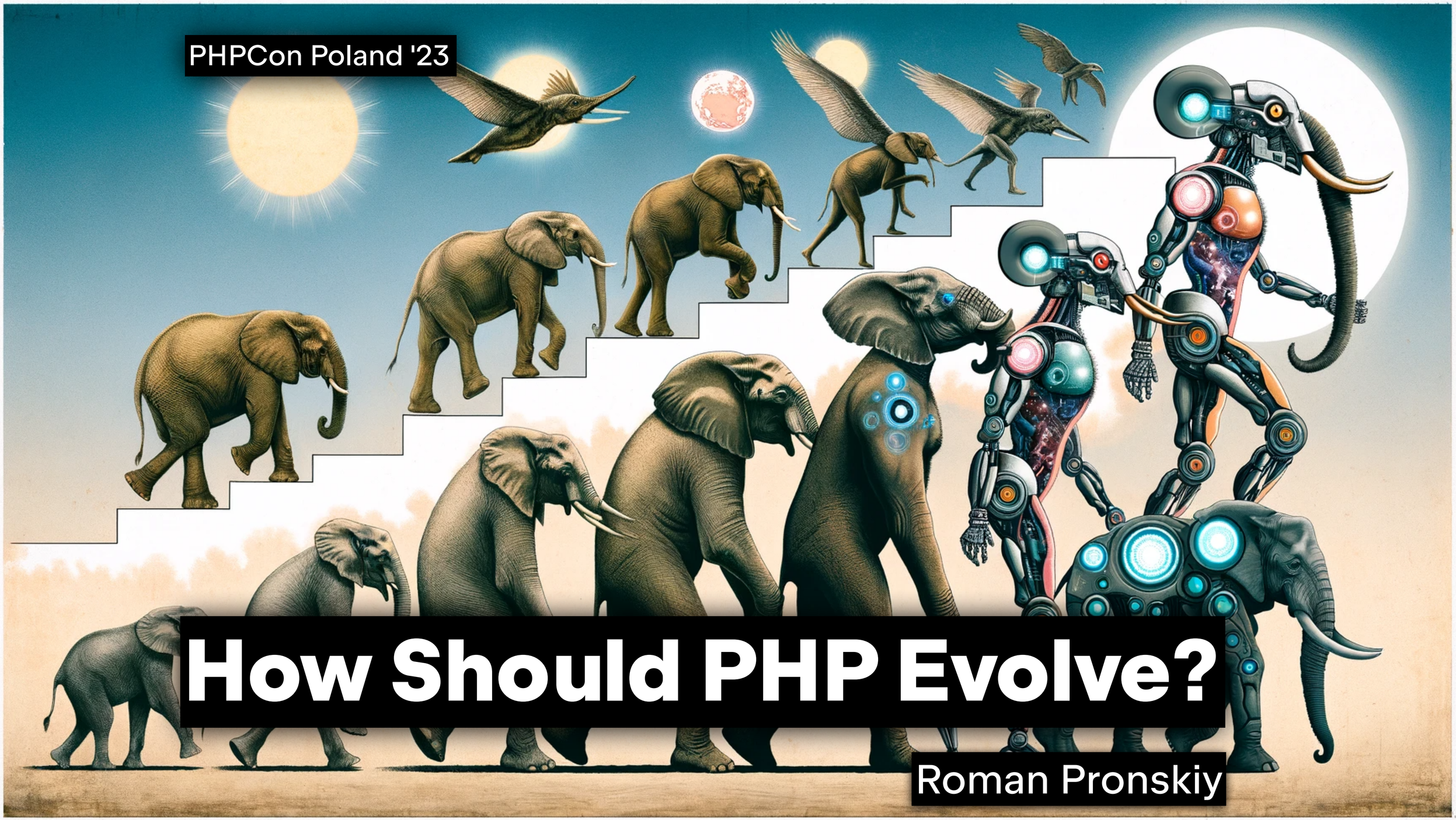
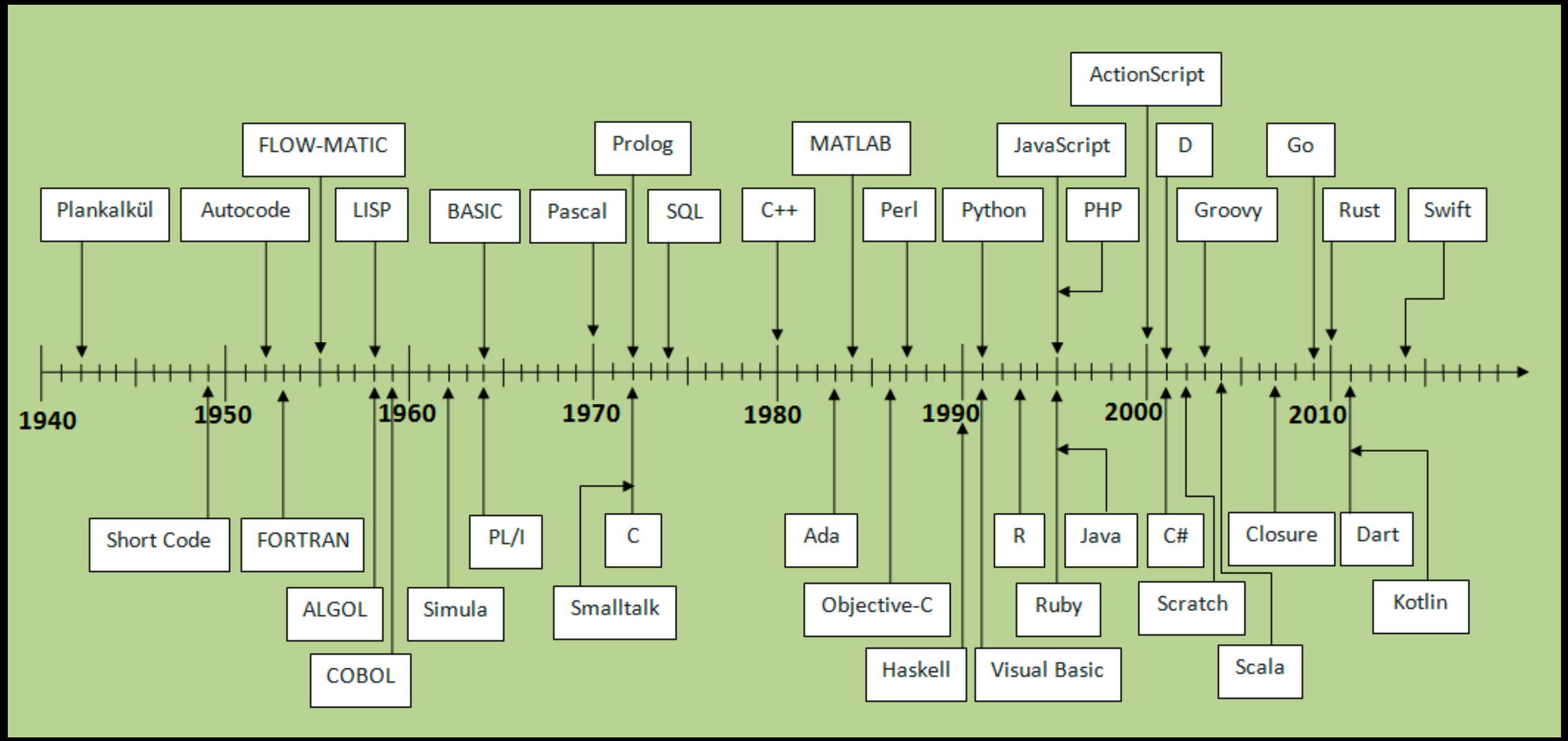
