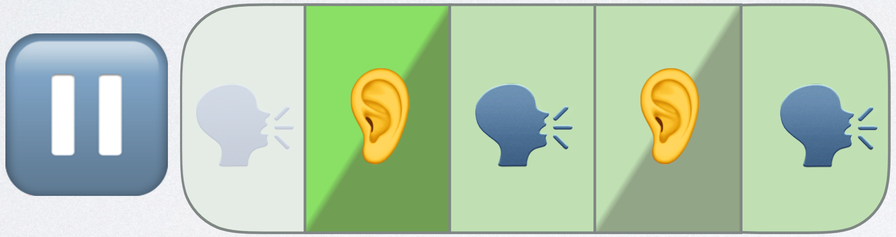
Event Loop

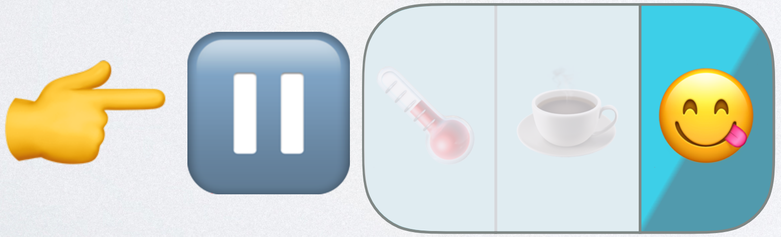
Event Loop

Event Loop

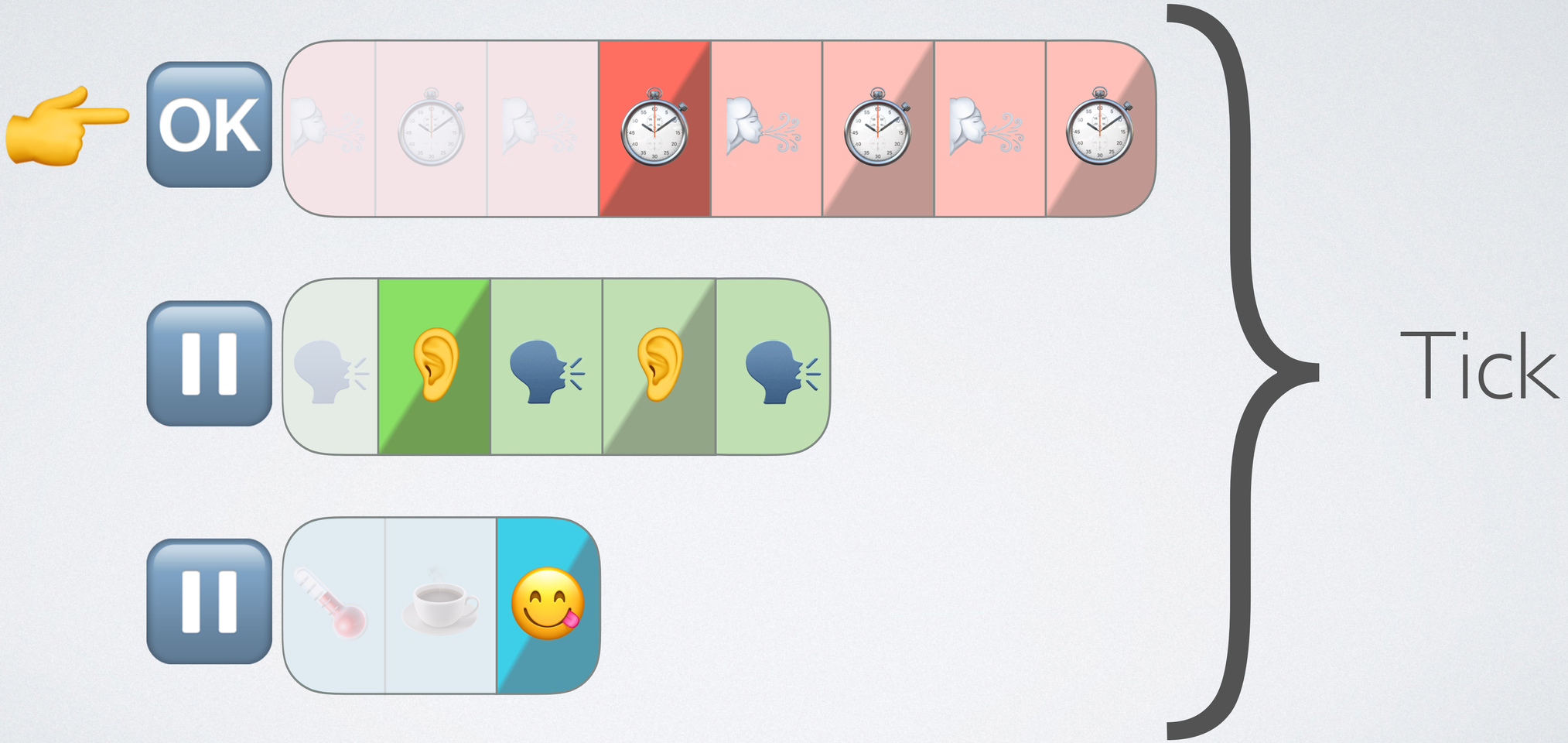
Event Loop





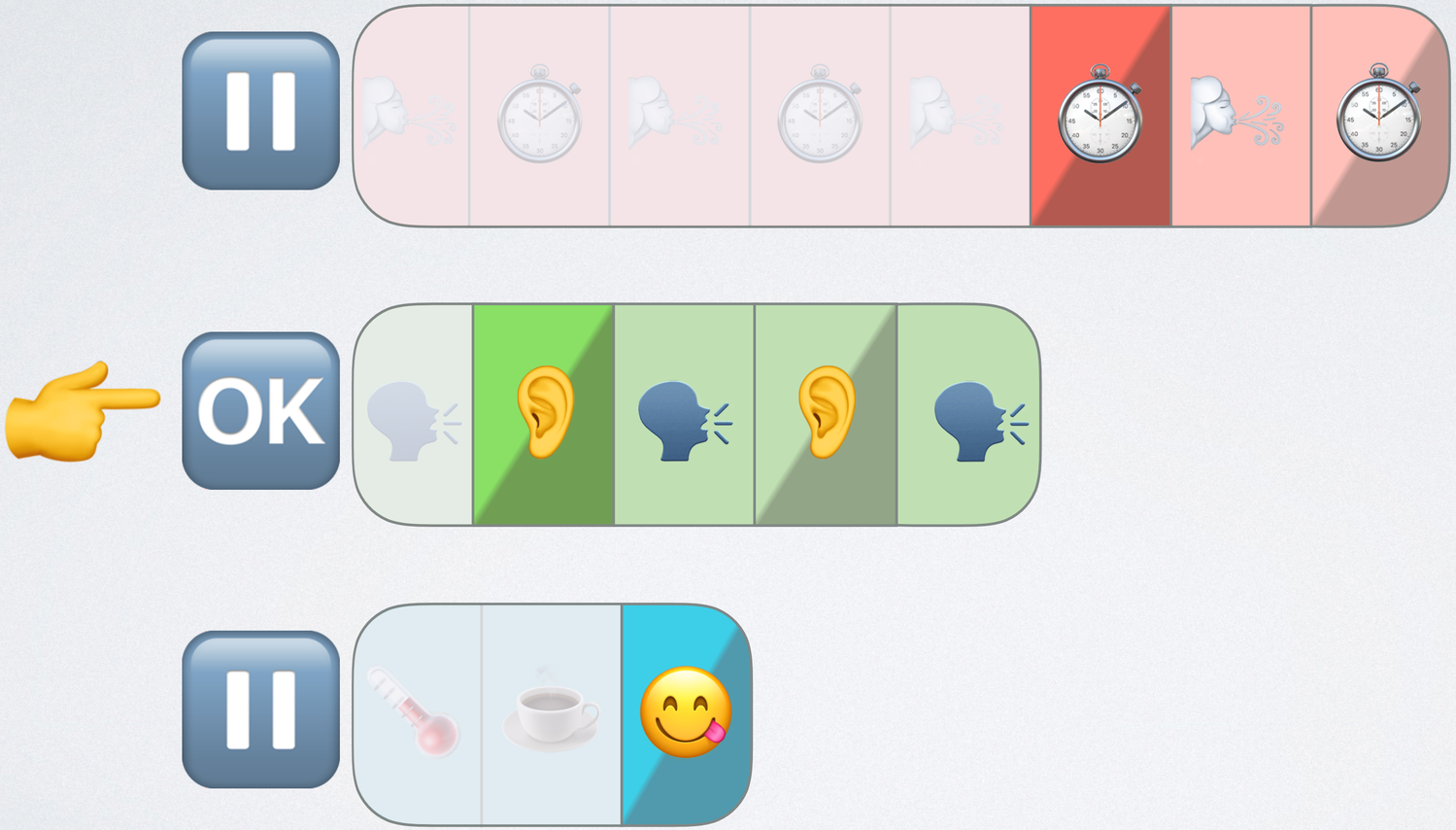


Event Loop



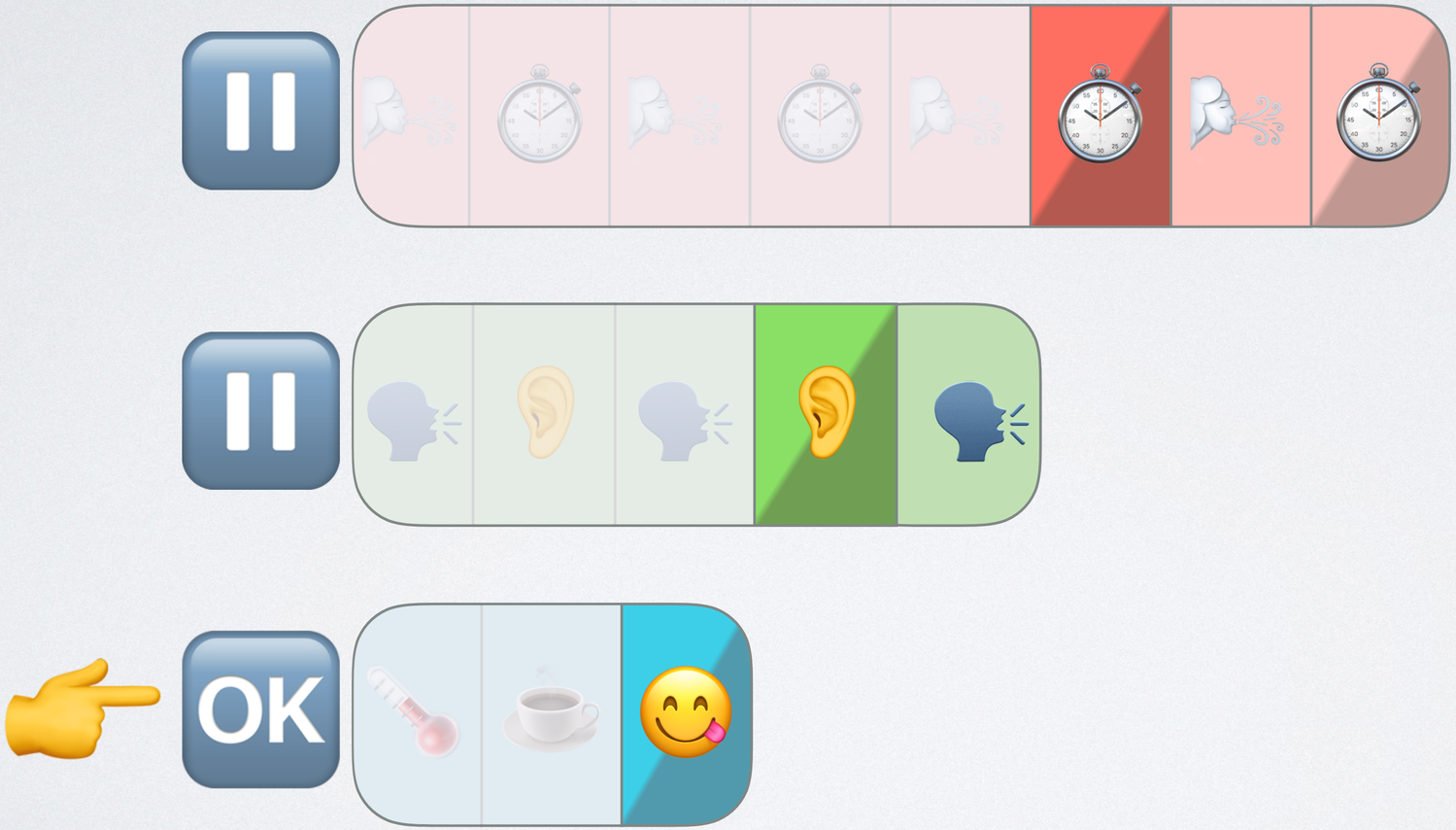
Event Loop

Event Loop



Event Loop

Event Loop



Event Loop

