ReactPhp AmPhp RevoltPhp

Asynchronous Frameworks Comparison

Benoit Viguier
@b_viguier@phpc.social

ReactPhp AmPhp RevoltPhp

Asynchronous Frameworks Comparison

Benoit Viguier
@b_viguier@phpc.social

FORUMPHP
PARIS2023

FORUMPHP PARIS2019

Generators for Asynchronous Programming

User Manual

Benoit Viguier
@b_viguier

6 WEB

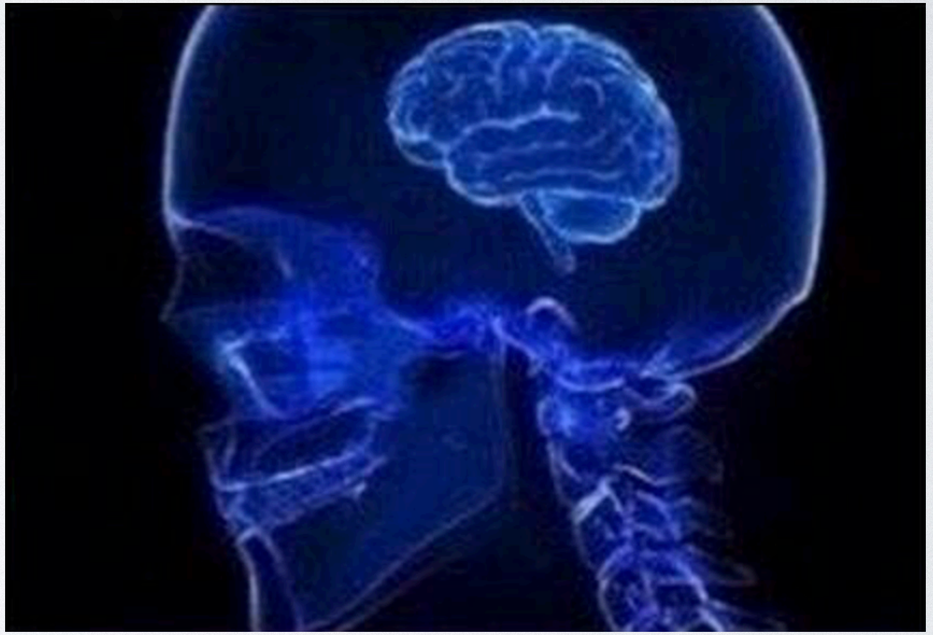
FORUMPHP PARIS2018

FORUMPHP PARIS2018

Long Story Short

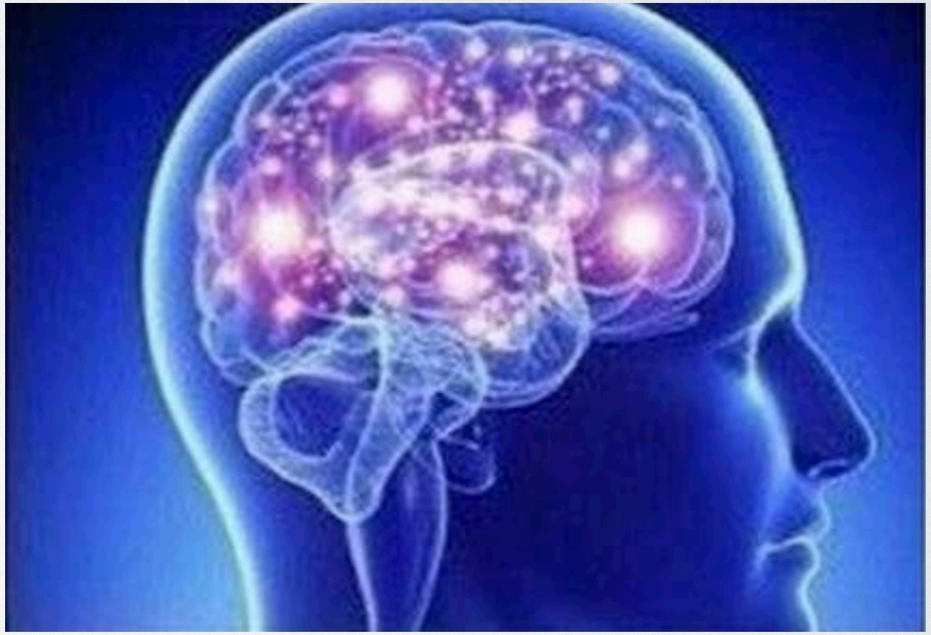
Long Story Short

- **Generators** are awesome for asynchronous programming!



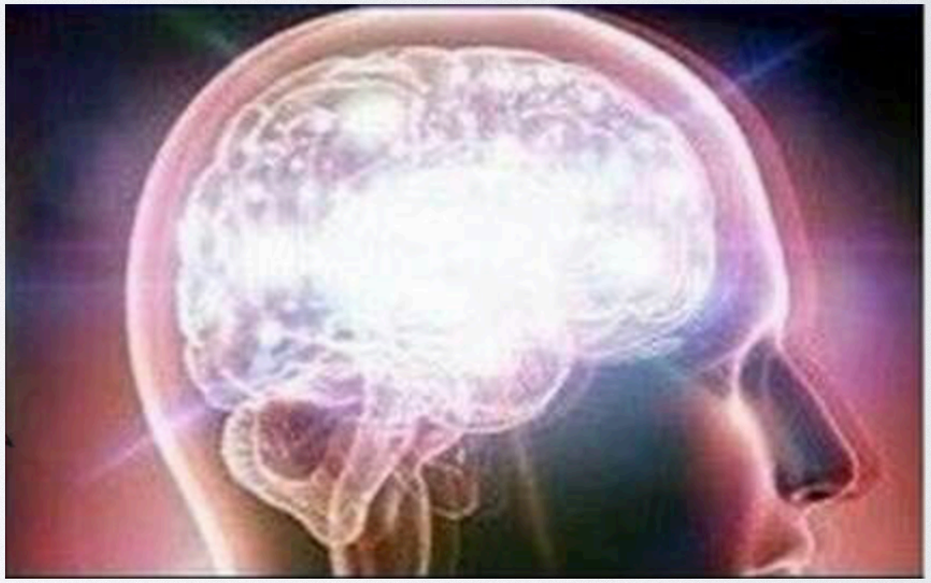
Long Story Short

- **Generators** are awesome for asynchronous programming!
- I used Generators in **production**!!



Long Story Short

- **Generators** are awesome for asynchronous programming!
- I used Generators in **production**!!
- Forget about Generators: use **Fibers**!!!



Long Story Short

- **Generators** are awesome for asynchronous programming!
- I used Generators in **production**!!
- Forget about Generators: use **Fibers**!!!
- Don't use Fibers, use a **Framework**!!!

lendable

Background

ReactPhp

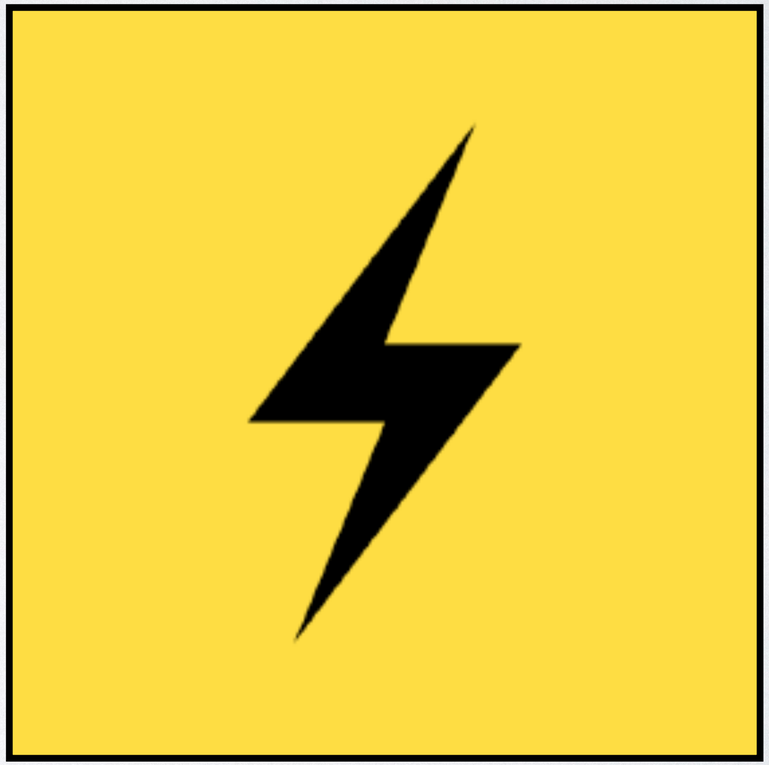
- **2012**: First Commit (SocketServer)
- « ***Thenable*** » promises
- 2022: react/async
- https://reactphp.org

AMPHP

- **2013**: First Commit (Alert)
- 2014: **Generators** / Coroutines
- 2022: 100% **RevoltPhp** based
- https://amphp.org

RevoltPhp

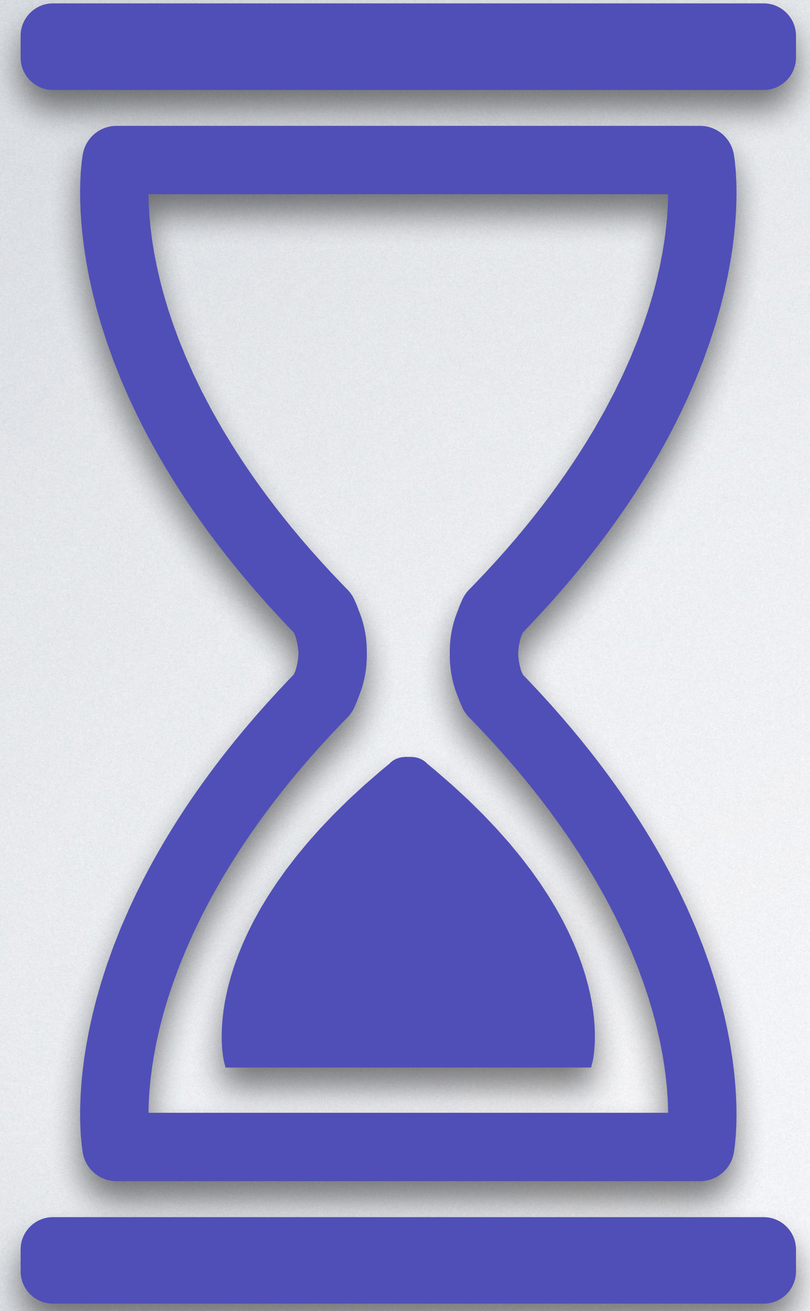
- **2021**: First Commit
- 2022: ReactPhp adapter
- 100% **Fibers**
- https://revolt.run