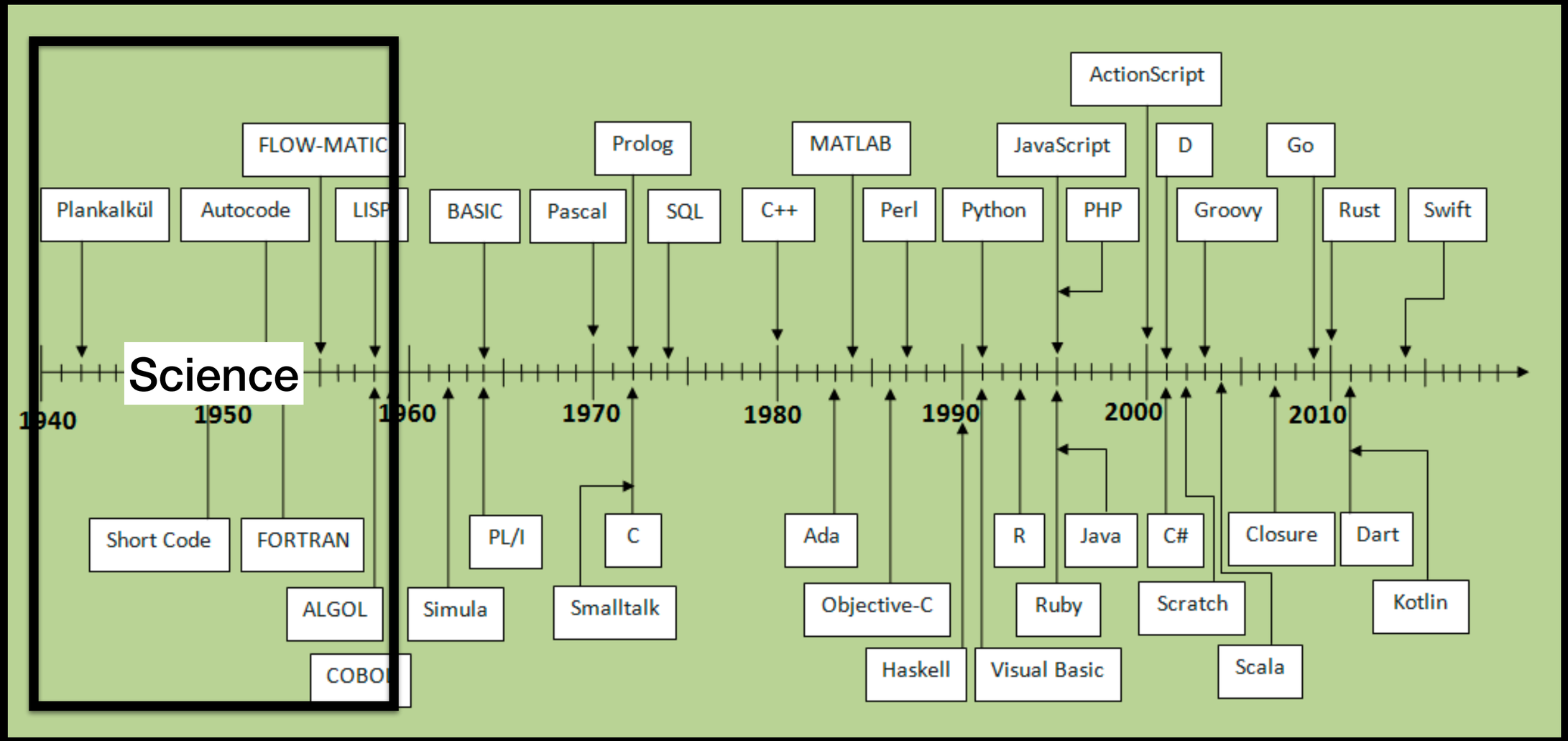