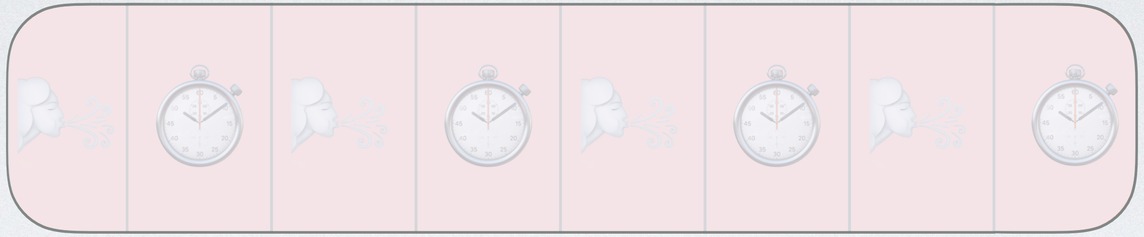
Event Loop

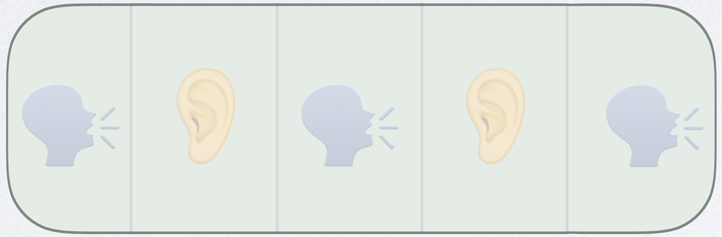
Event Loop

Event Loop

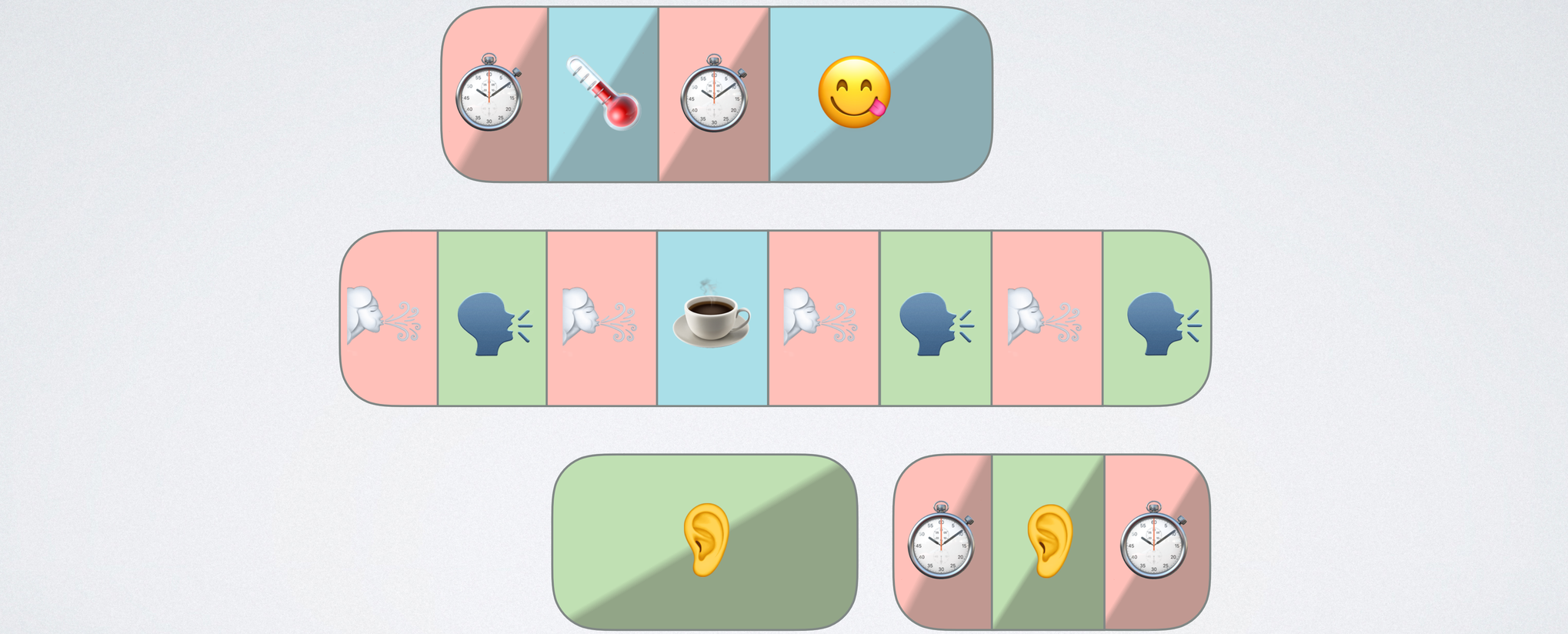
Event Loop

Event Loop



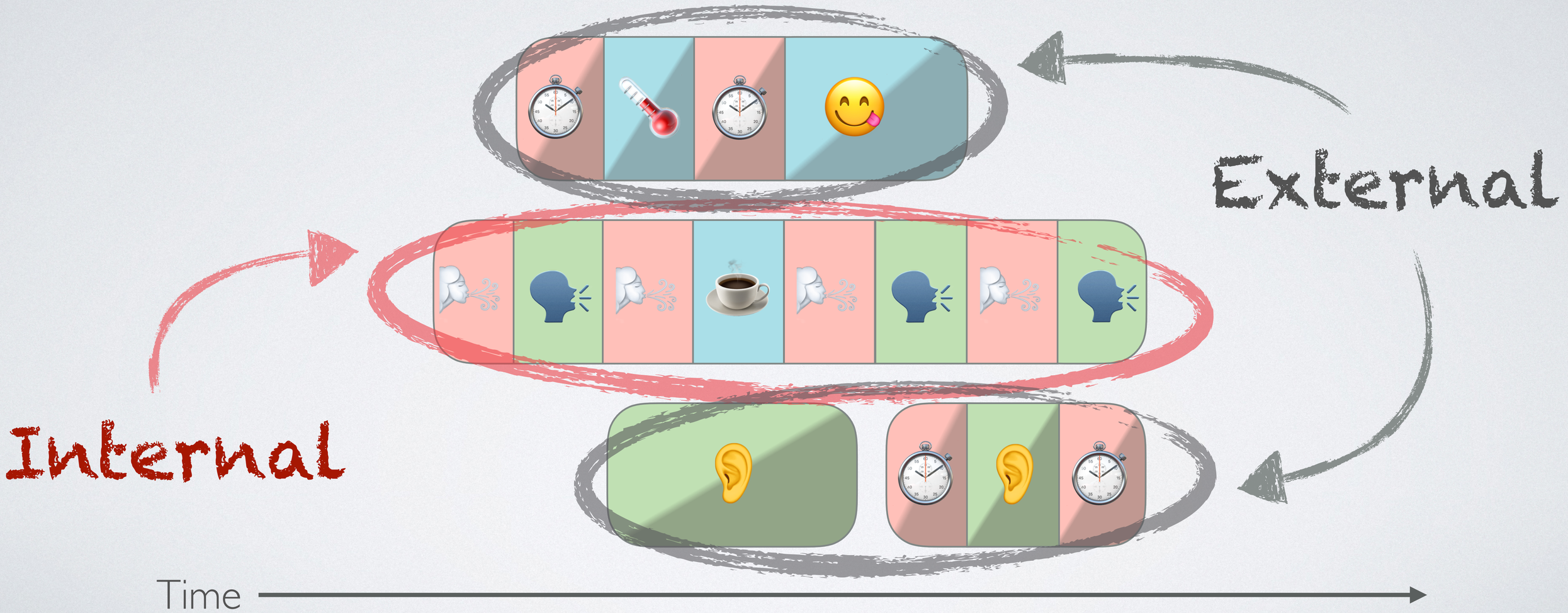


Asynchronous

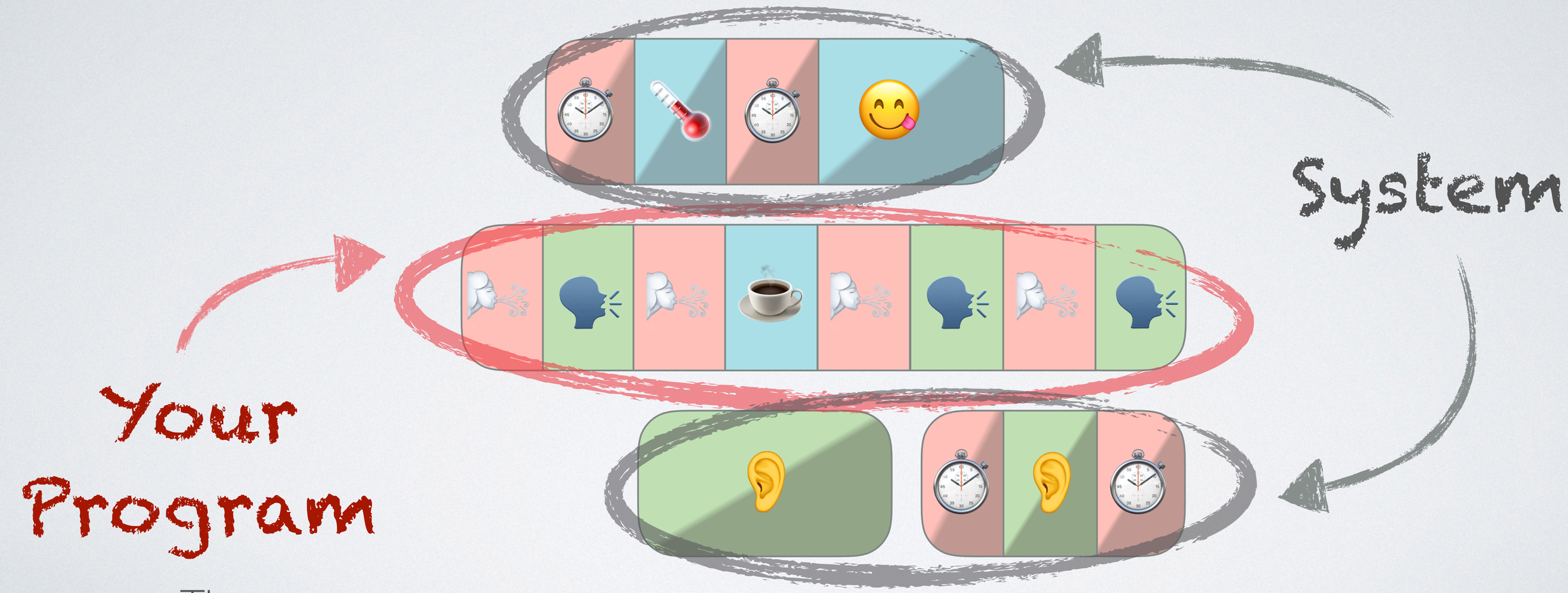


Time

Asynchronous



Asynchronous



How to dispatch code execution efficiently?