(Open) Swoole

- **2012**: First Commit
- Php **extension**
- Chinese community
- https://www.swoole.com/
- https://openswoole.com/

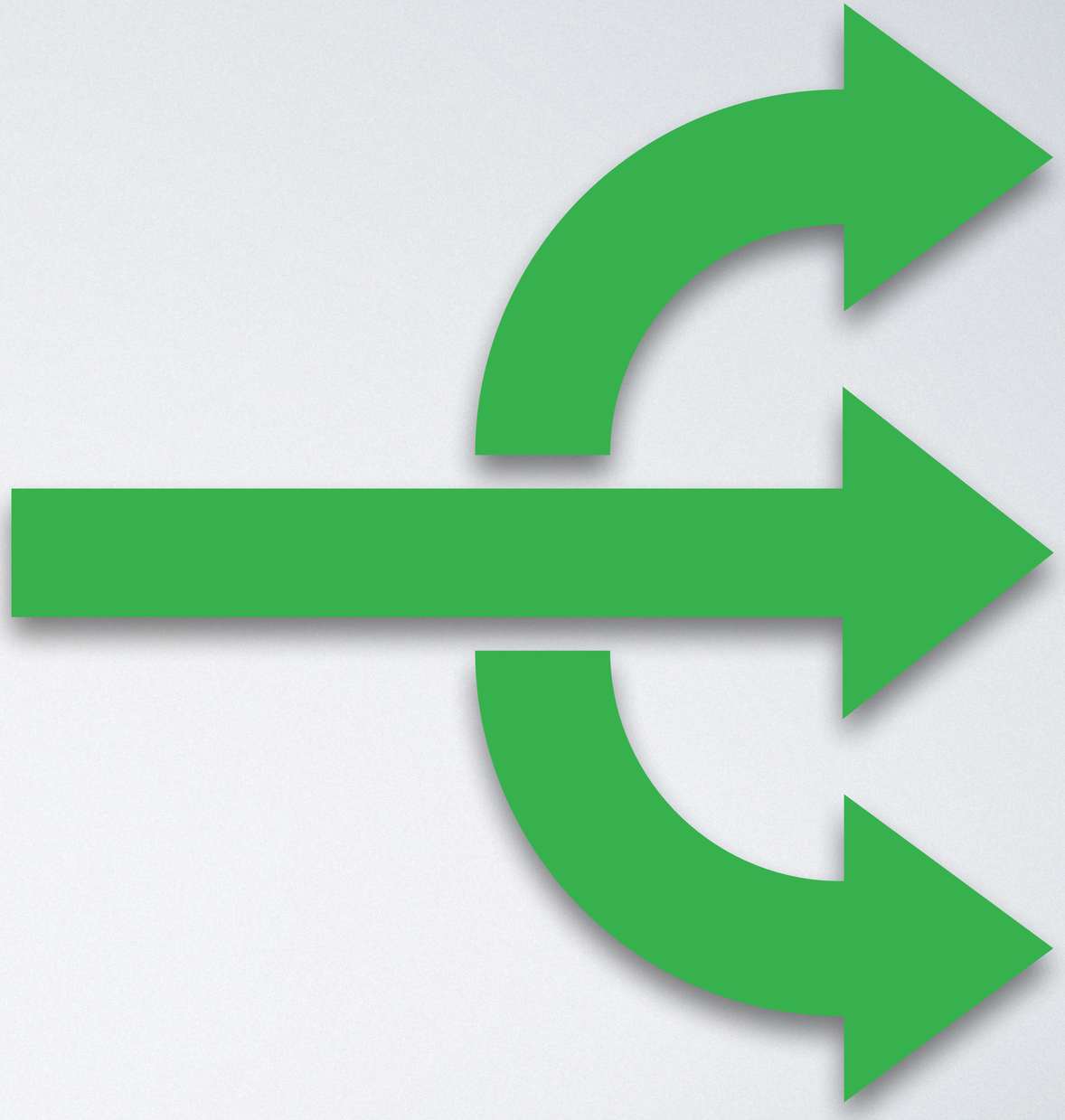
Asynchronous Programing: Core Components

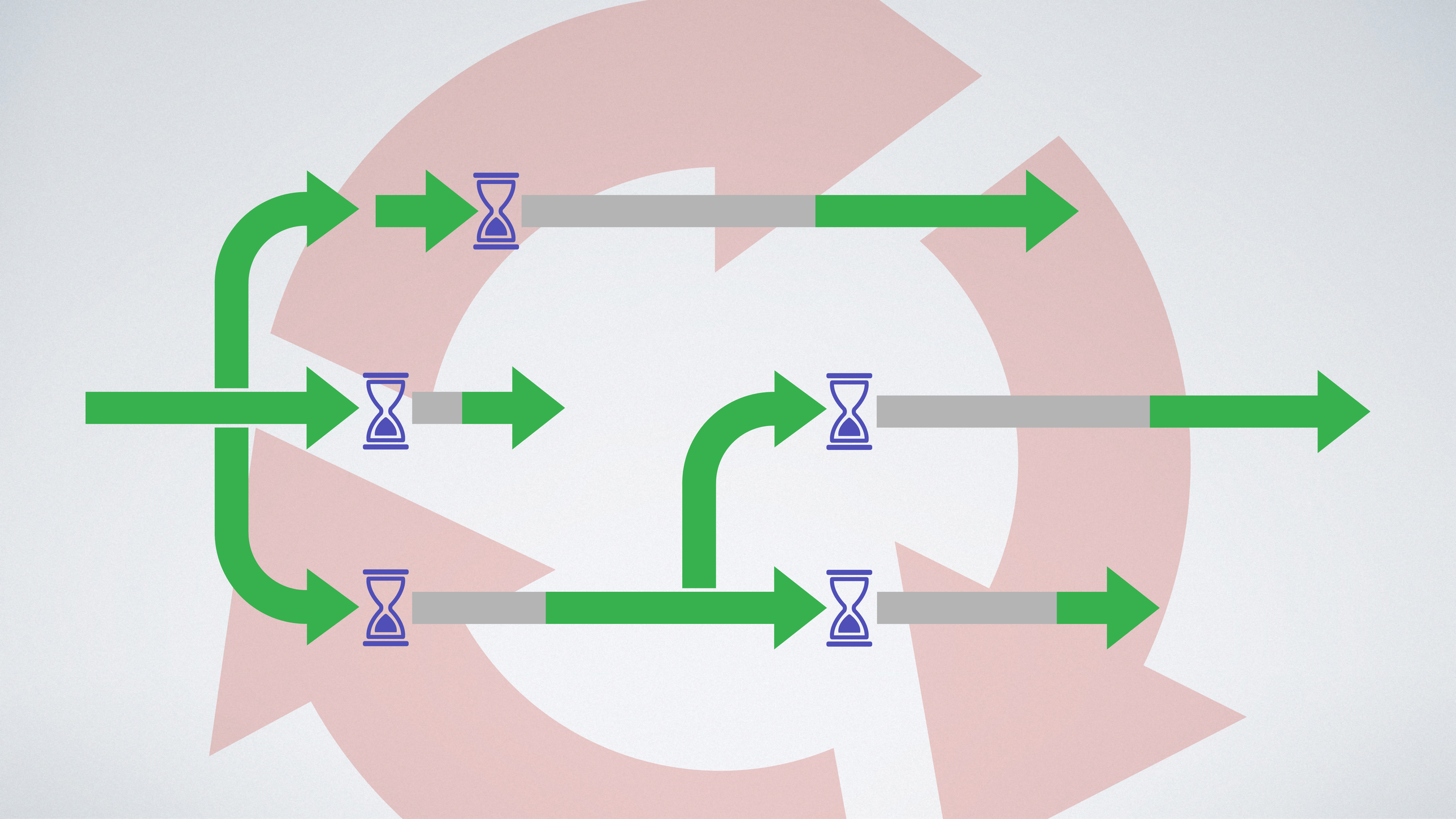


Asynchronous **Functions**


Event Loop

Concurrency

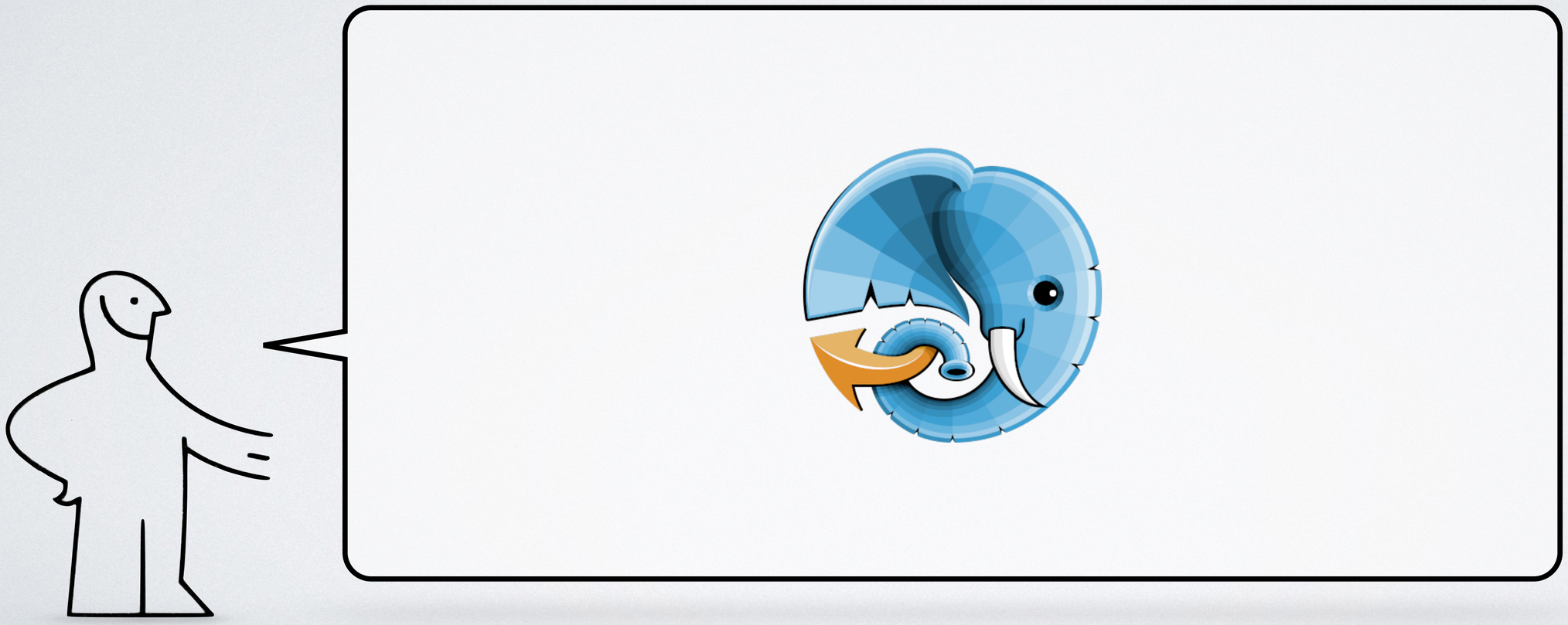




EXAMPLES!!