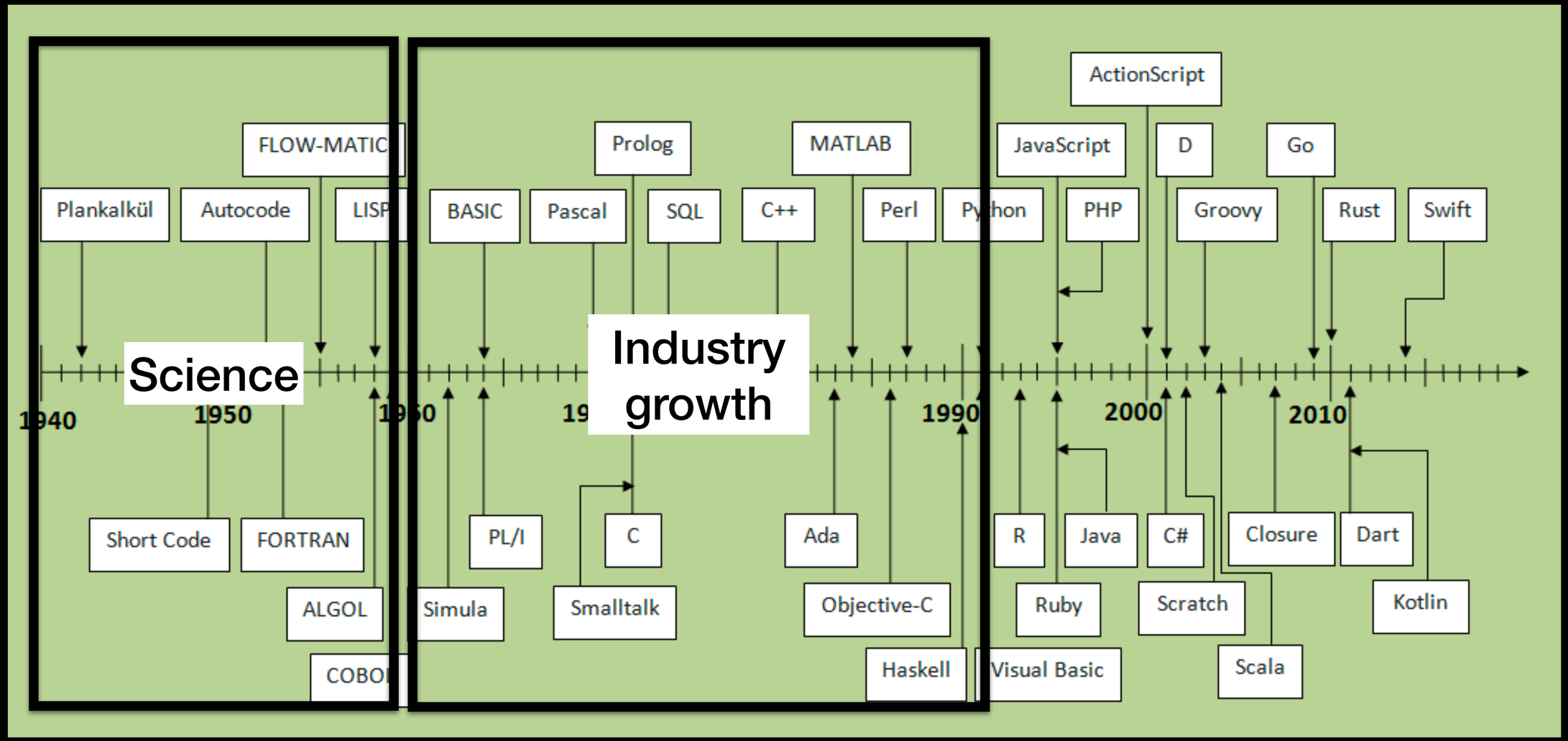
PHPCon Poland '23