Generators

Subset of coroutines

BEDROCK

Benoit Viguier

@b_viguier

ConFoo.CA

DEVELOPER CONFERENCE


PARENTAL
ADVISORY
EXPLICIT PHP

Example

```php
function gen_one_two_three() {
    for ($i = 1; $i <= 3; $i++) {
        yield $i;
    }
}

$generator = gen_one_two_three();
foreach ($generator as $value) {
    echo "$value\n";
}
```

Starting with Generators

… by the weird part…

Creation

```
// ❌ Does NOT work
$generator = new \Generator;

// PHP Fatal error:  Uncaught Error:
// The "Generator" class is reserved
// for internal use
// and cannot be manually instantiated
```

Creation

```
function create(): \Generator
{
    yield;
}

$generator1 = create();
$generator2 = create();
```

Only `yield` matters

```php
function emptyGenerator(): \Generator
{
    return 1;
    yield; // Never reached…
}

$generator = emptyGenerator();
```

Execution

```php
function dyingGenerator(): \Generator
{
    die('hard');
    yield;
}
// The function is not executed…
$generator = dyingGenerator();
// Will die ‘hard’
$generator->valid();
```

```
final class Generator implements Iterator {

    function rewind() {}
    function valid(): bool {}
    function current() {}
    function key() {}
    function next() {}

    function send($value) {}
    function throw(Throwable $exception) {}
    function getReturn() {}
}
```

Outside

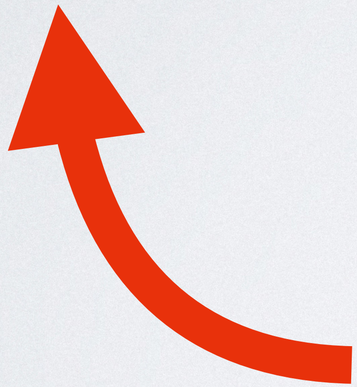
```php
$generator = createMyGenerator();
// Here, outside content
```

```php
function createMyGenerator():
\Generator
{
    // Here, inside content
}
```

Inside

Outside

```
$value = $generator->current();
$key = $generator->key();
```

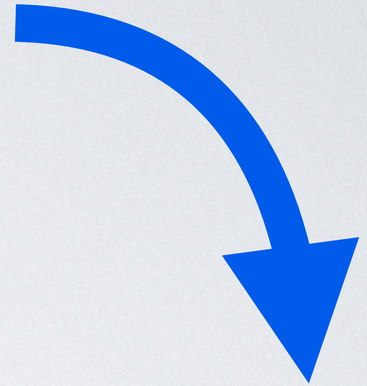


```
yield $key => $value;
```

Inside

Outside

```
$generator->send($value);
$generator->next();
```

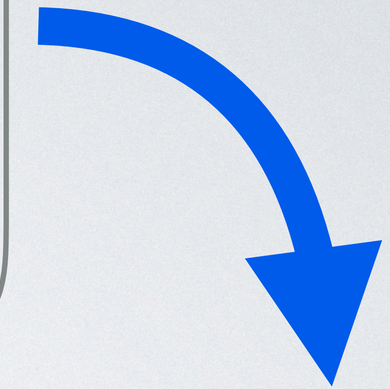


```
$value = yield;
$nullValue = yield;
```

Inside

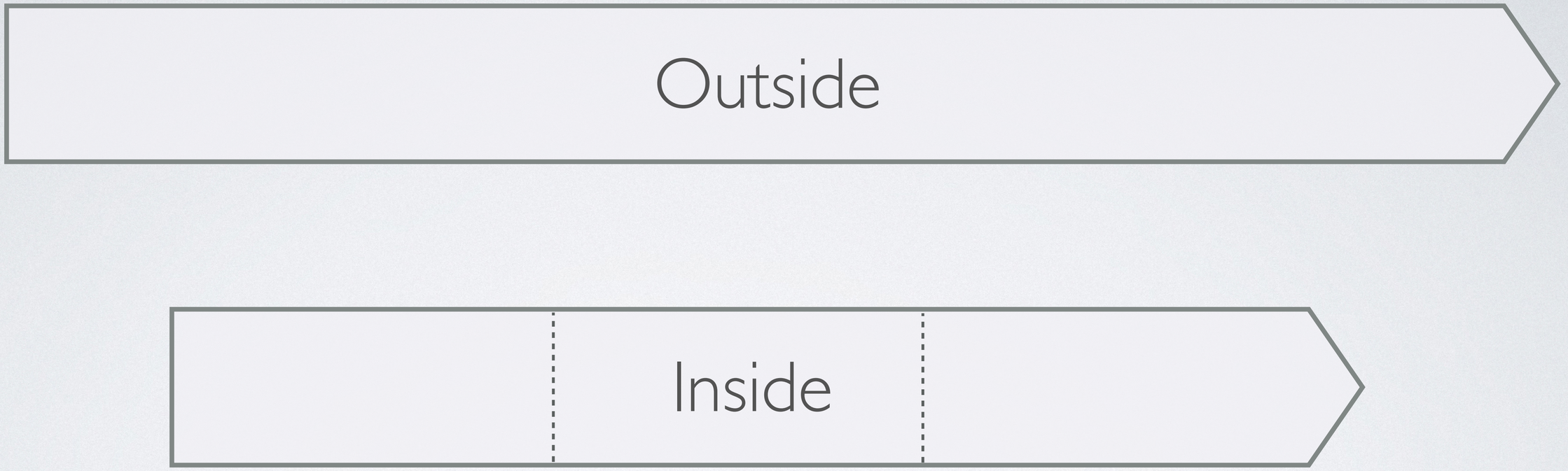
Outside

```
$v1 = $generator->current();
$v3 = $generator->send($v2);
```