Example:

Singin’



```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; Amp\delay(0.5);
        echo "Boom💥\n"; Amp\delay(0.5);
        echo "Clap👏\n"; Amp\delay(1);
    }
}
```

```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; Amp\delay(0.5);
        echo "Boom💥\n"; Amp\delay(0.5);
        echo "Clap👏\n"; Amp\delay(1);
    }
}
```

```php
function sing(): void
{
    echo "\tWe\n";          Amp\delay(1);
    echo "\tWill\n";        Amp\delay(1);
    echo "\tWe\n";          Amp\delay(1);
    echo "\tWill\n";        Amp\delay(1);
    echo "\tRock\n";        Amp\delay(0.5);
    echo "\tYou!\n";
}
```

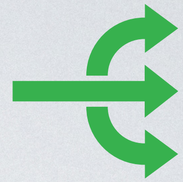
```php
function sing(): void
{
    echo "\tWe\n";
    echo "\tWill\n";
    echo "\tWe\n";
    echo "\tWill\n";
    echo "\tRock\n";
    echo "\tYou!\n";
}
```

```php
Amp\delay(1);
Amp\delay(1);
Amp\delay(1);
Amp\delay(1);
Amp\delay(0.5);
```

```php
Amp\async(beat(...));
Amp\async(sing(...));

echo "Start\n";
Revolt\EventLoop::run();
echo "Stop\n";
```

```php
Amp\async(beat(...));
Amp\async(sing(...));


echo "Start\n";
Revolt\EventLoop::run();
echo "Stop\n";
```

```php
Amp\async(beat(...));
Amp\async(sing(...));


echo "Start\n";
Revolt\EventLoop::run();
echo "Stop\n";
```



```php
Amp\async(beat(...));
Amp\async(sing(...));


echo "Start\n";
Revolt\EventLoop::run();
echo "Stop\n";
```

```
➜  amp php amp.php
```

Example:

Singin’

```php
function oneBar(): PromiseInterface {
    return React\Promise\resolve(null)
        ->then(function() {
            echo "Boom💥\n";
            return React\Promise\Timer\sleep(0.5);
        })
        ->then(function() {
            echo "Boom💥\n";
            return React\Promise\Timer\sleep(0.5);
        })
        ->then(function() {
            echo "Clap👏\n";
            return React\Promise\Timer\sleep(1);
        });
}
```

```php
function oneBar(): PromiseInterface {
    return React\Promise\resolve(null)
        ->then(function() {
            echo "Boom💥\n";
            return React\Promise\Timer\sleep(0.5);
        })
        ->then(function() {
            echo "Boom💥\n";
            return React\Promise\Timer\sleep(0.5);
        })
        ->then(function() {
            echo "Clap👏\n";
            return React\Promise\Timer\sleep(1);
        });
}
```

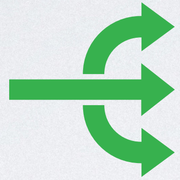
```
function beat(): PromiseInterface
{
    return oneBar()
        ->then(oneBar(...))
        ->then(oneBar(...))
        ->then(oneBar(...))
    ;
}
```

```php
function sing(): PromiseInterface {
    echo "\tWe\n";
    return React\Promise\Timer\sleep(1)
        ->then(function() {
            echo "\tWill\n";
            return React\Promise\Timer\sleep(1);
        })->then(function() {
            echo "\tWe\n";
            return React\Promise\Timer\sleep(1);
        })->then(function() {
            echo "\tWill\n";
            return React\Promise\Timer\sleep(1);
        })->then(function() {
            echo "\tRock\n";
            return React\Promise\Timer\sleep(0.5);
        })->then(function() {
            echo "\tYou\n";
        });
}
```

```php
function sing(): PromiseInterface {
    echo "\tWe\n";
    return React\Promise\Timer\sleep(1)
        ->then(function() {
            echo "\tWill\n";
            return React\Promise\Timer\sleep(1);
        })->then(function() {
            echo "\tWe\n";
            return React\Promise\Timer\sleep(1);
        })->then(function() {
            echo "\tWill\n";
            return React\Promise\Timer\sleep(1);
        })->then(function() {
            echo "\tRock\n";
            return React\Promise\Timer\sleep(0.5);
        })->then(function() {
            echo "\tYou\n";
        });
}
```

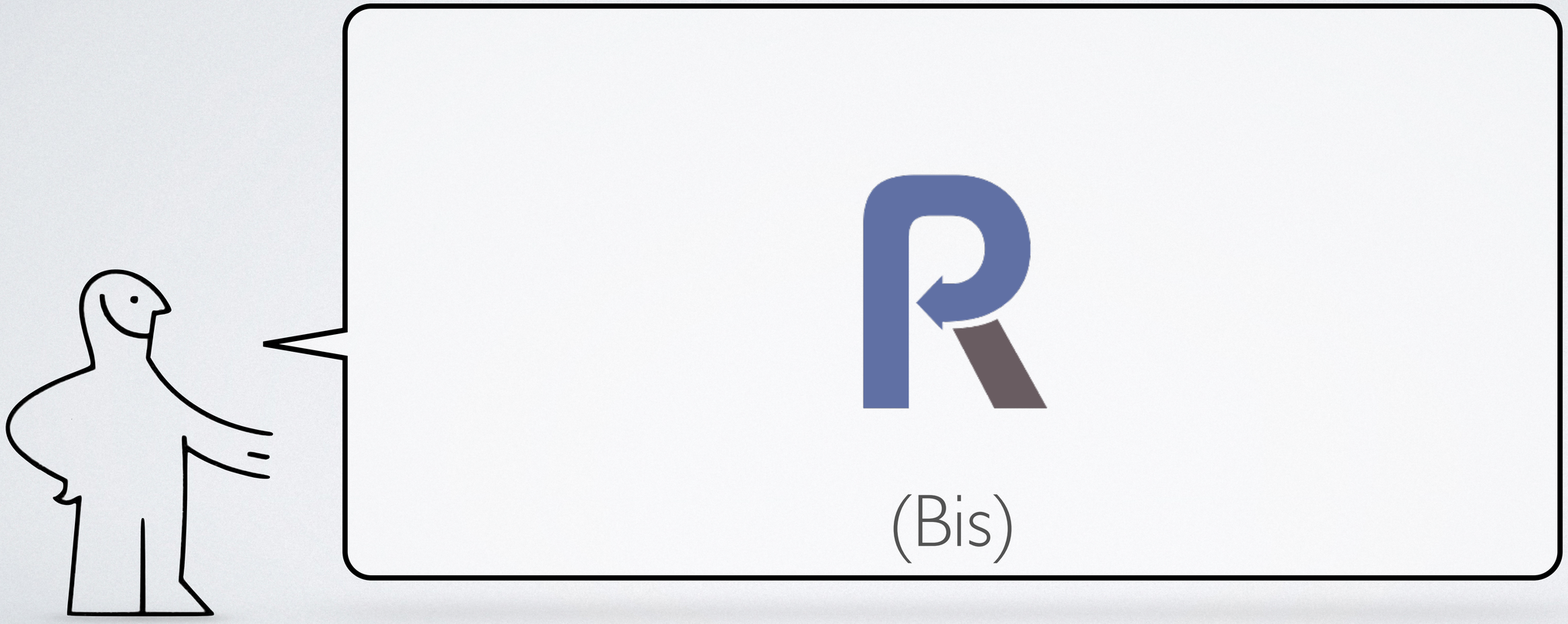
```php
echo "Start\n";

beat();
sing();

React\EventLoop\Loop::run();
echo "Stop\n";
```

```php
echo "Start\n";

beat();
sing();

React\EventLoop\Loop::run();
echo "Stop\n";
```



Example:

Singin’



```
composer require react/async
```

```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; Async\delay(0.5);
        echo "Boom💥\n"; Async\delay(0.5);
        echo "Clap👏\n"; Async\delay(1);
    }
}
```

```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; Async\delay(0.5);
        echo "Boom💥\n"; Async\delay(0.5);
        echo "Clap👏\n"; Async\delay(1);
    }
}
```

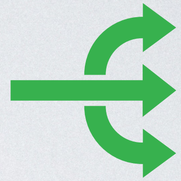
```
function sing(): void
{
    echo "\tWe\n";          Async\delay(1);
    echo "\tWill\n";        Async\delay(1);
    echo "\tWe\n";          Async\delay(1);
    echo "\tWill\n";        Async\delay(1);
    echo "\tRock\n";        Async\delay(0.5);
    echo "\tYou!\n";
}
```

```
function sing(): void
{
    echo "\tWe\n";
    echo "\tWill\n";
    echo "\tWe\n";
    echo "\tWill\n";
    echo "\tRock\n";
    echo "\tYou!\n";
}
```

```
Async\delay(1);
Async\delay(1);
Async\delay(1);
Async\delay(1);
Async\delay(0.5);
```

```php
echo "Start\n";

React\Async\async(beat(...))();
React\Async\async(sing(...))();

React\EventLoop\Loop::run();
echo "Stop\n";
```

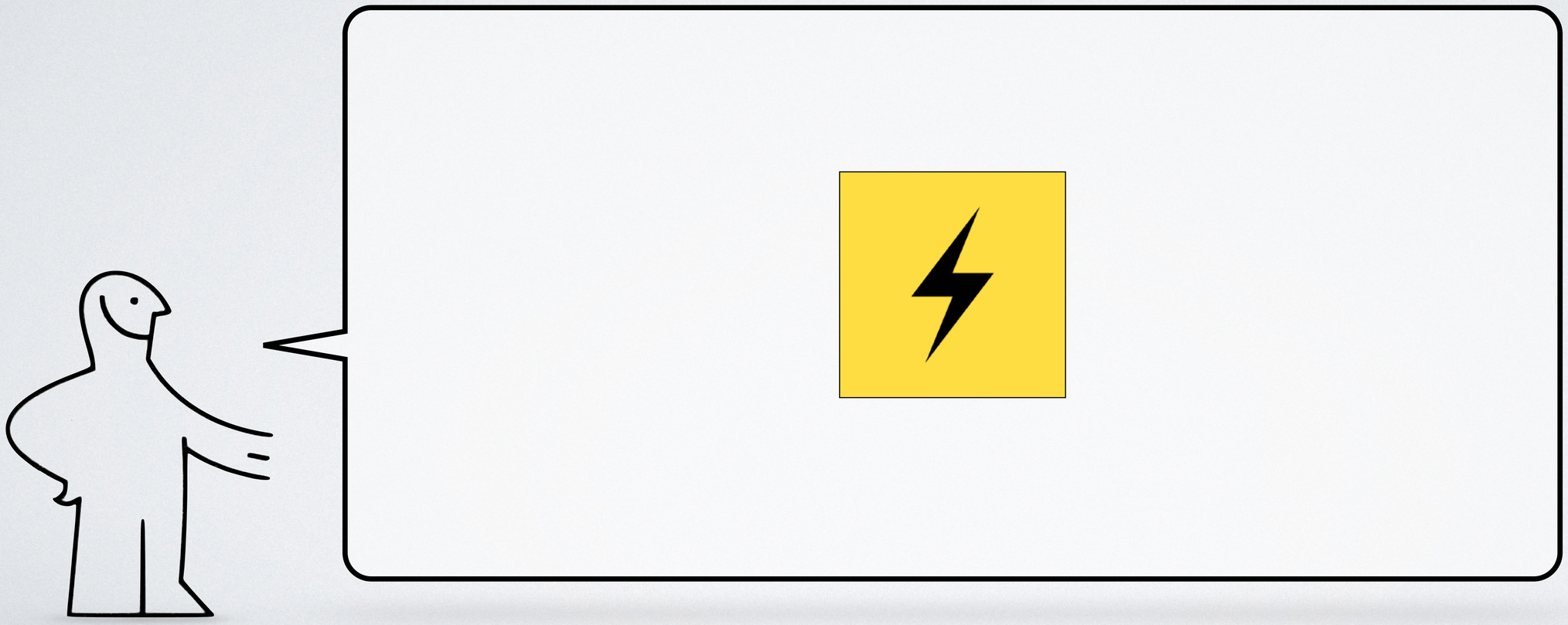
```php
echo "Start\n";

React\Async\async(beat(...))();
React\Async\async(sing(...))();

React\EventLoop\Loop::run();
echo "Stop\n";
```