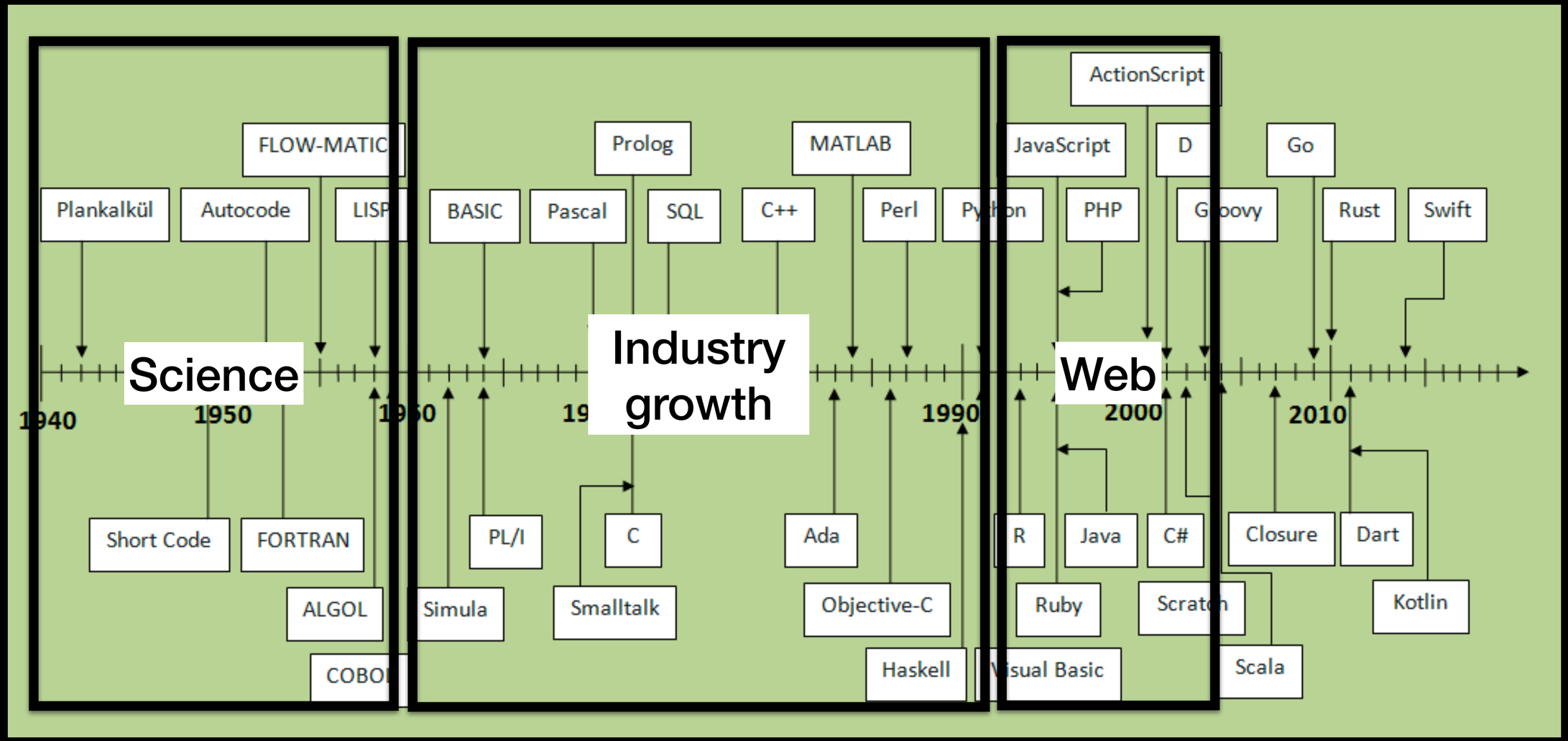
How Should PHP Evolve?