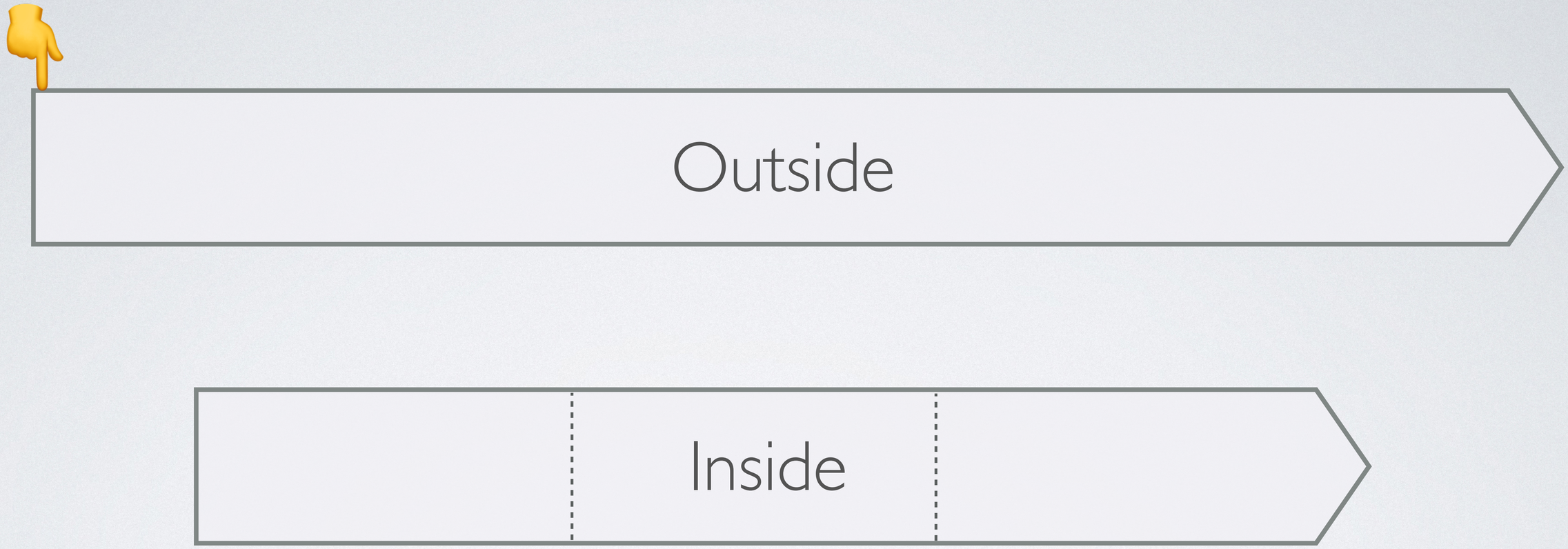


```
$v2 = yield $v1;
yield $v3;
```

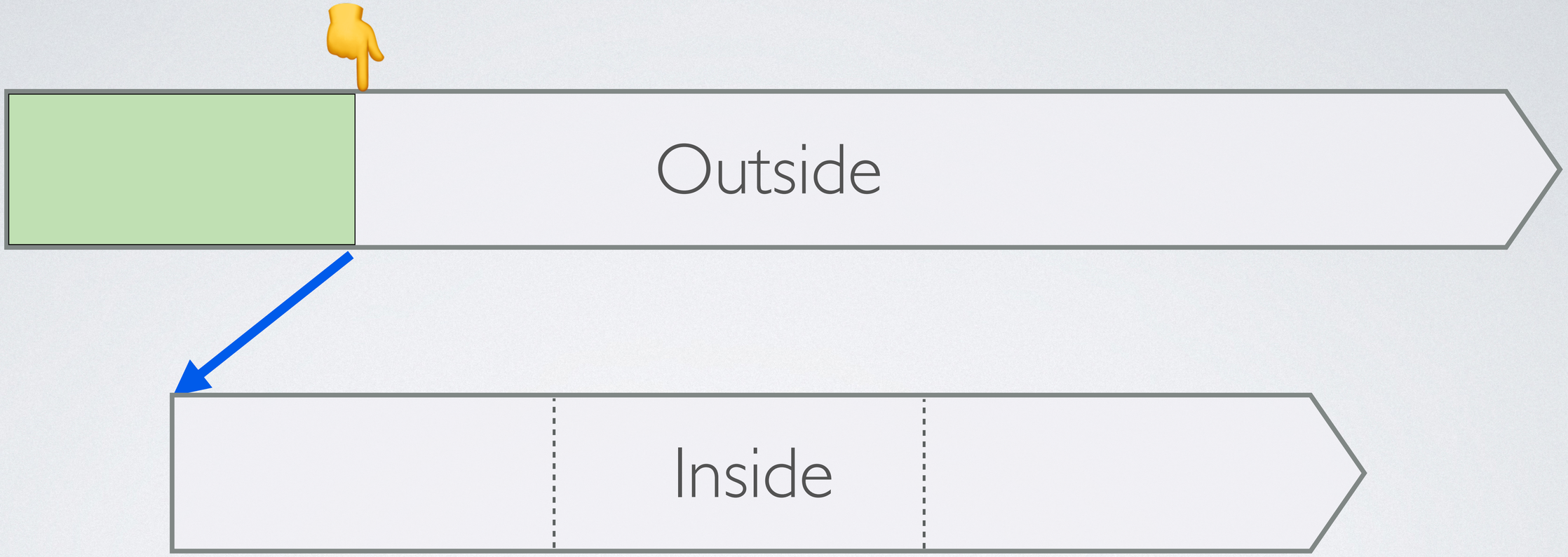
Inside


Outside
Inside

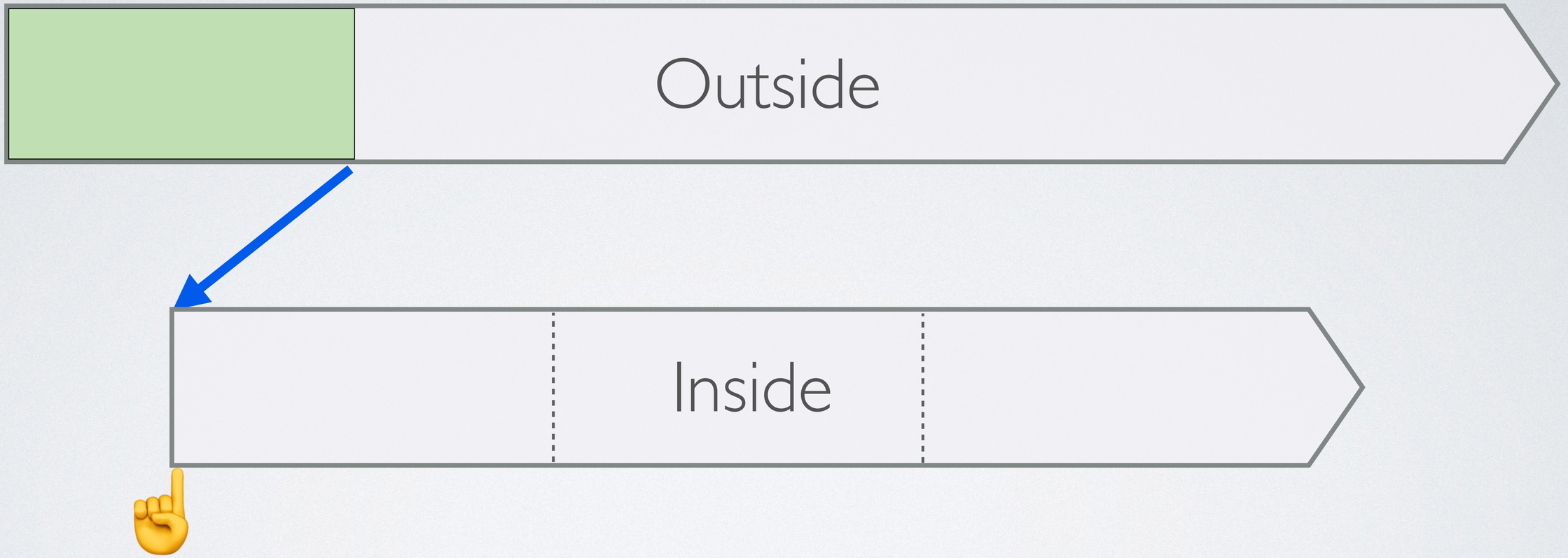
```
$generator = createMyGenerator()
```



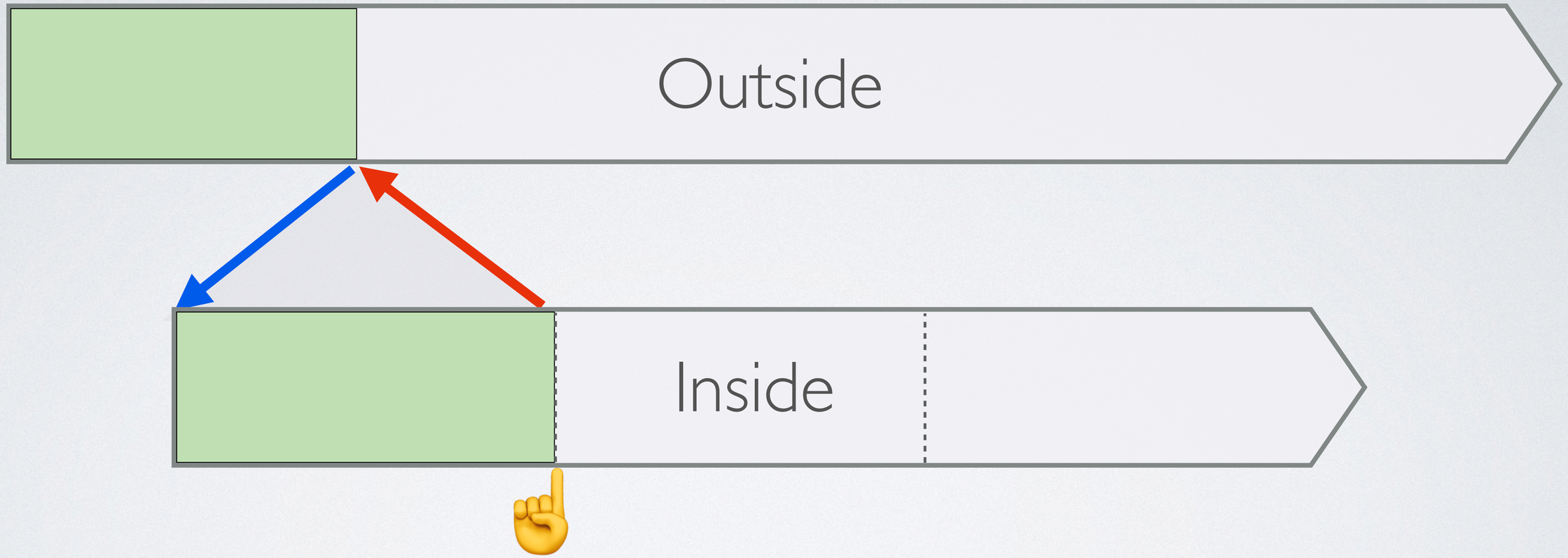
```
$v1 = $generator->current()
```



```
$v1 = $generator->current()
```

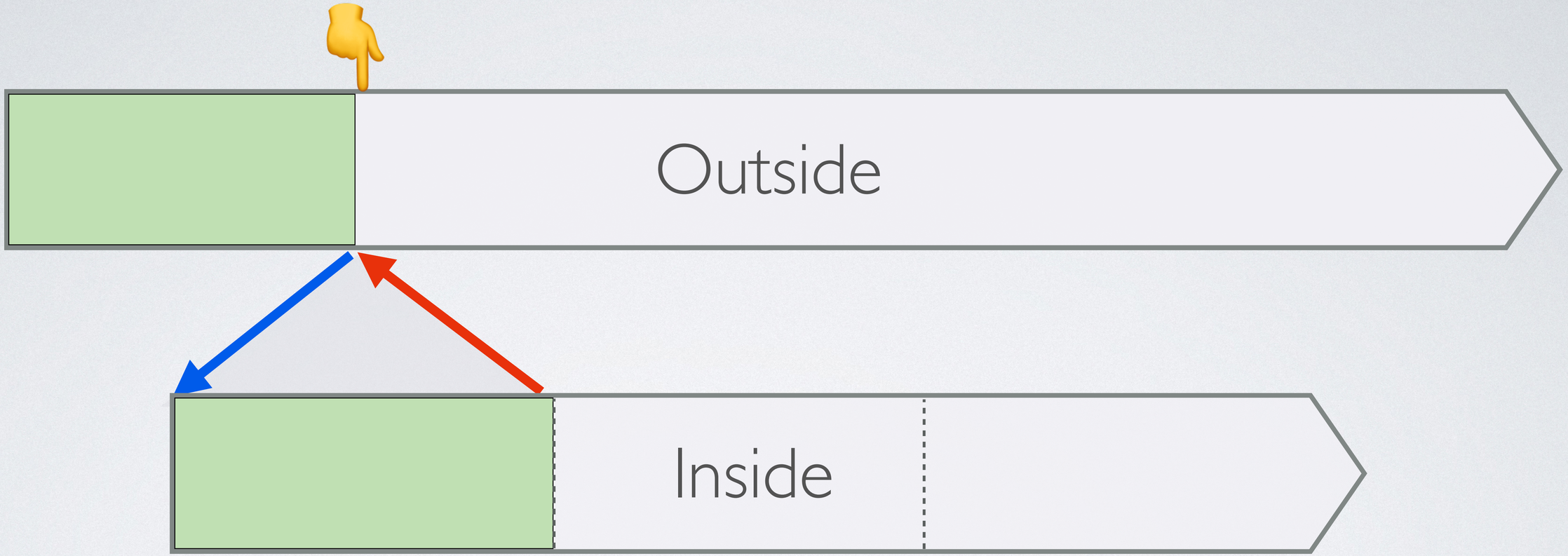


```
$v1 = $generator->current()
```



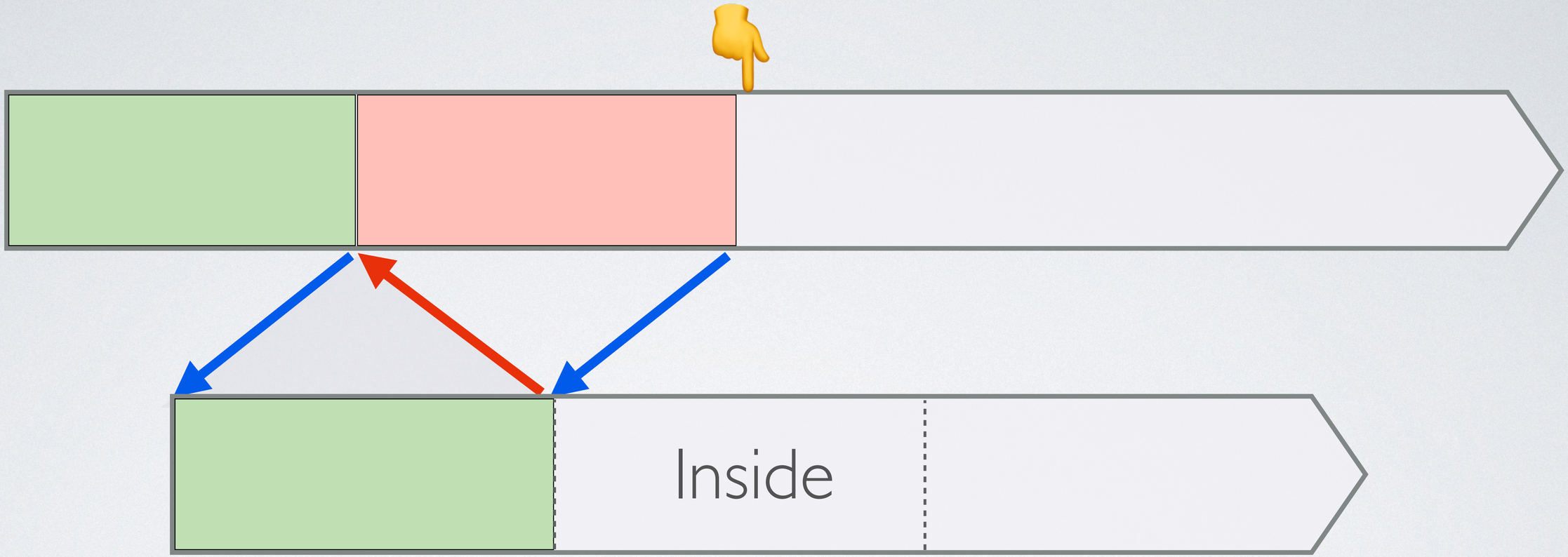

```
$v2 = yield $v1
```

```
$v1 = $generator->current()
```