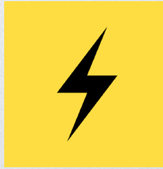
```php
echo "Start\n";

React\Async\async(beat(...))();
React\Async\async(sing(...))();

React\EventLoop\Loop::run();
echo "Stop\n";
```

Example:

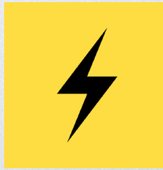
Singin’



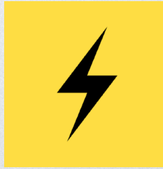
```php
function oneBar(callable $then): void
{
    echo "Boom💥\n";
    EventLoop::delay(0.5, function () use($then)
    {
        echo "Boom💥\n";
        EventLoop::delay(0.5, function () use($then)
        {
            echo "Clap👏\n";
            EventLoop::delay(1, $then);
        });
    });
}
```



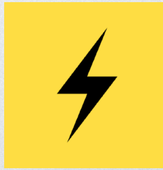
```php
function oneBar(callable $then): void
{
    echo "Boom💥\n";
    EventLoop::delay(0.5, function () use($then)
    {
        echo "Boom💥\n";
        EventLoop::delay(0.5, function () use($then)
        {
            echo "Clap👏\n";
            EventLoop::delay(1, $then);
        });
    });
}
```



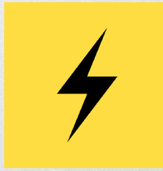
```
function beat(): void {
   oneBar(
         fn() => oneBar(
               fn() => oneBar(
                     fn() => oneBar(
                           fn() => null
                     )
               )
         )
   );
}
```



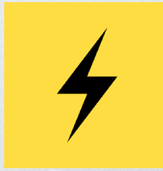
```php
function sing(): void {
    echo "\tWe\n";
    EventLoop::delay(1, function() {
        echo "\tWill\n";
        EventLoop::delay(1, function() {
            echo "\tWe\n";
            EventLoop::delay(1, function() {
                echo "\tWill\n";
                EventLoop::delay(1, function() {
                    echo "\tRock\n";
                    EventLoop::delay(0.5, function() {
                        echo "\tYou!\n";
                    });
}); }); }); }); }
```



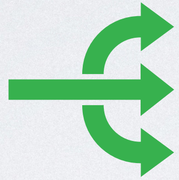
```php
function sing(): void {
    echo "\tWe\n";
    EventLoop::delay(1, function() {
        echo "\tWill\n";
        EventLoop::delay(1, function() {
            echo "\tWe\n";
            EventLoop::delay(1, function() {
                echo "\tWill\n";
                EventLoop::delay(1, function() {
                    echo "\tRock\n";
                    EventLoop::delay(0.5, function() {
                        echo "\tYou!\n";
                    });
}); }); }); }); }
```



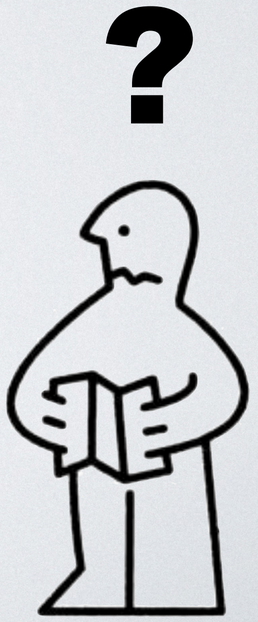
```php
echo "Start\n";

beat();
sing();

Revolt\EventLoop::run();
echo "Stop\n";
```



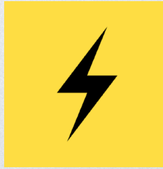
```php
echo "Start\n";

beat();
sing();

Revolt\EventLoop::run();
echo "Stop\n";
```



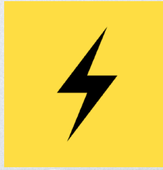
What did we **miss**?



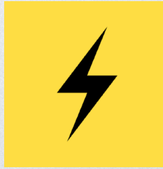
```php
interface Suspension
{
    public function suspend(): mixed;

    public function resume(mixed $v = null): void;

    public function throw(\Throwable $t): void;
}
```



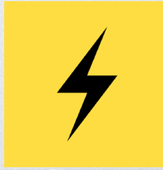
```php
$suspension = EventLoop::getSuspension();
```



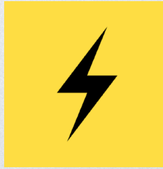
```php
$suspension = EventLoop::getSuspension();

$v = $suspension->suspend(); // ❌
$suspension->resume(42);     // ❌
```



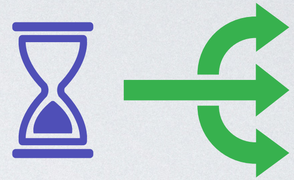
```php
$suspension = EventLoop::getSuspension();

$v = $suspension->suspend(); // ❌
$suspension->resume(42);     // ❌

$suspension->resume(42);     // ❌
$v = $suspension->suspend(); // ❌
```

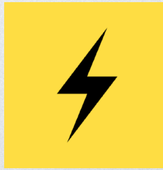


```php
$suspension = EventLoop::getSuspension();
EventLoop::delay(1,
 function() use($suspension) {
    echo "... time elapsed!\n";
    $suspension->resume(42);
 });

echo "Let's stop...\n";
$v = $suspension->suspend();
echo "Finished: $v\n";
```

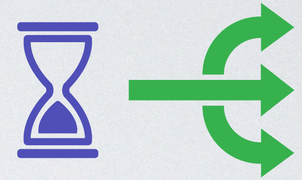


```php
$suspension = EventLoop::getSuspension();
EventLoop::delay(1,
  function() use($suspension) {
      echo "... time elapsed!\n";
      $suspension->resume(42);
  });

echo "Let's stop...\n";
$v = $suspension->suspend();
echo "Finished: $v\n";
```

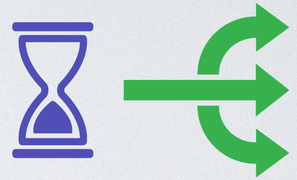


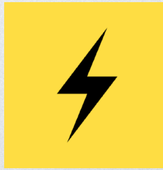


```php
$suspension = EventLoop::getSuspension();
EventLoop::delay(1,
 function() use($suspension) {
     echo "... time elapsed!\n";
     $suspension->resume(42);
 });

echo "Let's stop...\n";
$v = $suspension->suspend();
echo "Finished: $v\n";
```



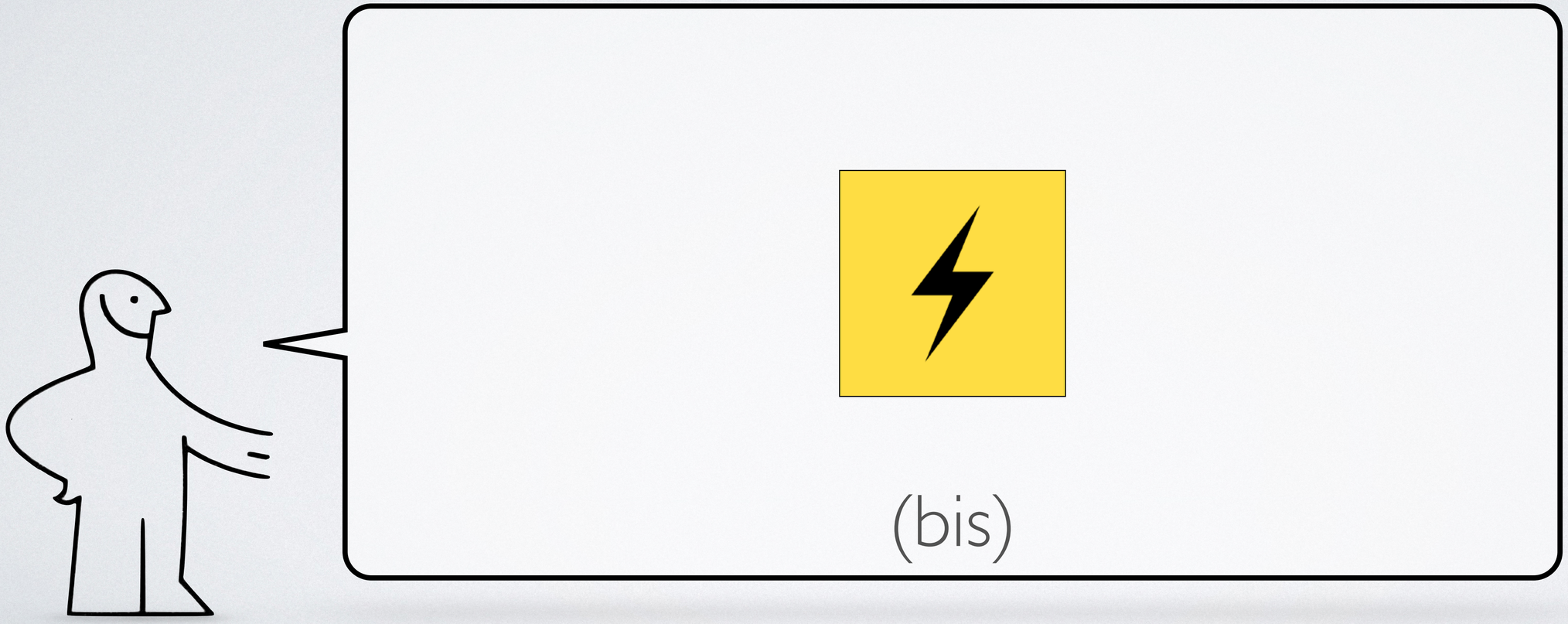
```php
$suspension = EventLoop::getSuspension();
EventLoop::delay(1,
 function() use($suspension) {
     echo "... time elapsed!\n";
     $suspension->resume(42);
 });

echo "Let's stop...\n";
$v = $suspension->suspend();
echo "Finished: $v\n";
```



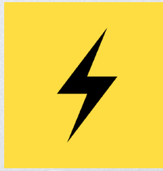


Example:

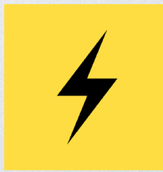
Singin’



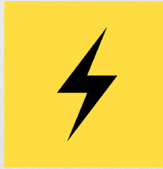
```php
function myDelay(float $seconds): void
{
    $s = EventLoop::getSuspension();

    EventLoop::delay(
        $seconds,
        $s->resume(...)
    );

    $s->suspend();
}
```