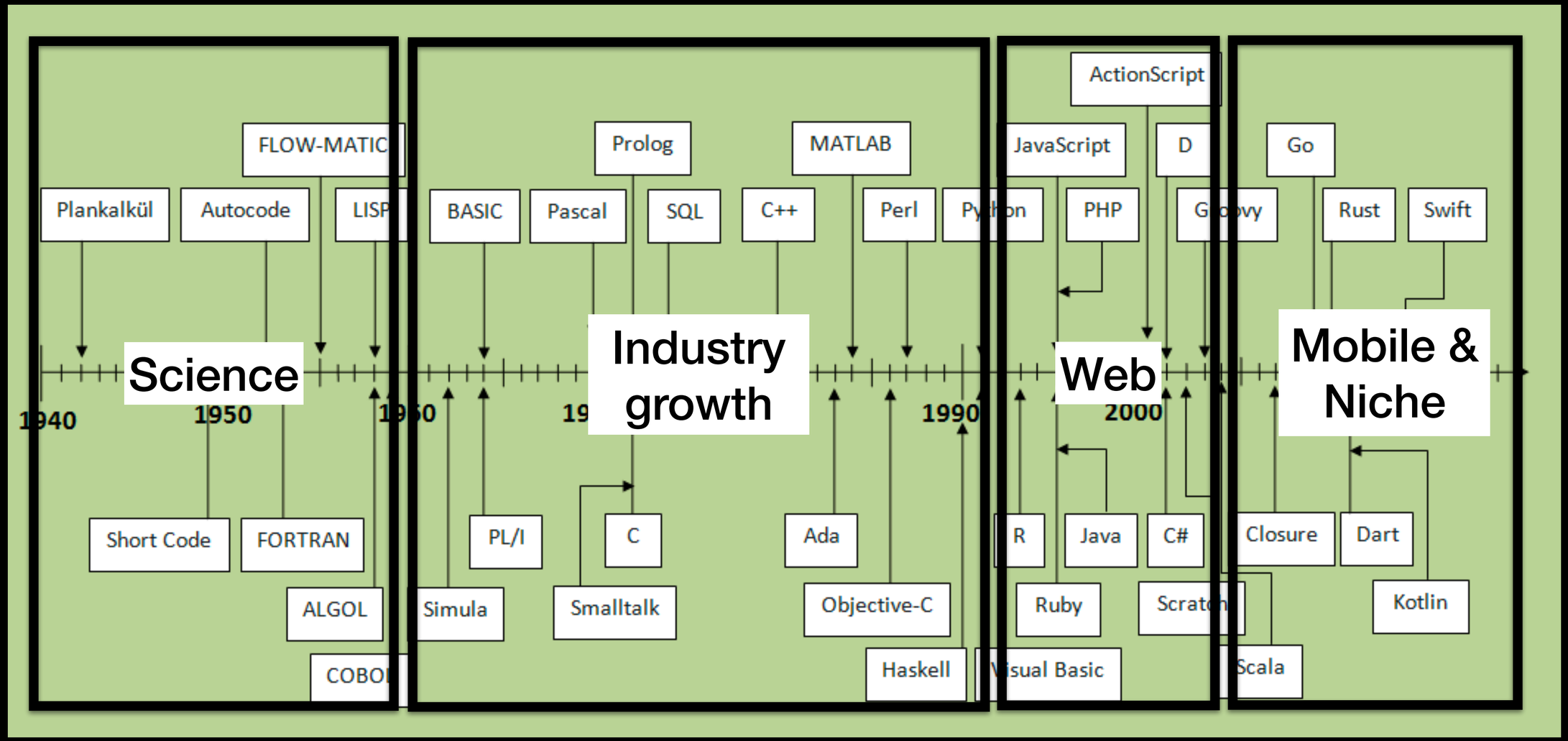
Roman Pronskiy

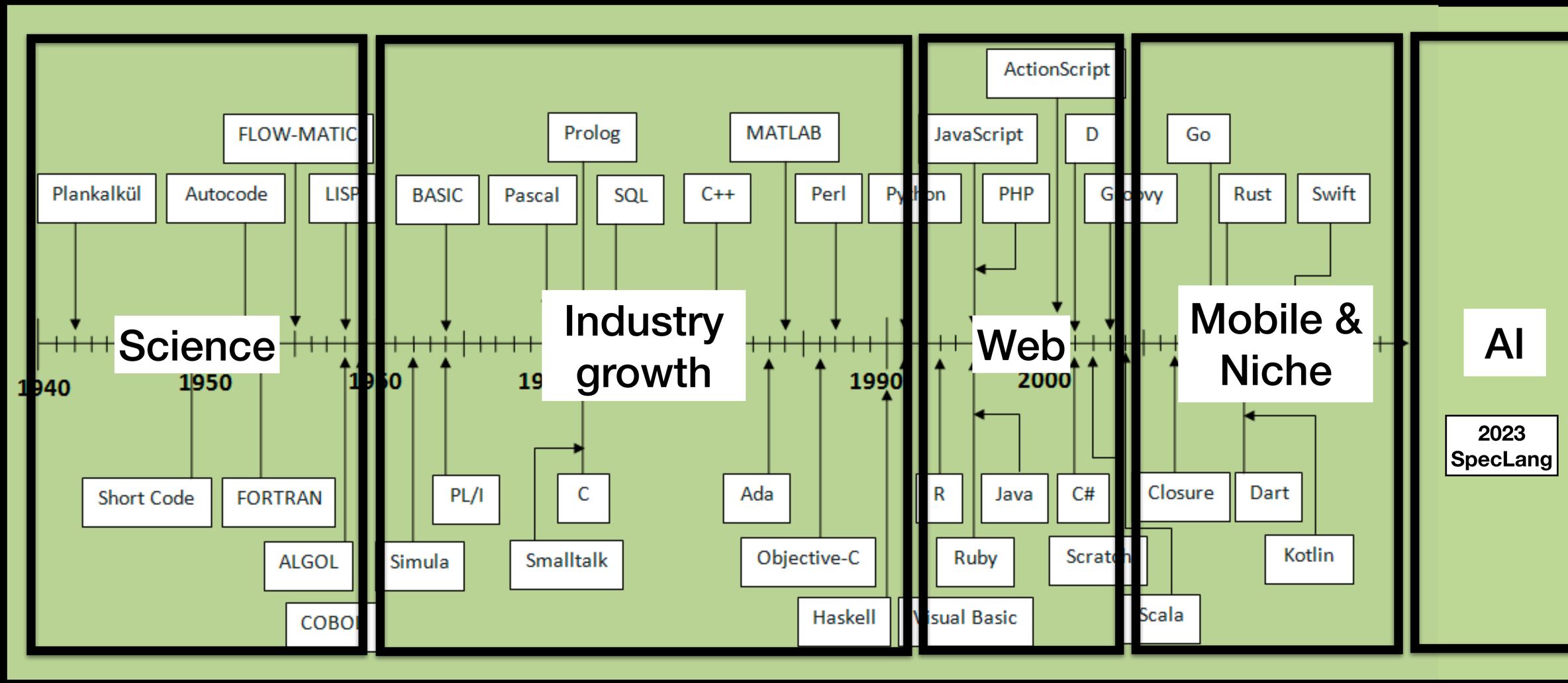

ActionScript
FLOW-MATIC
Prolog
MATLAB
JavaScript
D
Go
Plankalkül
Autocode
LISP
BASIC
Pascal
SQL
C++
Perl
Python
PHP
Groovy
Rust
Swift
1940
1950
1960
1970
1980
1990
2000
2010
Short Code
FORTRAN
PL/I
C
Ada
R
Java
C#
Closure
Dart
ALGOL
Simula
Smalltalk
Objective-C
Ruby
Scratch
Kotlin
COBOL
Haskell
Visual Basic
Scala