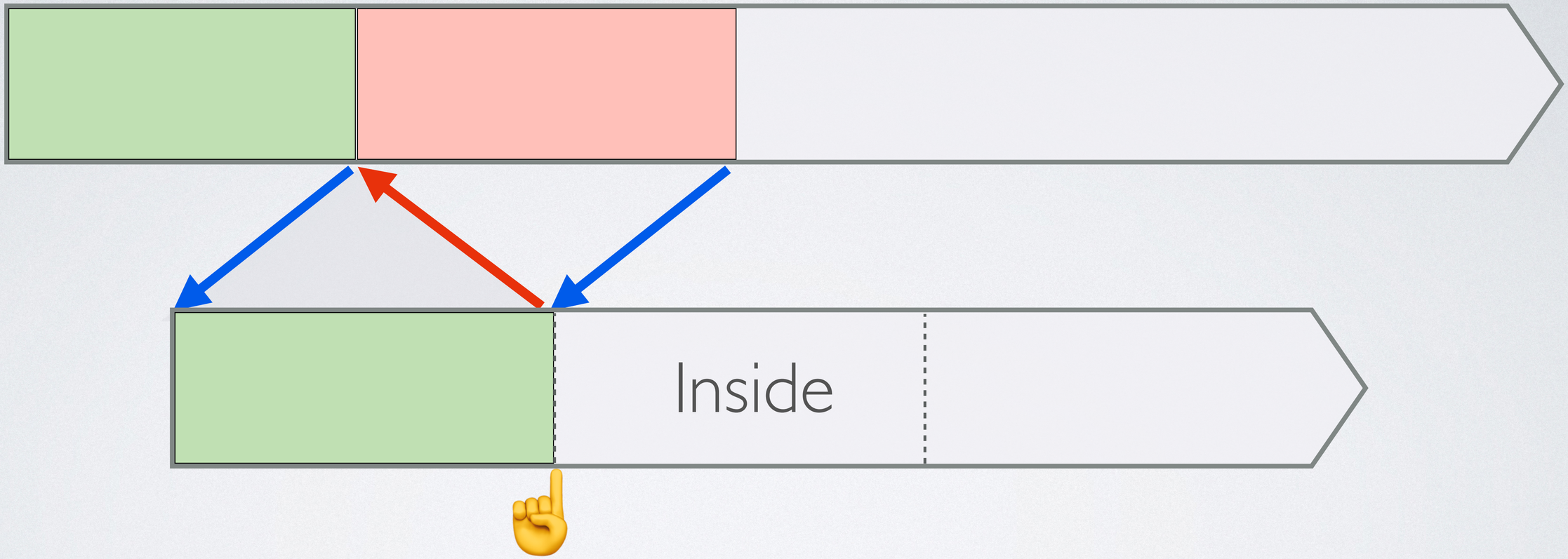

```
$v2 = yield $v1
```

```
$v3 = $generator->send($v2)
```



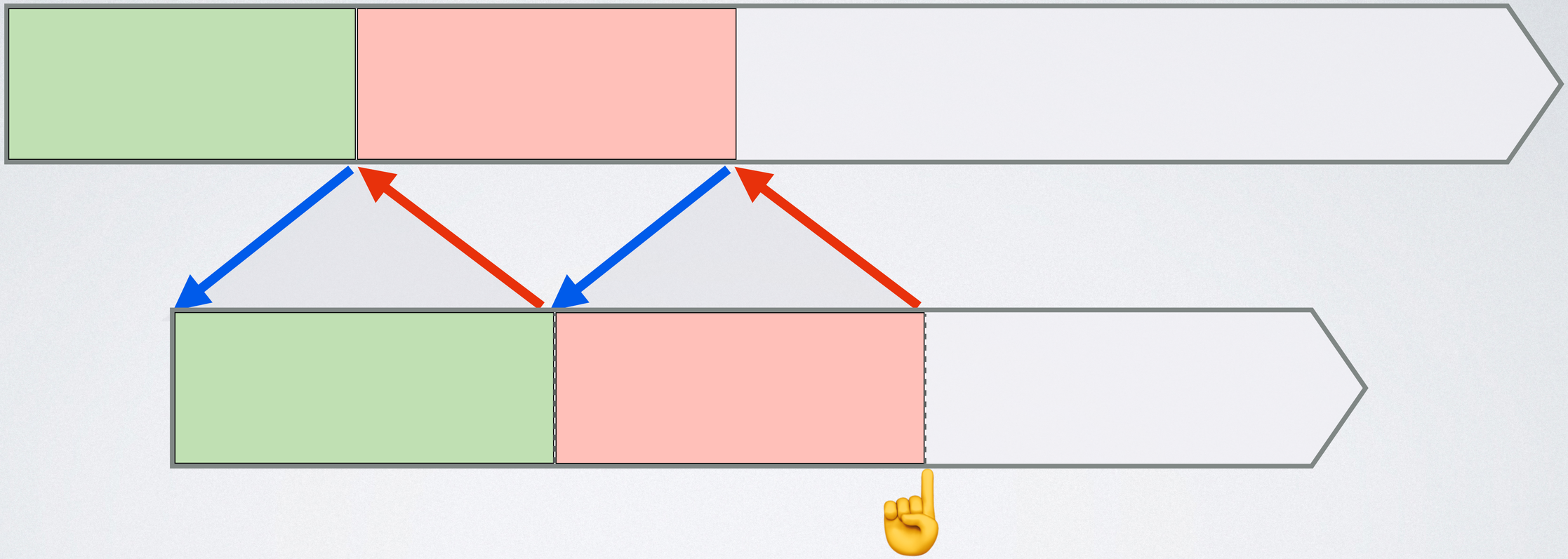

```
$v2 = yield $v1
```

```
$v3 = $generator->send($v2)
```



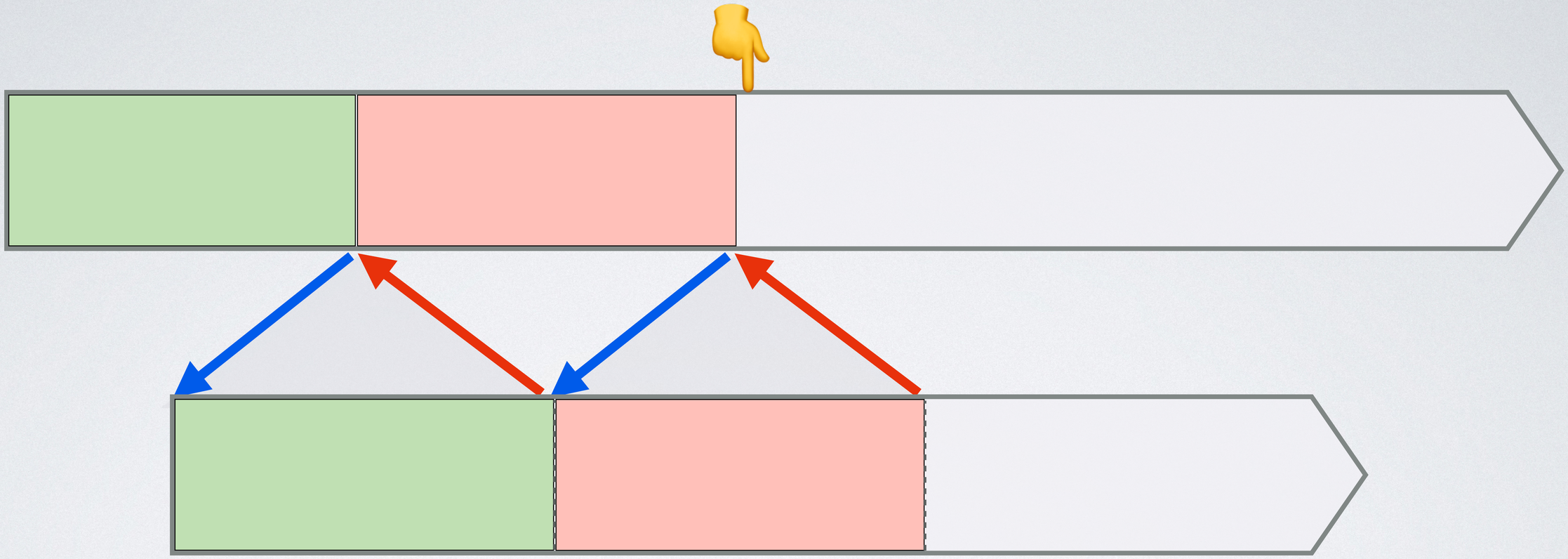

```
$v2 = yield $v1
```

```
$v3 = $generator->send($v2)
```



```
yield $v3
```

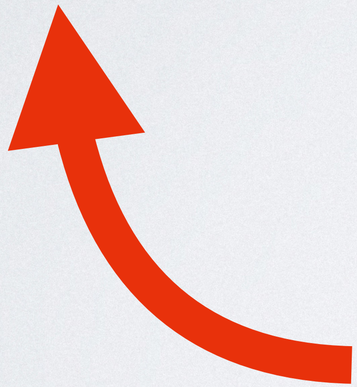
```
$v3 = $generator->send($v2)
```



```
yield $v3
```

Outside

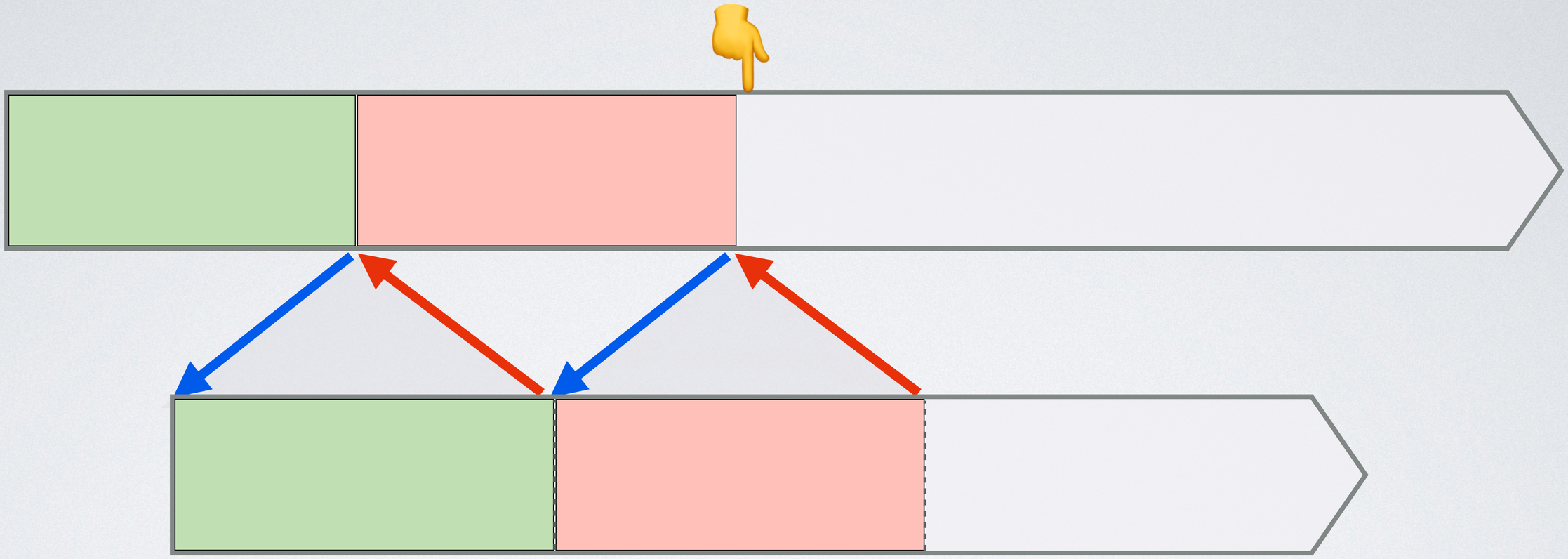
```php
if (!$generator->valid()) {
    $value = $generator->getReturn();
}
```



```php
return $value;
```