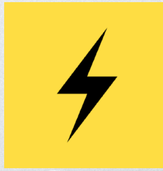
```php
function myDelay(float $seconds): void
{
    $s = EventLoop::getSuspension();

    EventLoop::delay(
        $seconds,
        $s->resume(...)
    );

    $s->suspend();
}
```



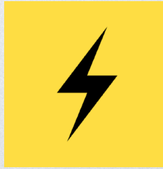
```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; myDelay(0.5);
        echo "Boom💥\n"; myDelay(0.5);
        echo "Clap👏\n"; myDelay(1);
    }
}
```



```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; myDelay(0.5);
        echo "Boom💥\n"; myDelay(0.5);
        echo "Clap👏\n"; myDelay(1);
    }
}
```



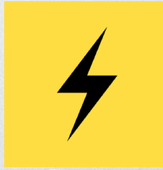
```
function sing(): void
{
    echo "\tWe\n";          myDelay(1);
    echo "\tWill\n";        myDelay(1);
    echo "\tWe\n";          myDelay(1);
    echo "\tWill\n";        myDelay(1);
    echo "\tRock\n";        myDelay(0.5);
    echo "\tYou!\n";
}
```

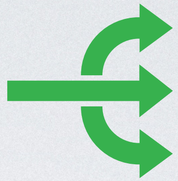


```
function sing(): void
{
    echo "\tWe\n";
    echo "\tWill\n";
    echo "\tWe\n";
    echo "\tWill\n";
    echo "\tRock\n";
    echo "\tYou!\n";
}
```

```
myDelay(1);
myDelay(1);
myDelay(1);
myDelay(1);
myDelay(0.5);
```

```php
Revolt\EventLoop::queue(beat(...));
Revolt\EventLoop::queue(sing(...));

echo "Start\n";
Revolt\EventLoop::run();
echo "Stop\n";
```





```php
Revolt\EventLoop::queue(beat(...));
Revolt\EventLoop::queue(sing(...));

echo "Start\n";
Revolt\EventLoop::run();
echo "Stop\n";
```

Example:

Return

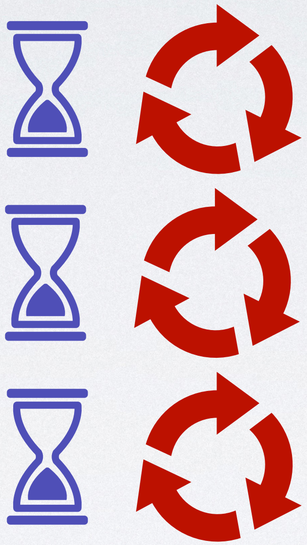
```php
function getNumber(int $n): int
{
    Amp\delay($n);
    echo "Computed $n\n";

    return $n;
}
```

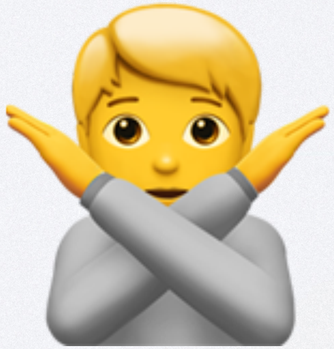
```
function getNumber(int $n): int
{
    Async\delay($n);
    echo "Computed $n\n";

    return $n;
}
```

```php
function getNumber(int $n): int
{
    myDelay($n);
    echo "Computed $n\n";

    return $n;
}
```

```
$s = getNumber(1)
   + getNumber(3)
   + getNumber(2)
;
```

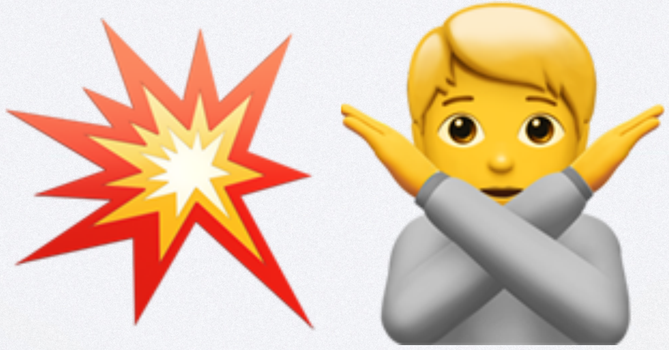
```
$s = getNumber(1)
   + getNumber(3)
   + getNumber(2)
;
```





```
$s = Amp\async(getNumber(...), 1)
   + Amp\async(getNumber(...), 3)
   + Amp\async(getNumber(...), 2)
;
```

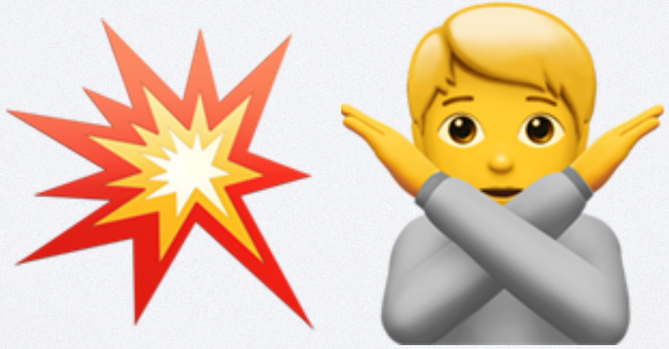
```
$s  = Amp\async(getNumber(...), 1)
    + Amp\async(getNumber(...), 3)
    + Amp\async(getNumber(...), 2)
;
```



```
$s = Amp\async(getNumber(...), 1)
   + Amp\async(getNumber(...), 3)
   + Amp\async(getNumber(...), 2)
;
```

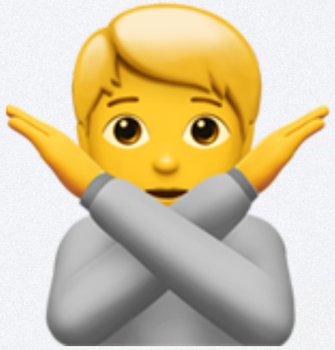
```
$s = Async\async(getNumber(...))(1)
   + Async\async(getNumber(...))(3)
   + Async\async(getNumber(...))(2)
;
```

```
$s = Async\async(getNumber(...))(1)
   + Async\async(getNumber(...))(3)
   + Async\async(getNumber(...))(2)
;
```



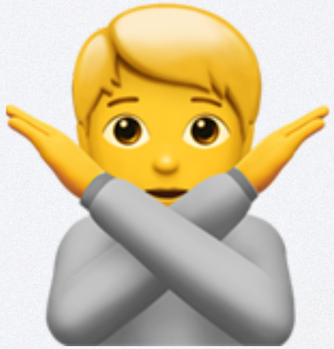
```
$s =
    Async\await(Async\async(getNumber(...))(1))
 + Async\await(Async\async(getNumber(...))(3))
 + Async\await(Async\async(getNumber(...))(2))
;
```

```
$s =
    Async\await(Async\async(getNumber(...))(1))
  + Async\await(Async\async(getNumber(...))(3))
  + Async\await(Async\async(getNumber(...))(2))
;
```



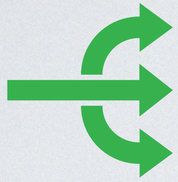
```
$s = Amp\async(getNumber(...), 1)->await()
   + Amp\async(getNumber(...), 3)->await()
   + Amp\async(getNumber(...), 2)->await()
;
```

```
$s = Amp\async(getNumber(...), 1)->await()
   + Amp\async(getNumber(...), 3)->await()
   + Amp\async(getNumber(...), 2)->await()
;
```



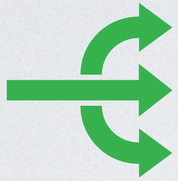
```
$r = Amp\Future\await([
    Amp\async(getNumber(...), 1),
    Amp\async(getNumber(...), 3),
    Amp\async(getNumber(...), 2),
]);
$s = array_sum($r);
```

```
$r = Amp\Future\await([
    Amp\async(getNumber(...), 1),
    Amp\async(getNumber(...), 3),
    Amp\async(getNumber(...), 2),
]);
$s = array_sum($r);
```

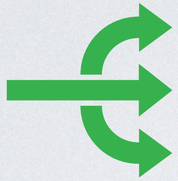


```
$r = React\Async\await(
    \React\Promise\all([
        Async\async(getNumber(...))(1),
        Async\async(getNumber(...))(3),
        Async\async(getNumber(...))(2),
    ])
);
$s = array_sum($r);
```

```
$r = React\Async\await(
    \React\Promise\all([
        Async\async(getNumber(...))(1),
        Async\async(getNumber(...))(3),
        Async\async(getNumber(...))(2),
    ])
);
$s = array_sum($r);
```

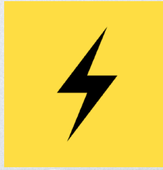


```
$r = React\Async\await(
    \React\Promise\all([
        Async\async(getNumber(...))(1),
        Async\async(getNumber(...))(3),
        Async\async(getNumber(...))(2),
    ])
);
$s = array_sum($r);
```

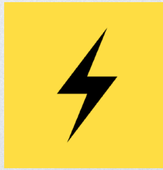


```
$r = all(
      fn() => getNumber(1),
      fn() => getNumber(3),
      fn() => getNumber(2),
);
$s = array_sum($r);
```

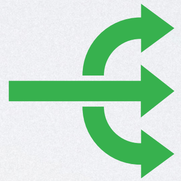
```php
function all(callable ...$functions): array {
    $s = Revolt\EventLoop::getSuspension();

    $wrapper = …;

    foreach($functions as $idx => $f) {
        Revolt\EventLoop::queue(
            $wrapper, $idx, $f
        );
    }

    return $s->suspend();
}
```

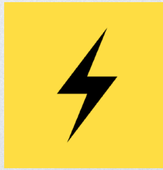


```php
function all(callable ...$functions): array {
    $s = Revolt\EventLoop::getSuspension();

    $wrapper = …;

    foreach($functions as $idx => $f) {
        Revolt\EventLoop::queue(
            $wrapper, $idx, $f
        );
    }

    return $s->suspend();
}
```

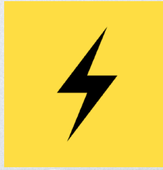


```php
function all(callable ...$functions): array {
    $s = Revolt\EventLoop::getSuspension();

    $wrapper = …;

    foreach($functions as $idx => $f) {
        Revolt\EventLoop::queue(
            $wrapper, $idx, $f
        );
    }

    return $s->suspend();
}
```

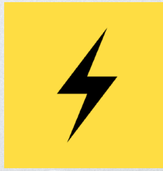




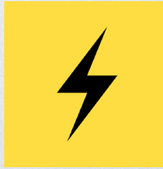
```php
$nb = count($functions);
$res = [];
$wrapper =
  function(int $idx, callable $function)
    use(&$res, &$nb, $s) {

    $res[$idx] = $function();
    if(--$nb === 0) {
        $s->resume($res);
    }
  }
;
```



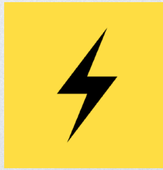
```php
$nb = count($functions);
$res = [];
$wrapper =
   function(int $idx, callable $function)
      use(&$res, &$nb, $s) {

      $res[$idx] = $function();
      if(--$nb === 0) {
            $s->resume($res);
      }
   }
;
```

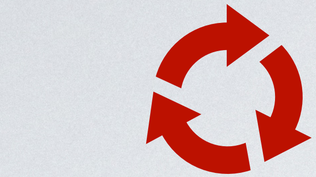


```php
$nb = count($functions);
$res = [];
$wrapper =
  function(int $idx, callable $function)
    use(&$res, &$nb, $s) {

    $res[$idx] = $function();
    if(--$nb === 0) {
        $s->resume($res);
    }
  }
;
```