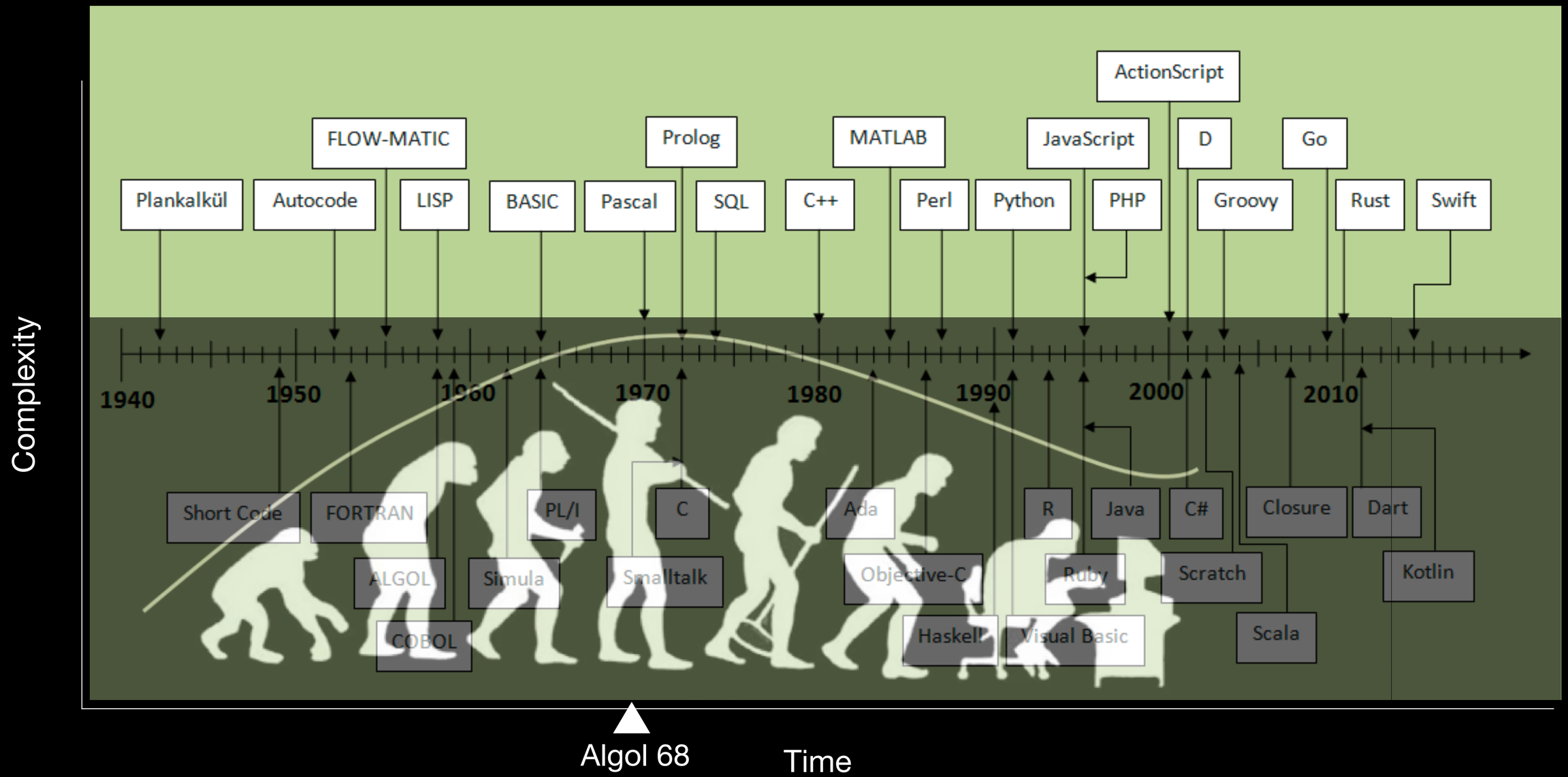

Science
1940
1950
1960
1970
1980
1990
2000
2010
Plankalkül
Autocode
FLOW-MATIC
LISP
BASIC
Pascal
Prolog
SQL
C++
MATLAB
Perl
Python
JavaScript
PHP
ActionScript
D
Groovy
Go
Rust
Swift
Short Code
FORTRAN
ALGOL
COBOL
Simula
PL/I
Smalltalk
C
Ada
Objective-C
Haskell
Visual Basic
R
Ruby
Java
C#
Scratch
Scala
Closure
Dart
Kotlin


Science
Industry growth
1940
1950
1960
1990
2000
2010
Plankalkül
Autocode
FLOW-MATIC
LISP
Short Code
FORTRAN
ALGOL
COBOL
BASIC
Pascal
SQL
Prolog
C++
MATLAB
Perl
Python
PL/I
Simula
C
Smalltalk
Ada
Objective-C
Haskell
JavaScript
PHP
ActionScript
D
Groovy
Go
Rust
Swift
R
Ruby
Visual Basic
Java
C#
Scratch
Closure
Scala
Dart
Kotlin


Science
Industry growth
Web
1940
1950
1960
1990
2000
2010
Plankalkül
Autocode
FLOW-MATIC
LISP
Short Code
FORTRAN
ALGOL
COBOL
BASIC
Pascal
Prolog
SQL
C++
MATLAB
Perl
PL/I
C
Simula
Smalltalk
Ada
Objective-C
Haskell
JavaScript
PHP
ActionScript
D
Go
Rust
Swift
R
Java
C#
Ruby
Closure
Dart
Kotlin
Scala


Science
Industry growth
Web
Mobile & Niche
1940
1950
1960
1990
2000
Plankalkül
Autocode
FLOW-MATIC
LISP
Short Code
FORTRAN
ALGOL
COBOL
BASIC
Pascal
Prolog
SQL
C++
MATLAB
Perl
Python
PL/I
Simula
C
Smalltalk
Ada
Objective-C
Haskell
JavaScript
ActionScript
PHP
D
Groovy
R
Java
C#
Ruby