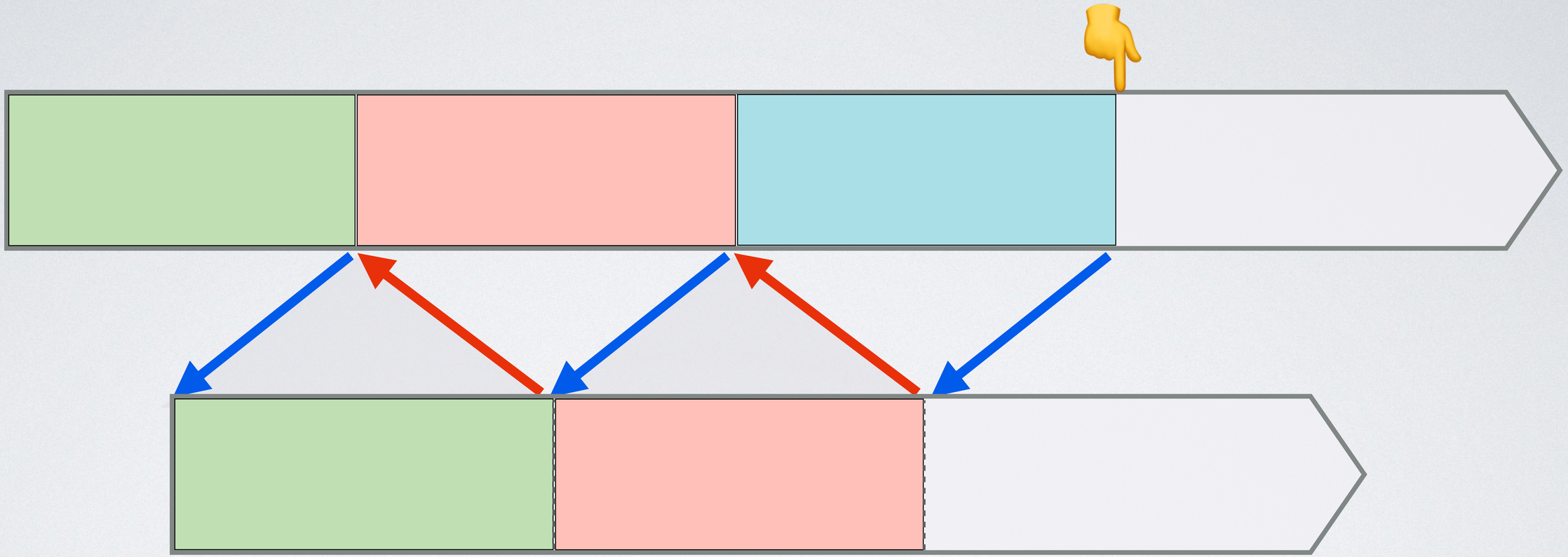
Inside

```
$v3 = $generator->send($v2)
```



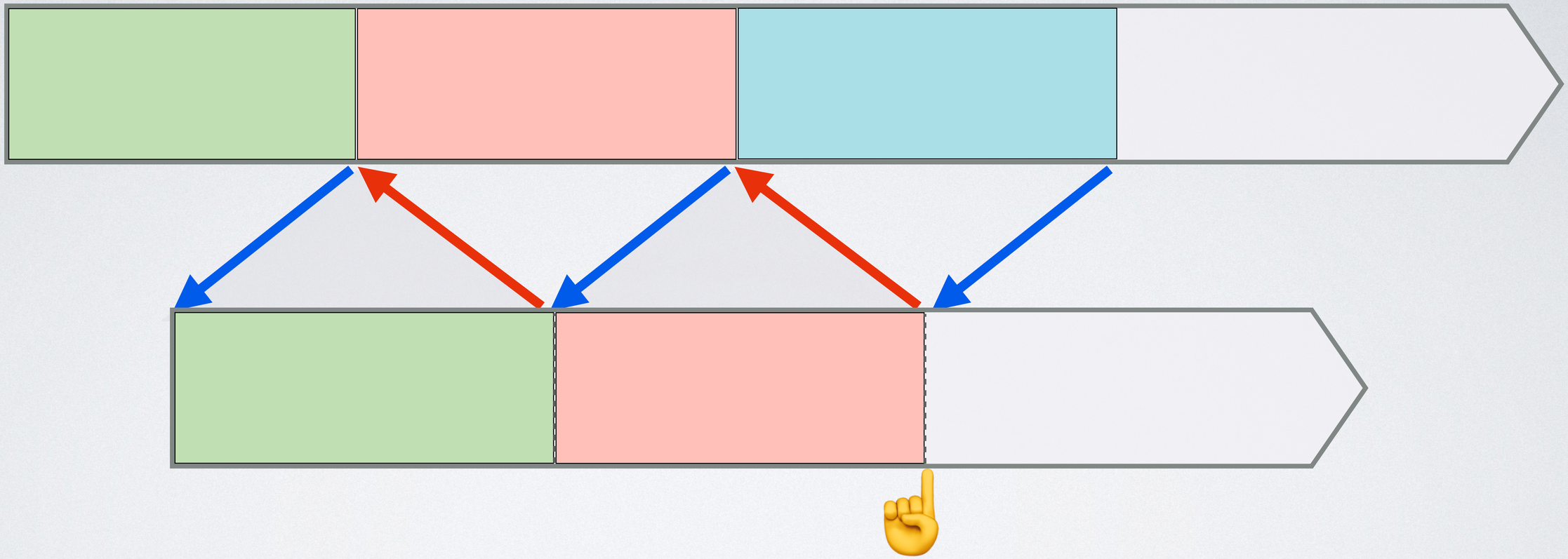
```
yield $v3
```

```
$generator->next()
```



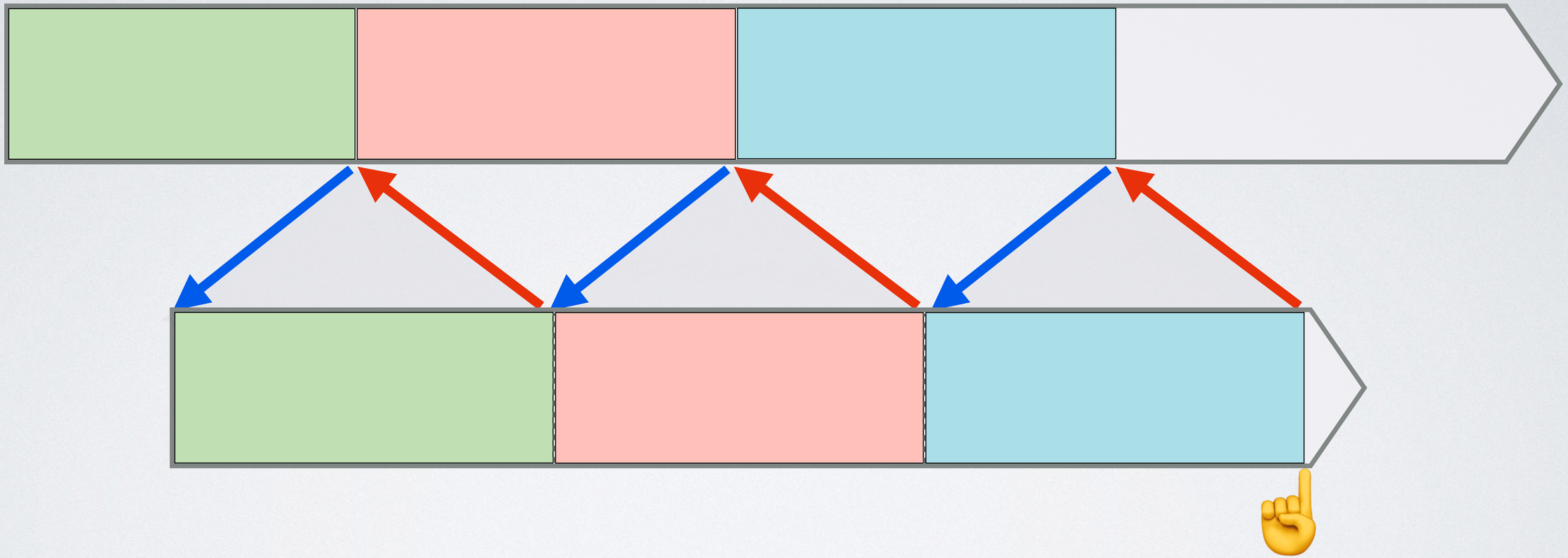
```
yield $v3
```

$generator->next()



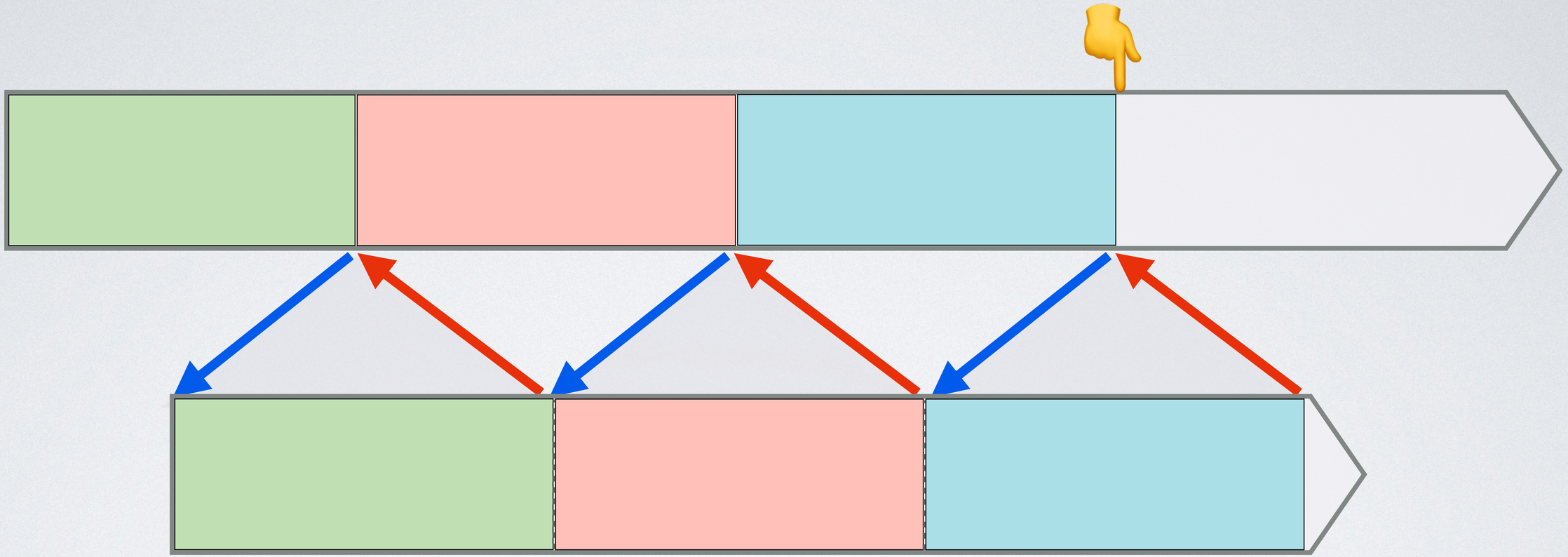
yield $v3

```
$generator->next()
```



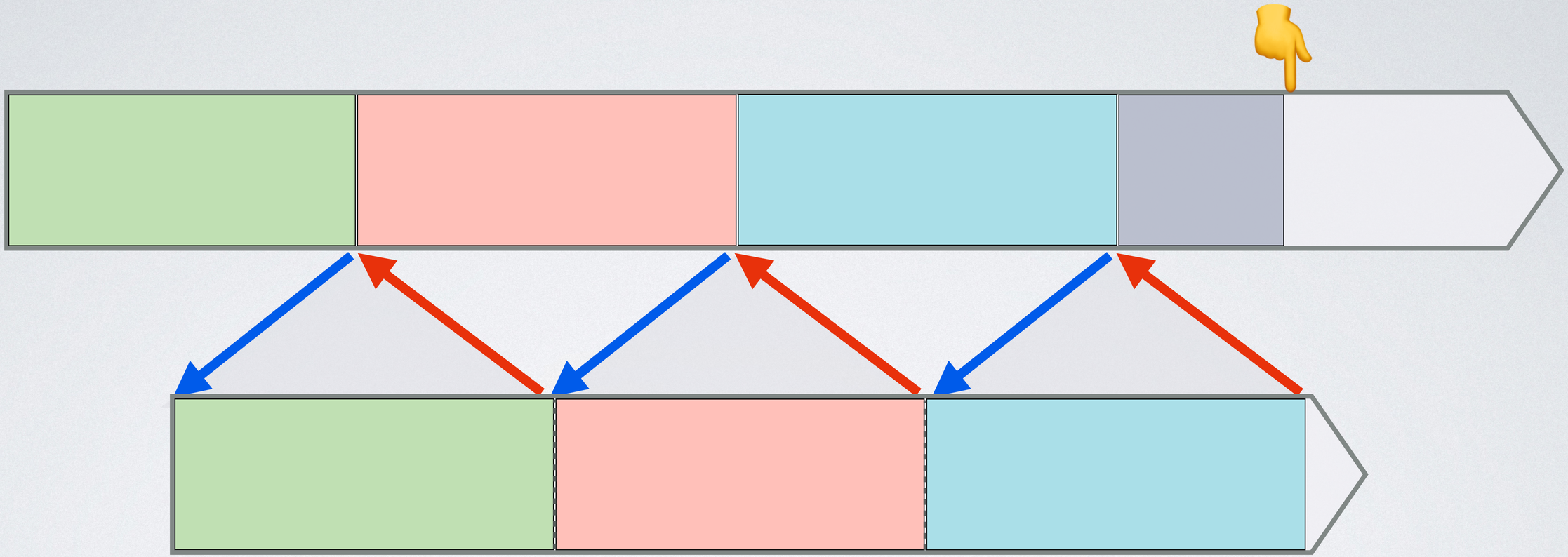
```
return $value;
```

```
$generator->next()
```