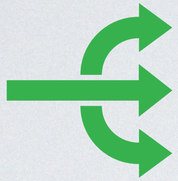


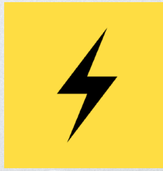
```
$r = all(
      fn() => getNumber(1),
      fn() => getNumber(3),
      fn() => getNumber(2),
);
$s = array_sum($r);
```





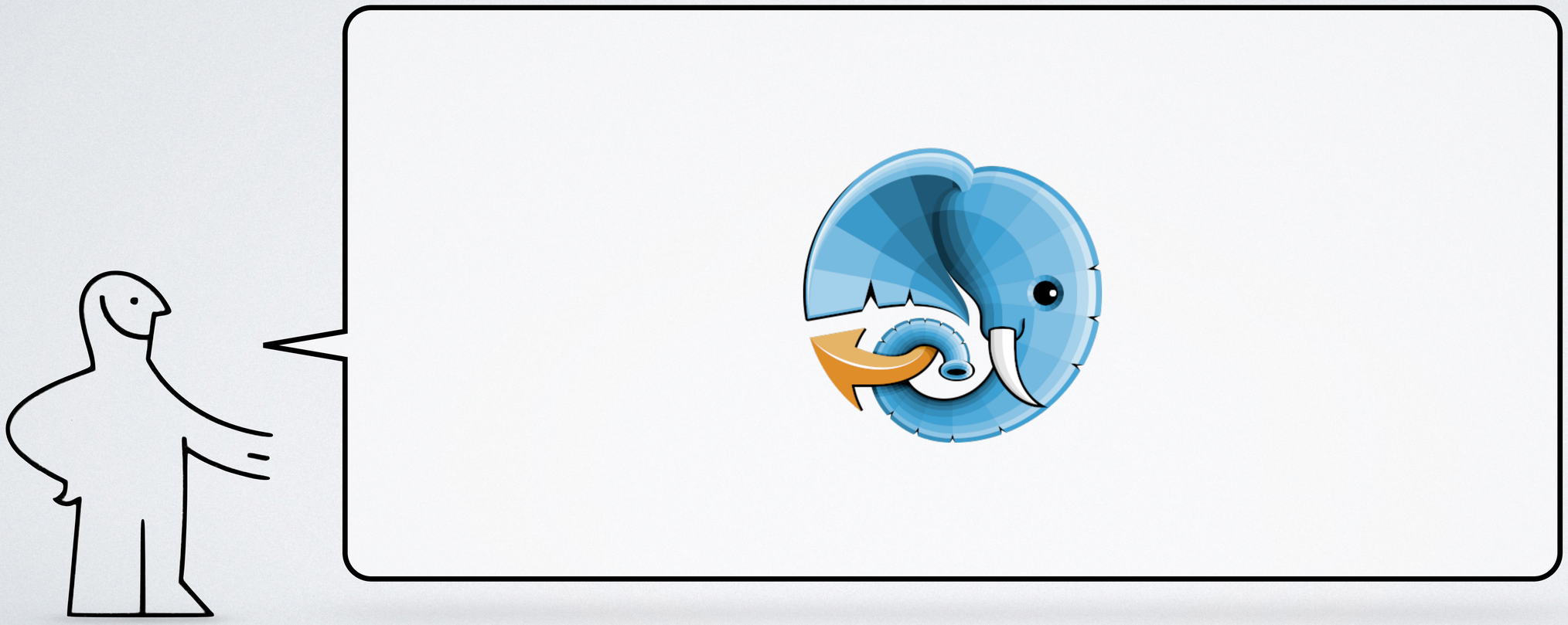
```
$r = all(
    fn() => getNumber(1),
    fn() => getNumber(3),
    fn() => getNumber(2),
);
$s = array_sum($r);
```





Example:

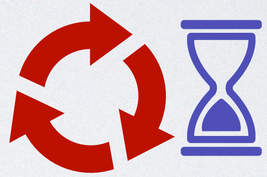
Http



```
composer require amphp/http-client
```

```php
$client =
    HttpClientBuilder::buildDefault();

$response = $client->request(
    new Request(
        "https://httpbin.org/delay/2"
    )
);
```

```php
$client =
    HttpClientBuilder::buildDefault();

$response = $client->request(
    new Request(
        "https://httpbin.org/delay/2"
    )
);
```

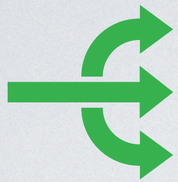


```php
echo "Start\n";
$r = Amp\Future\await([
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/2"
        ))->getStatus()
    ),
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/1"
        ))->getStatus()
    ),
]);
var_dump($r);  echo "Stop\n";
```

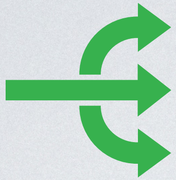
```php
echo "Start\n";
$r = Amp\Future\await([
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/2"
        ))->getStatus()
    ),
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/1"
        ))->getStatus()
    ),
]);
var_dump($r);  echo "Stop\n";
```

```php
echo "Start\n";
$r = Amp\Future\await([
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/2"
        ))->getStatus()
    ),
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/1"
        ))->getStatus()
    ),
]);
var_dump($r);  echo "Stop\n";
```

```php
echo "Start\n";
$r = Amp\Future\await([
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/2"
        ))->getStatus()
    ),
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/1"
        ))->getStatus()
    ),
]);
var_dump($r);  echo "Stop\n";
```



```php
echo "Start\n";
$r = Amp\Future\await([
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/2"
        ))->getStatus()
    ),
    Amp\async(fn() =>
        $client->request(new Request(
            "https://httpbin.org/delay/1"
        ))->getStatus()
    ),
]);
var_dump($r);  echo "Stop\n";
```



Example:

Http

```
composer require react/http
```

```php
$client = new React\Http\Browser();

$response = React\Async\await(
    $client->get(
        "https://httpbin.org/delay/2"
    )
);
```

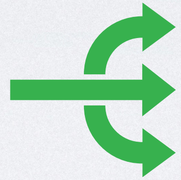
```php
$client = new React\Http\Browser();

$response = React\Async\await(
    $client->get(
        "https://httpbin.org/delay/2"
    )
);
```

```php
echo "Start\n";
$r = React\Async\await(\React\Promise\all([
  Async\async(fn() => React\Async\await(
      $client->get("https://httpbin.org/delay/2")
    )->getStatusCode()
  )(),
  Async\async(fn() => React\Async\await(
      $client->get("https://httpbin.org/delay/1")
    )->getStatusCode()
  )(),
]));
var_dump($r); echo "Stop\n";
```

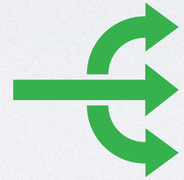
```php
echo "Start\n";
$r = React\Async\await(\React\Promise\all([
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/2")
    )->getStatusCode()
    )(),
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/1")
    )->getStatusCode()
    )(),
]));
var_dump($r); echo "Stop\n";
```

```php
echo "Start\n";
$r = React\Async\await(\React\Promise\all([
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/2")
    )->getStatusCode()
    )(),
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/1")
    )->getStatusCode()
    )(),
]));
var_dump($r); echo "Stop\n";
```

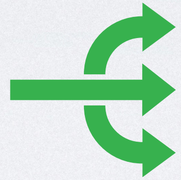
```php
echo "Start\n";
$r = React\Async\await(\React\Promise\all([
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/2")
    )->getStatusCode()
    )(),
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/1")
    )->getStatusCode()
    )(),
]));
var_dump($r); echo "Stop\n";
```



```php
echo "Start\n";
$r = React\Async\await(\React\Promise\all([
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/2")
    )->getStatusCode()
    )(),
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/1")
    )->getStatusCode()
    )(),
]));
var_dump($r); echo "Stop\n";
```

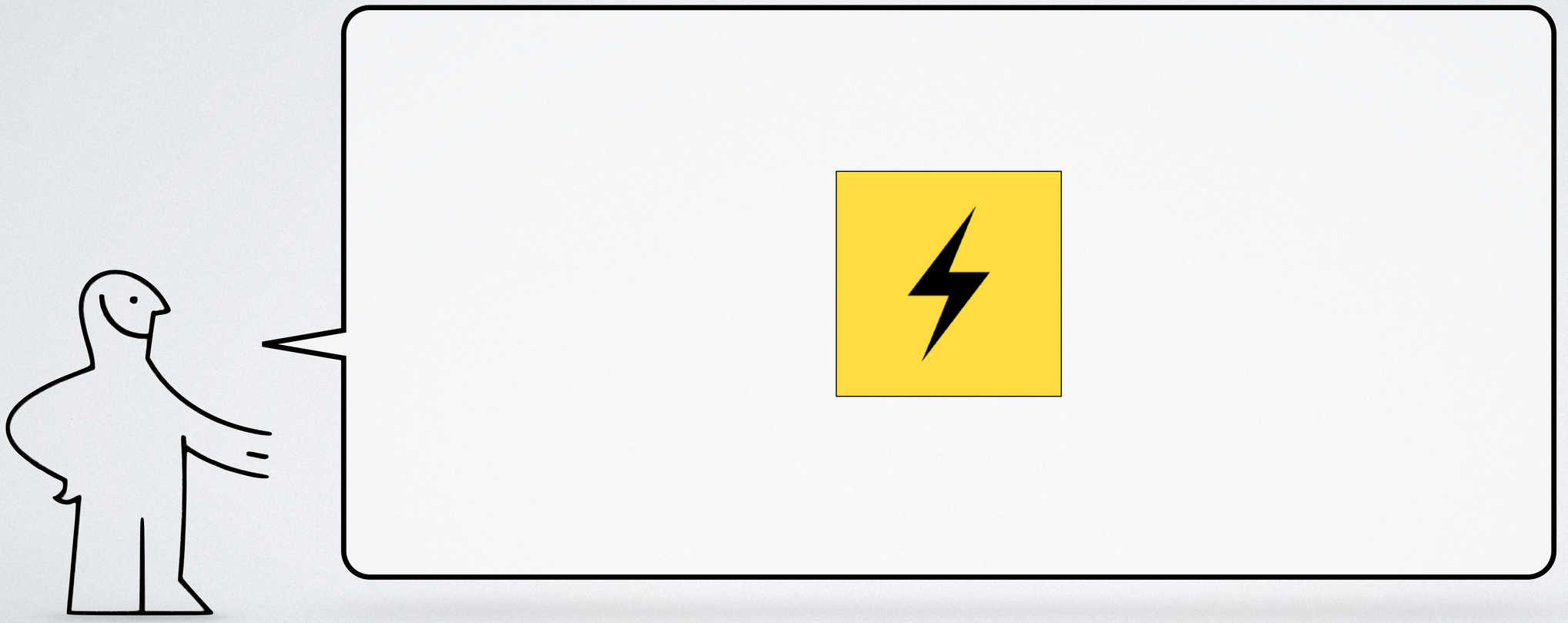


```php
echo "Start\n";
$r = React\Async\await(\React\Promise\all([
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/2")
    )->getStatusCode()
    )(),
    Async\async(fn() => React\Async\await(
        $client->get("https://httpbin.org/delay/1")
    )->getStatusCode()
    )(),
]));
var_dump($r); echo "Stop\n";
```



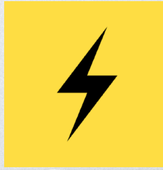
Example:

Http

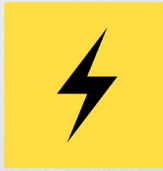


// .....

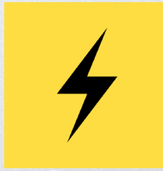
```php
function get(string $host): string
{
    $stream = \stream_socket_client(
        "tcp://$host:80"
    );
    \stream_set_blocking($stream, false);
    \fwrite(
        $stream,
        "GET / HTTP/1.1\r\nHost: $host\r\n"
        ."Connection: close\r\n\r\n"
    );
```



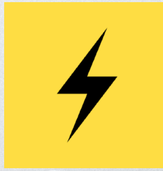
```
    "GET / HTTP/1.1\r\nHost: $host\r\n"
    ."Connection: close\r\n\r\n"
);



$s = Revolt\EventLoop::getSuspension();
$content = '';



EventLoop::onReadable(
    $stream,
    function ($watcher, $stream)
```



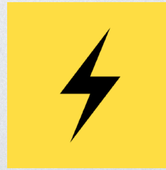
```
EventLoop::onReadable(
    $stream,
    function ($watcher, $stream)
    use($s, &$content) {

        $chunk = \fread($stream, 64 * 1024);
        if ($chunk === '') {
            EventLoop::cancel($watcher);
            \fclose($stream);
            $s->resume($content);
            return;
        }
```



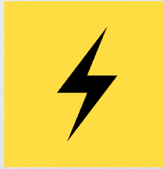
```
$stream,
function ($watcher, $stream)
use($s, &$content) {

  $chunk = \fread($stream, 64 * 1024);
  if ($chunk === '') {
    EventLoop::cancel($watcher);
    \fclose($stream);
    $s->resume($content);
    return;
  }

  $content .= $chunk;
}
);
```



```
        if ($chunk === '') {
            EventLoop::cancel($watcher);
            \fclose($stream);
            $s->resume($content);
            return;
        }

        $content .= $chunk;
    }
);

return $s->suspend();
}
```

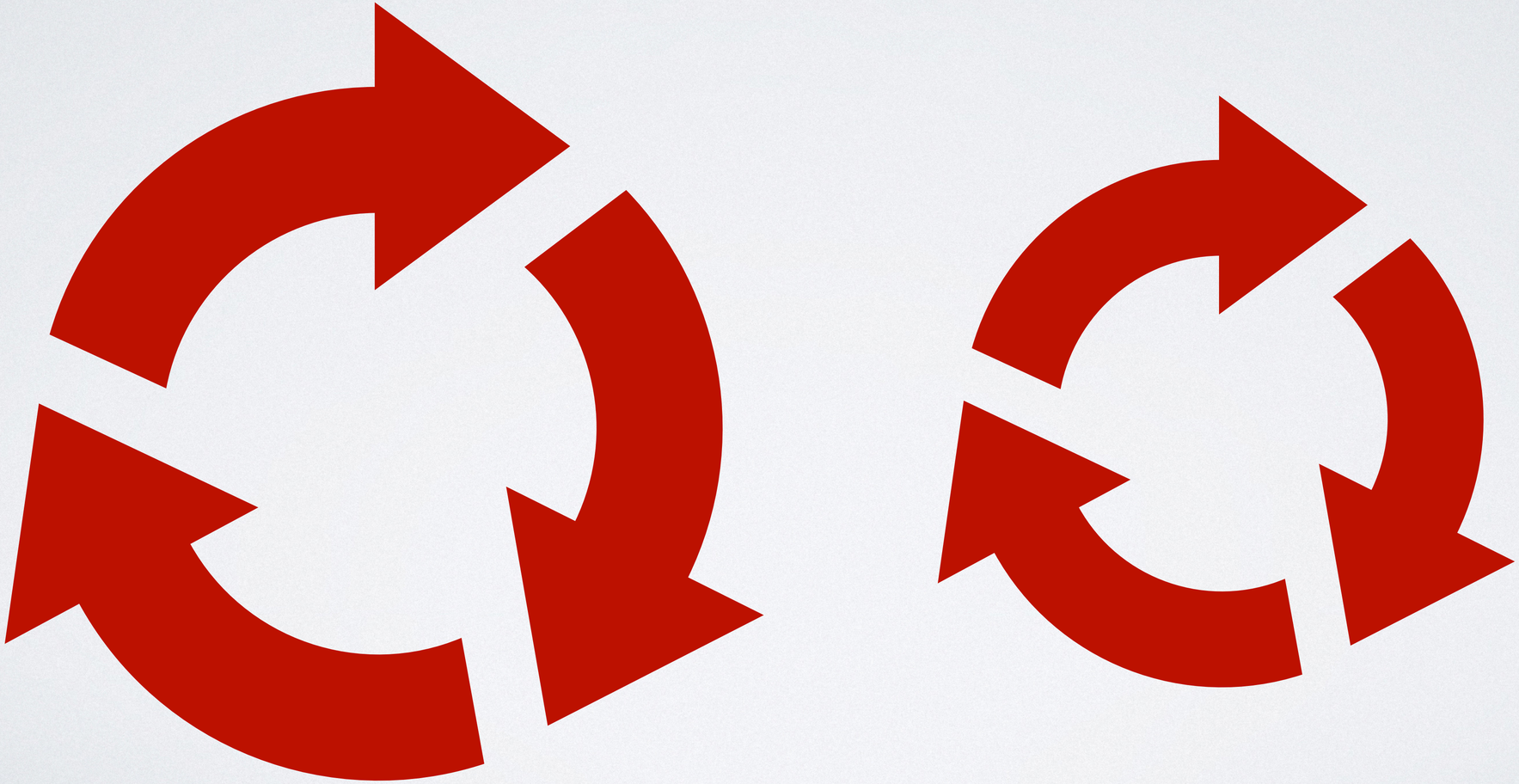


```php
$c = get('httpbin.org');

var_dump($c);
```



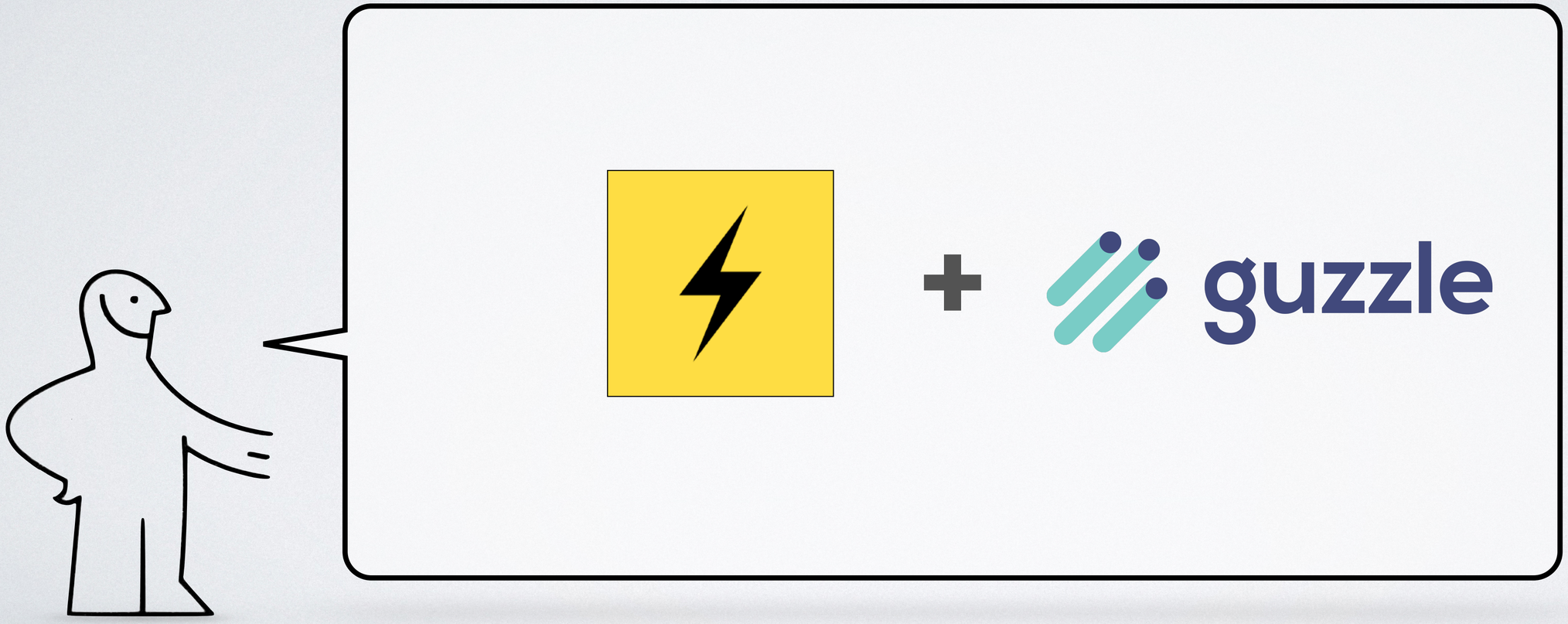
Interoperability

Managing 2 Event Loops

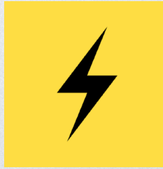


Example:

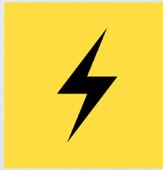
Guzzle



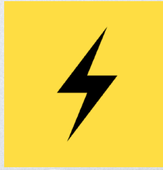
```php
$curlMultiHandler =
    new GuzzleHttp\Handler\CurlMultiHandler([
        'select_timeout' => 0,
    ]);

$stack = GuzzleHttp\HandlerStack::create(
    $curlMultiHandler
);

$client =
    new GuzzleHttp\Client([
        'handler' => $stack,
    ]);
```



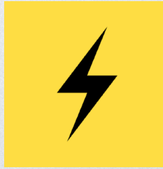
```php
$curlMultiHandler =
    new GuzzleHttp\Handler\CurlMultiHandler([
        'select_timeout' => 0,
    ]);

$stack = GuzzleHttp\HandlerStack::create(
    $curlMultiHandler
);

$client =
    new GuzzleHttp\Client([
        'handler' => $stack,
    ]);
```



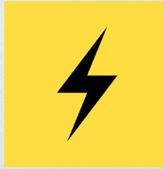
```
function guzzleLoop(
    CurlMultiHandler $curlMultiHandler
) {
    while(true) {
        $curlMultiHandler->tick();
        myDelay(0);
    }
}
```



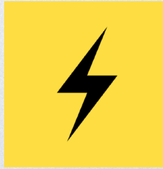
```php
function guzzleLoop(
    CurlMultiHandler $curlMultiHandler
) {
    while(true) {
        $curlMultiHandler->tick();
        myDelay(0);
    }
}
```



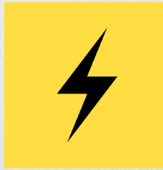
```
function guzzleLoop(
    CurlMultiHandler $curlMultiHandler
) {
    while(true) {
        $curlMultiHandler->tick();
        myDelay(0);
    }
}
```



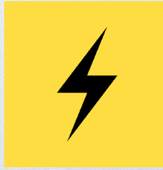
```
function timerLoop() {
    $start = microtime(true);
    while(true) {
        printf(
            "Elapsed: %f\r",
            microtime(true) - $start
        );
        myDelay(0.1);
    }
}
```



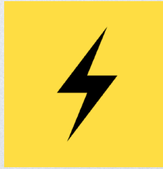
```php
function timerLoop() {
    $start = microtime(true);
    while(true) {
        printf(
            "Elapsed: %f\r",
            microtime(true) - $start
        );
        myDelay(0.1);
    }
}
```