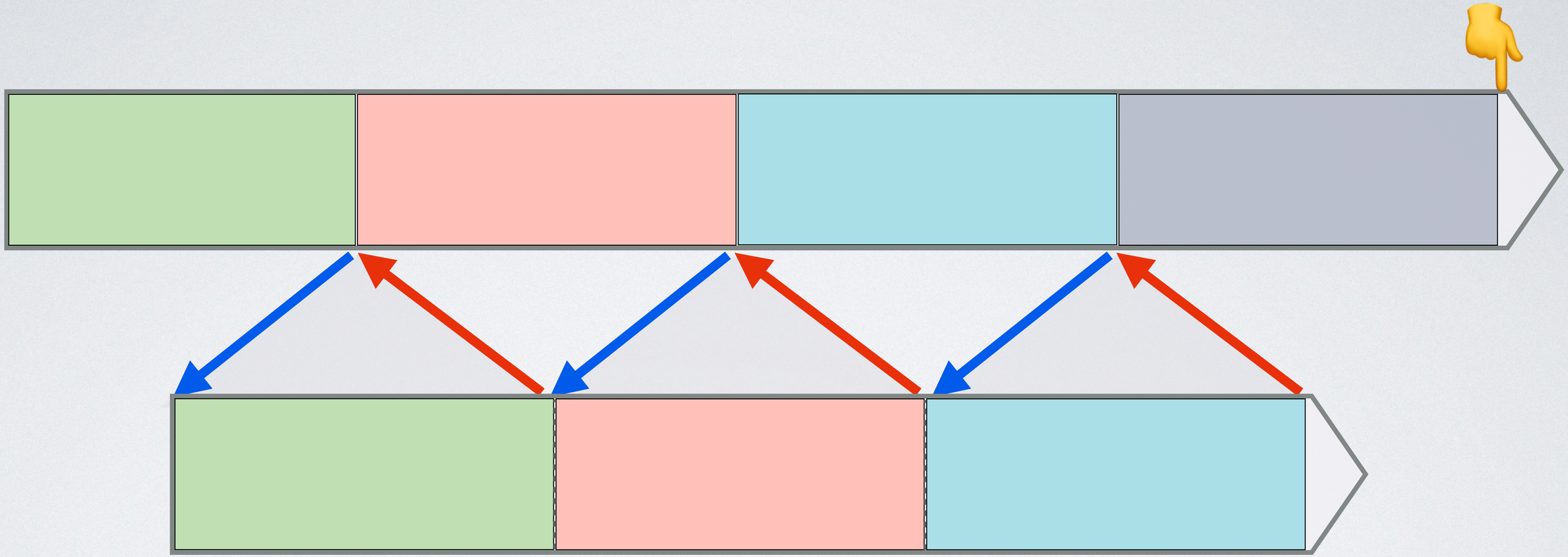
```
return $value;
```

```
!$generator->valid()
```



```
return $value;
```

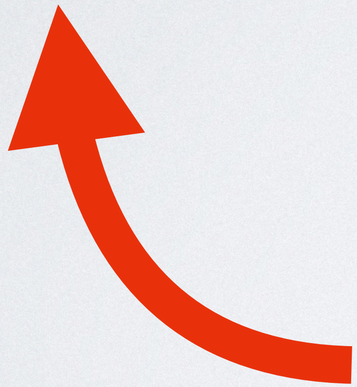
```
$value = $generator->getReturn()
```



```
return $value;
```

Outside

```php
try {
    $generator->next();
} catch (\Exception $exception)
{}
```

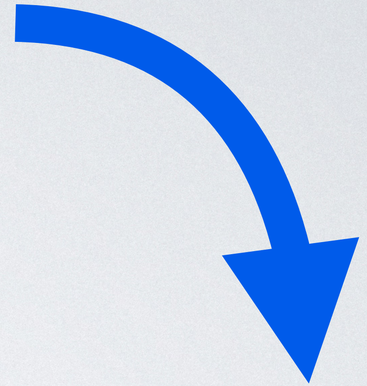


```php
yield;
throw new \Exception();
```

Inside

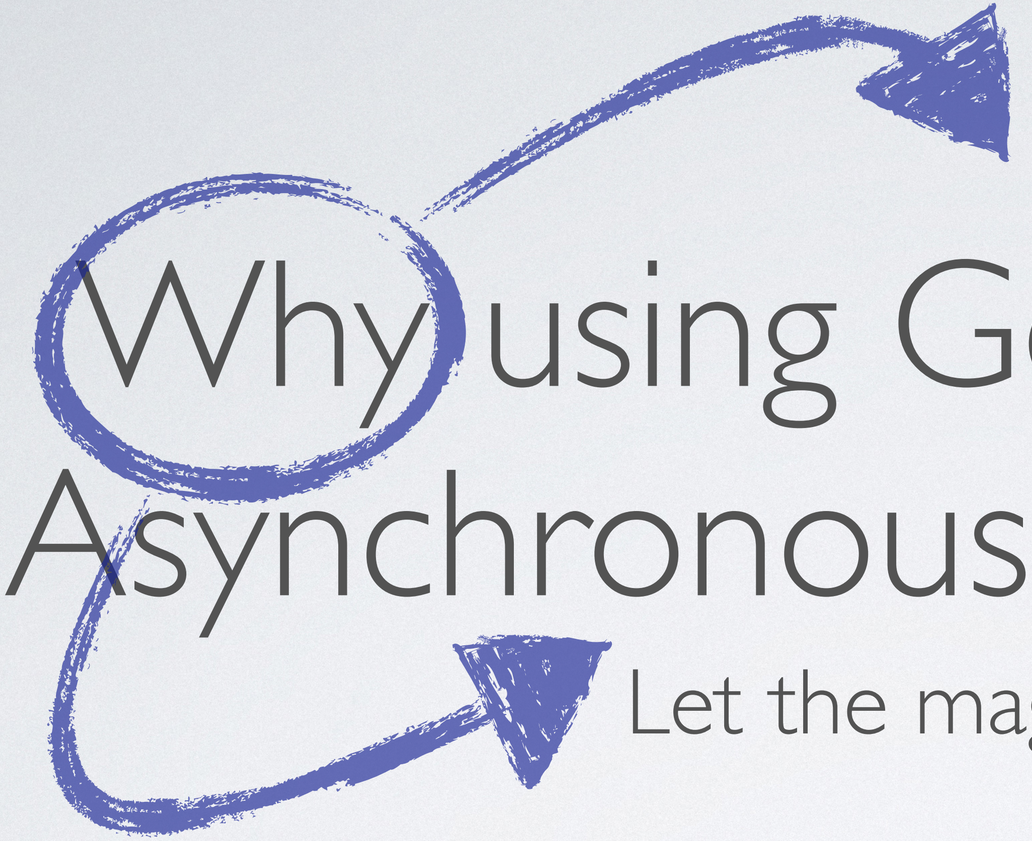
Outside

```
$generator->throw($exception);
```



Inside

```
try {
    yield;
} catch (\Exception $exception)
{}
```



Why using Generators for Asynchronous Programming

Let the magic happen

BEDROCK

Benoit Viguier

@b_viguier

ConFoo.CA

DEVELOPER CONFERENCE

Promises

Synchronous

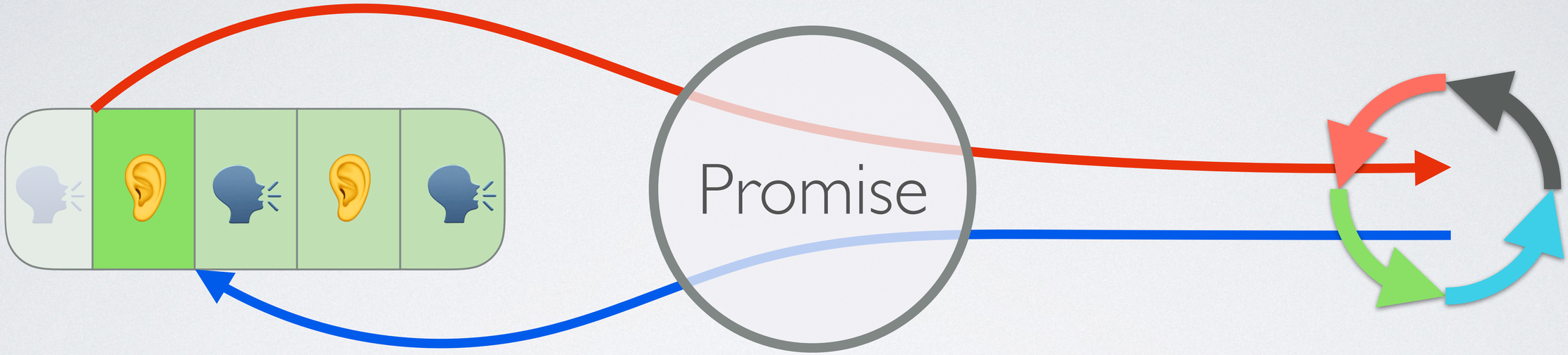
```php
function talk(string $question1, string $question2)
{
    /** @var string $response */
    $response = askQuestion($question1);
    echo $response;

    /** @var string $response */
    $response = askQuestion($question2);
    echo $response;
}
```

Synchronous

```
function askQuestion(string $question): string
{
    // …
}
```

Waiting for a result…



Here your result!