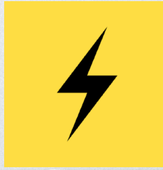
```
Revolt\EventLoop::queue(
    fn() => guzzleLoop($curlMultiHandler)
);

Revolt\EventLoop::queue(timerLoop(...));
```



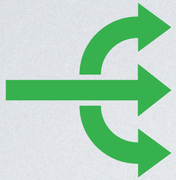
```
Revolt\EventLoop::queue(
    fn() => guzzleLoop($curlMultiHandler)
);

Revolt\EventLoop::queue(timerLoop(...));
```

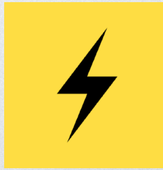


```
GuzzleHttp\Promise\Utils::all([
    $client->getAsync(
        "https://httpbin.org/delay/2"
    )->then(function() {echo "Finished 2 \n »;}),

    $client->getAsync(
        "https://httpbin.org/delay/1"
    )->then(function() {echo "Finished 1 \n";}),
])
->then(function() {
    echo "All done\n";
    EventLoop::getDriver()->stop();
});
```

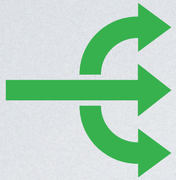


```
GuzzleHttp\Promise\Utils::all([
    $client->getAsync(
        "https://httpbin.org/delay/2"
    )->then(function() {echo "Finished 2 \n »;}),

    $client->getAsync(
        "https://httpbin.org/delay/1"
    )->then(function() {echo "Finished 1 \n";}),
])
->then(function() {
    echo "All done\n";
    EventLoop::getDriver()->stop();
});
```

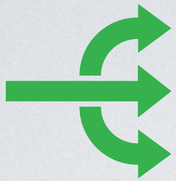


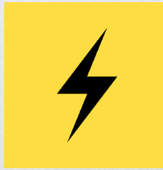


```
GuzzleHttp\Promise\Utils::all([
    $client->getAsync(
        "https://httpbin.org/delay/2"
    )->then(function() {echo "Finished 2 \n »;}),

    $client->getAsync(
        "https://httpbin.org/delay/1"
    )->then(function() {echo "Finished 1 \n";}),
])
->then(function() {
    echo "All done\n";
    EventLoop::getDriver()->stop();
});
```

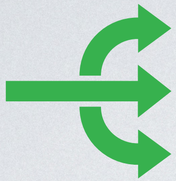


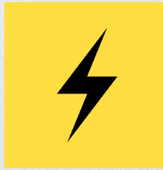
```
GuzzleHttp\Promise\Utils::all([
    $client->getAsync(
        "https://httpbin.org/delay/2"
    )->then(function() {echo "Finished 2 \n »;}),

    $client->getAsync(
        "https://httpbin.org/delay/1"
    )->then(function() {echo "Finished 1 \n";}),
])
->then(function() {
    echo "All done\n";
    EventLoop::getDriver()->stop();
});
```





```
GuzzleHttp\Promise\Utils::all([
    $client->getAsync(
        "https://httpbin.org/delay/2"
    )->then(function() {echo "Finished 2 \n »;}),

    $client->getAsync(
        "https://httpbin.org/delay/1"
    )->then(function() {echo "Finished 1 \n";}),
])
->then(function() {
    echo "All done\n";
    EventLoop::getDriver()->stop();
});
```





```php
echo "Start\n";

EventLoop::run();

echo "Stop\n";
```

```php
echo "Start\n";

EventLoop::run(); 

echo "Stop\n";
```

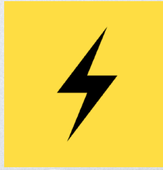
Trade Off

Managing single Event Loop

Example:

Singin'

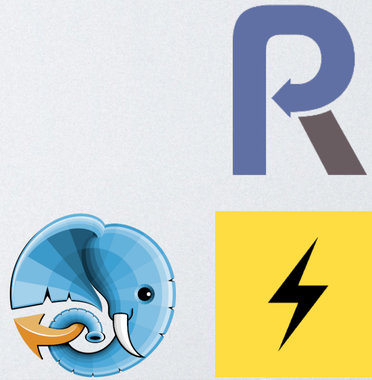
```
composer require revolt/event-loop-adapter-react
```



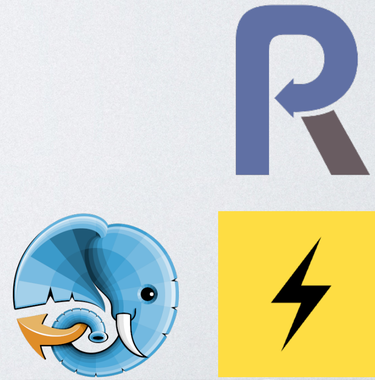
```php
function beat(): void
{
    for($i=0; $i<4; $i++) {
        echo "Boom💥\n"; Amp\delay(0.5);
        echo "Boom💥\n"; Amp\delay(0.5);
        echo "Clap👏\n"; Amp\delay(1);
    }
}
```

```
function sing(): void
{
    echo "\tWe\n";          Async\delay(1);
    echo "\tWill\n";        Async\delay(1);
    echo "\tWe\n";          Async\delay(1);
    echo "\tWill\n";        Async\delay(1);
    echo "\tRock\n";        Async\delay(0.5);
    echo "\tYou!\n";
}
```

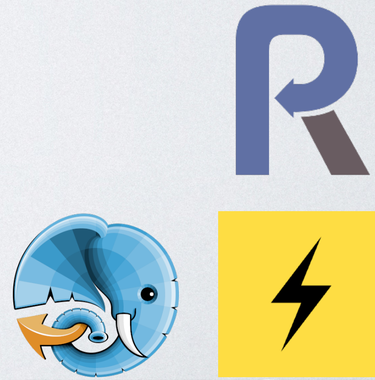
```php
Amp\async(beat(...));

echo "Start\n";
React\Async\async(sing(...))();

Revolt\EventLoop::run();
echo "Stop\n";
```



```php
Amp\async(beat(...));

echo "Start\n";
React\Async\async(sing(...))();

Revolt\EventLoop::run();
echo "Stop\n";
```



```php
Amp\async(beat(...));

echo "Start\n";
React\Async\async(sing(...))();

Revolt\EventLoop::run();
echo "Stop\n";
```

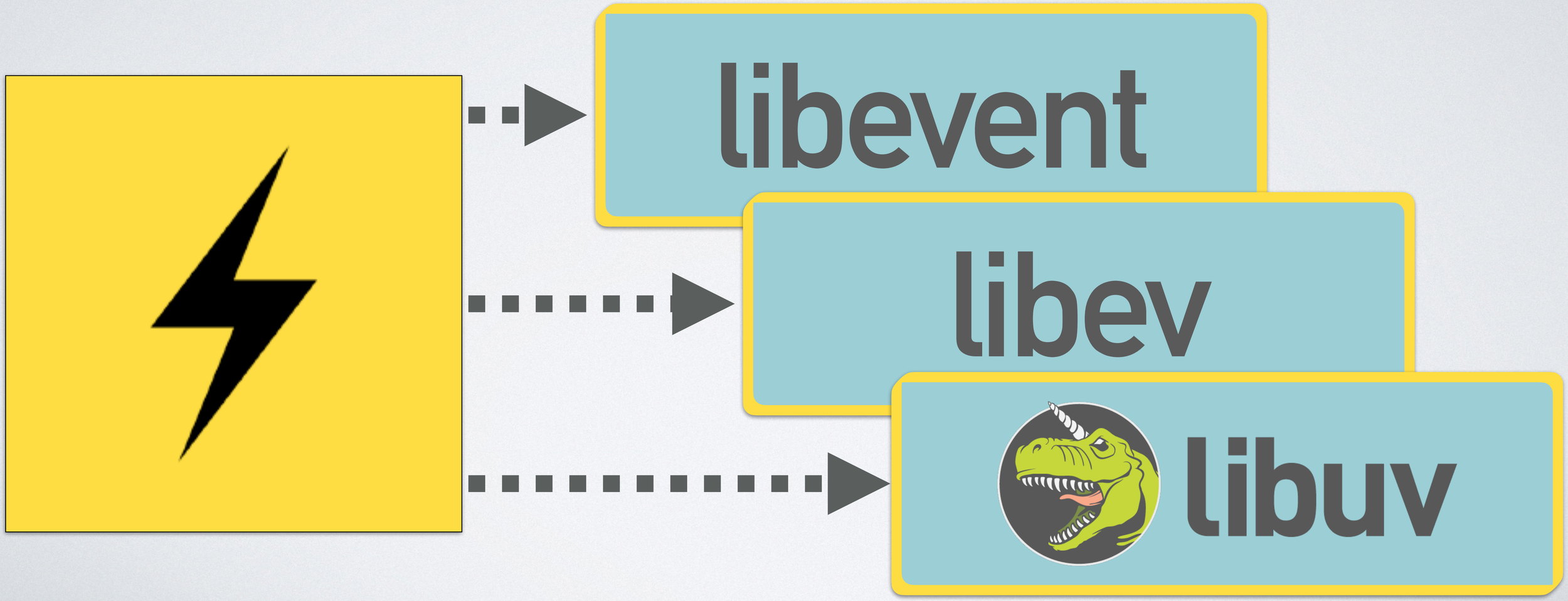


```php
Amp\async(beat(...));

echo "Start\n";
React\Async\async(sing(...))();

Revolt\EventLoop::run();
echo "Stop\n";
```

Symfony

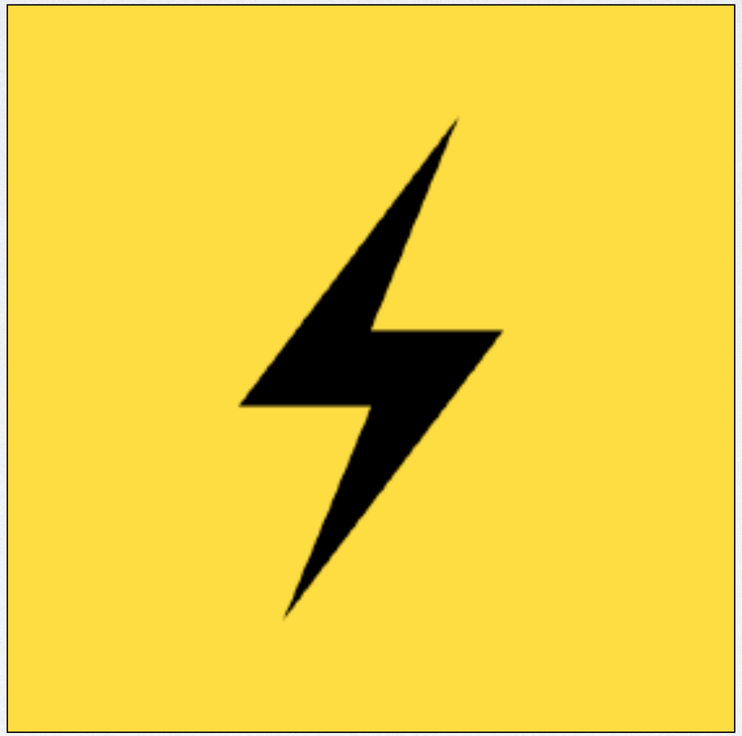
- HttpClient/**Amp**HttpClient (⚠️ v5)
- Symfony **Runtime**

Drivers



How to **choose**?

Small impact?



Big Project?

/

Benoit Viguier
@b_viguier@phpc.social

ReactPhp AmPhp RevoltPhp

Asynchronous Frameworks Comparison

Benoit Viguier
@b_viguier@phpc.social

FORUMPHP
PARIS2023