Asynchronous

```
function askQuestion(string $question): Promise
{
    // …
}
```

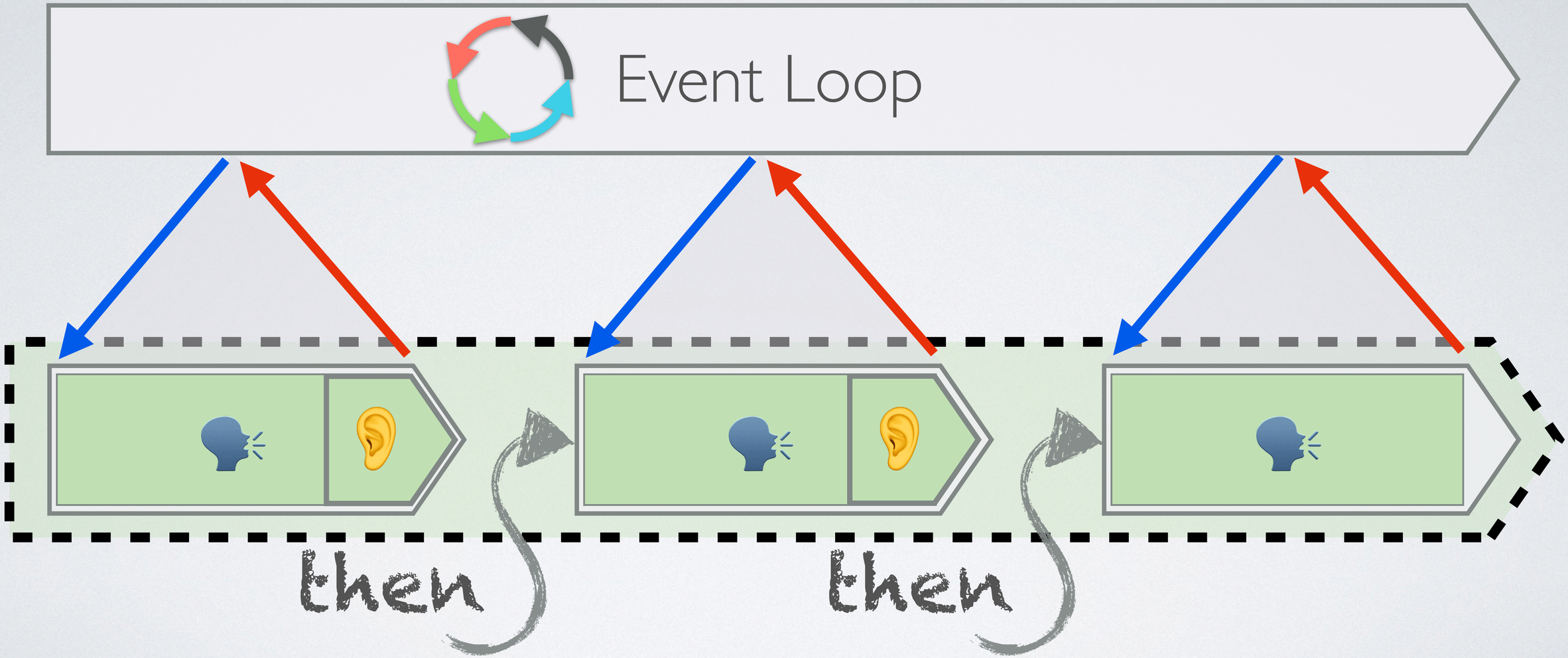
Asynchronous

```
function talk(string $question1, string $question2)
{
    /** @var Promise $promise */
    $promise = askQuestion($question1);
    // !? echo $response;

    /** @var Promise $promise */
    $promise = askQuestion($question2);
    // !? echo $response;
}
```

Promises A+

Thenable Promises


Event Loop
then
then

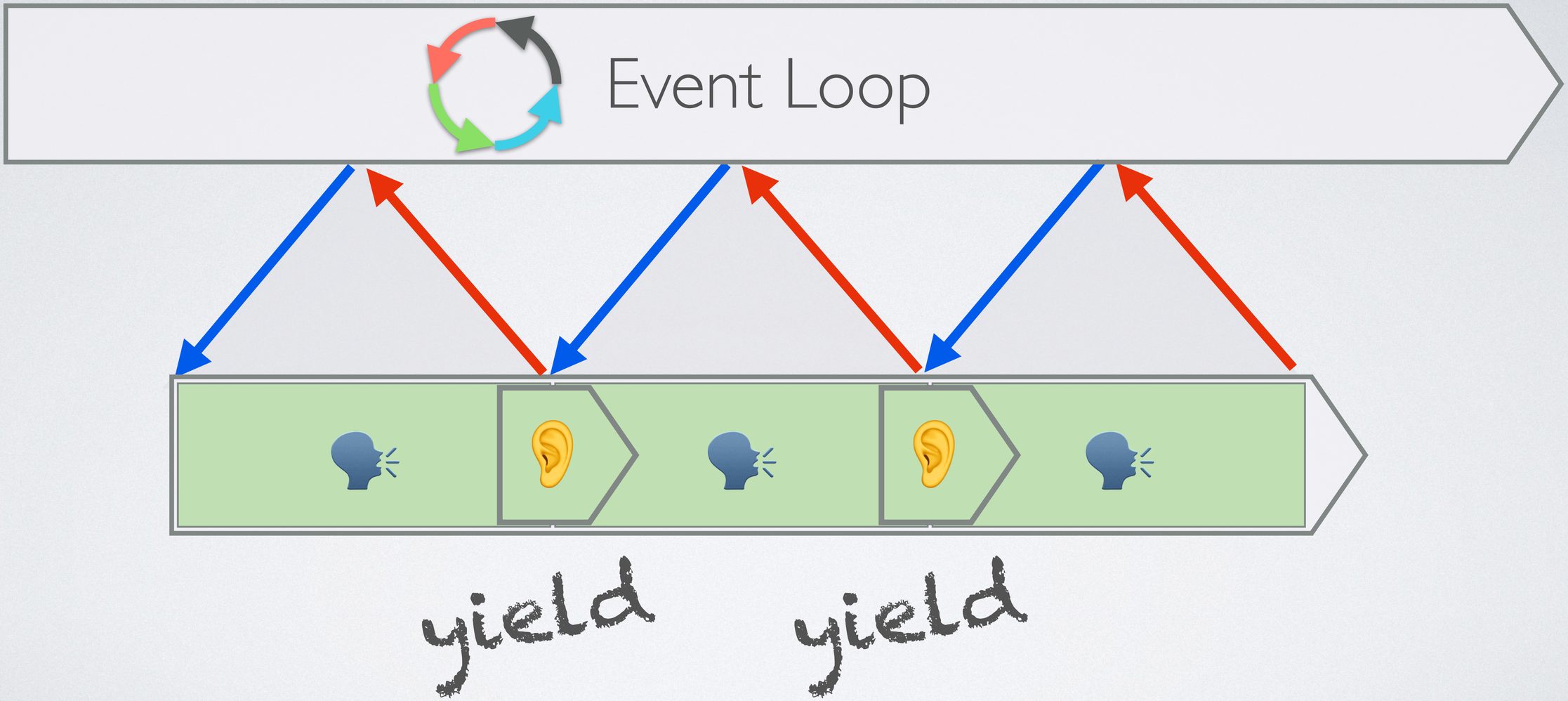
```php
function talk(string $question1, string $question2)
{
    $promise = askQuestion($question1);
    $promise->then(
        function(string $response) use($question2) {
            echo $response;

            return askQuestion($question2);
        }
    )->then(
        function(string $response) {
            echo $response;
        }
    );
}
```

🔥Callback Hell🔥

```
$promise
    ->then(function ($x) {
        return $x + 1;
    })
    ->then(function ($x) {
        throw new \Exception($x + 1);
    })
    ->otherwise(function (\Exception $x) {
        return $x->getMessage() + 1;
    })
    ->then(function ($x) {
        echo 'Mixed '.$x;
    });
```

Generator Promises


Event Loop
yield
yield

Synchronous

```php
function talk(string $question1, string $question2)
{

    /** @var string $response */
    $response = askQuestion($question1);
    echo $response;



    /** @var string $response */
    $response = askQuestion($question2);
    echo $response;
}
```

Asynchronous

```
function talk(string $question1, string $question2)
{

    /** @var string $response */
    $response = yield askQuestion($question1);
    echo $response;



    /** @var string $response */
    $response = yield askQuestion($question2);
    echo $response;
}
```

Asynchronous

```
function talk(string $question1, string $question2)
{
    $promise = askQuestion($question1);
    /** @var string $response */
    $response = yield $promise;
    echo $response;

    $promise = askQuestion($question2);
    /** @var string $response */
    $response = yield $promise;
    echo $response;
}
```

To asynchronously *wait*
the value of a **promise**,
use **`yield`**.

Asynchronous Function:

a **generator**

yielding only *promises*.

BEDROCK

Benoit Viguier
@b_viguier

ConFoo.CA
DEVELOPER CONFERENCE

Choose your Event Loop

reactphp/react

👍 Community

😩 *Thenable* Promises

amphp/amp

👍 Generators

🤔 Coupling ?

m6web/tornado

👍 Generators

😘 Interfaces / Adapters

🤔 M6Web

Tea for Two

Let's code it!

Discussing with a friend

```php
function talk(string $question1, string $question2)
: \Generator
{
    /** @var string $response */
    $response = yield askQuestion($question1);
    echo $response;

    /** @var string $response */
    $response = yield askQuestion($question2);
    echo $response;
}
```

Breathing

```
function breath(EventLoop $eventLoop, int $count)
: \Generator
{
    while (--$count >= 0) {
        echo "Breathing\n";
        yield $eventLoop->idle();
    }
}
```

Drinking Tea

```
function tMax(EventLoop $eventLoop, int $maximum)
: \Generator
{
    do {
        yield $eventLoop->idle();
        $current = rand(20, 40);
    } while ($current > $maximum);

    return $current;
}
```

Drinking Tea

```
function drink(EventLoop $eventLoop): \Generator
{
    $generator = tMax($eventLoop, 35);
    $promise = $eventLoop->async($generator);
    $temperature = yield $promise;

    echo "Drinking [$temperature °]\n";

    yield $eventLoop->delay(1000/*ms*/);
}
```

Drinking Tea

```
function drink(EventLoop $eventLoop): \Generator
{
    $generator = tMax($eventLoop, 35);
    $promise = $eventLoop->async($generator);
    $temperature = yield $promise;

    echo "Drinking [$temperature °]\n";

    yield $eventLoop->delay(1000/*ms*/);
}
```

Drinking Tea

```
function drink(EventLoop $eventLoop): \Generator
{
    $generator = tMax($eventLoop, 35);
    $promise = $eventLoop->async($generator);
    $temperature = yield $promise;

    echo "Drinking [$temperature °]\n";

    yield $eventLoop->delay(1000/*ms*/);
}
```

Drinking Tea

```
function drink(EventLoop $eventLoop): \Generator
{
    $generator = tMax($eventLoop, 35);
    $promise = $eventLoop->async($generator);
    $temperature = yield $promise;

    echo "Drinking [$temperature °]\n";

    yield $eventLoop->delay(1000/*ms*/);
}
```

Wait for it…

```
$concurrentPromise = $eventLoop->promiseAll(

    $eventLoop->async( breath($eventLoop, 10)       ),
    $eventLoop->async( discuss('Where?', ‘What?') ),
    $eventLoop->async( drink($eventLoop)            )

);

$eventLoop->wait($concurrentPromise);
```

Wait for it…

```
$concurrentPromise = $eventLoop->promiseAll(

    $eventLoop->async( breath($eventLoop, 10)       ),
    $eventLoop->async( discuss('Where?', ‘What?') ),
    $eventLoop->async( drink($eventLoop)            )

);

$eventLoop->wait($concurrentPromise);
```

Wait for it…

```
$concurrentPromise = $eventLoop->promiseAll(

    $eventLoop->async( breath($eventLoop, 10)       ),
    $eventLoop->async( discuss('Where?', ‘What?') ),
    $eventLoop->async( drink($eventLoop)            )

);

$eventLoop->wait($concurrentPromise);
```

So...

BEDROCK

Benoit Viguier

@b_viguier

Asynchronous Programming **doesn't** require…

~~Http Server~~


Threads


JavaScript™

Asynchronous Programming **requires**…


Event Loop

Interruptible Tasks

Generators:
Powerful tool for
Asynchronous Programming


PRACTICE

One year of asynchronous PHP in production

ConFoo.CA
DEVELOPER CONFERENCE

Tomorrow, 13:00

Why and how to use generators for asynchronous programming

Benoit Viguier
@b_viguier

Thanks !

```
$answer = yield $you->ask($me);
```

BEDROCK

Benoit Viguier
@b_viguier

ConFoo.CA
DEVELOPER CONFERENCE