Scratch
Visual Basic
Go
Rust
Swift
Closure
Dart
Kotlin
Scala


Science
Industry growth
Web
Mobile & Niche
AI
1940
1950
1960
1990
2000
2023 SpecLang
Plankalkül
Autocode
FLOW-MATIC
LISP
Short Code
FORTRAN
ALGOL
COBOL
BASIC
Pascal
Prolog
SQL
C++
MATLAB
Perl
PL/I
Simula
C
Smalltalk
Ada
Objective-C
Haskell
JavaScript
PHP
ActionScript
D
R
Java
C#
Ruby
Visual Basic
Go
Rust
Swift
Closure
Dart
Kotlin
Scala


Complexity
ActionScript
FLOW-MATIC
Prolog
MATLAB
JavaScript
D
Go
Plankalkül
Autocode
LISP
BASIC
Pascal
SQL
C++
Perl
Python
PHP
Groovy
Rust
Swift
1940
1950
1960
1970
1980
1990
2000
2010
Short Code
FORTRAN
PL/I
C
Ada
R
Java
C#
Closure
Dart
ALGOL
Simula
Smalltalk
Objective-C
Ruby
Scratch
Kotlin
COBOL
Haskell
Visual Basic
Scala
Algol 68
Time

Market demand

Market demand

PHP in 1995: a simple tool to build websites; by/for a person who has C background and does not like Perl.

Market demand

PHP in 1995: a simple tool to build websites; by/for a person who has C background and does not like Perl.

Product Management 101

Product Management 101

- Evaluating a product
- When to say no to new features
- Which new features to build
- Getting those features used

What features to add?

What features to add?

Simple and elegant

What features to add?

Simple and elegant

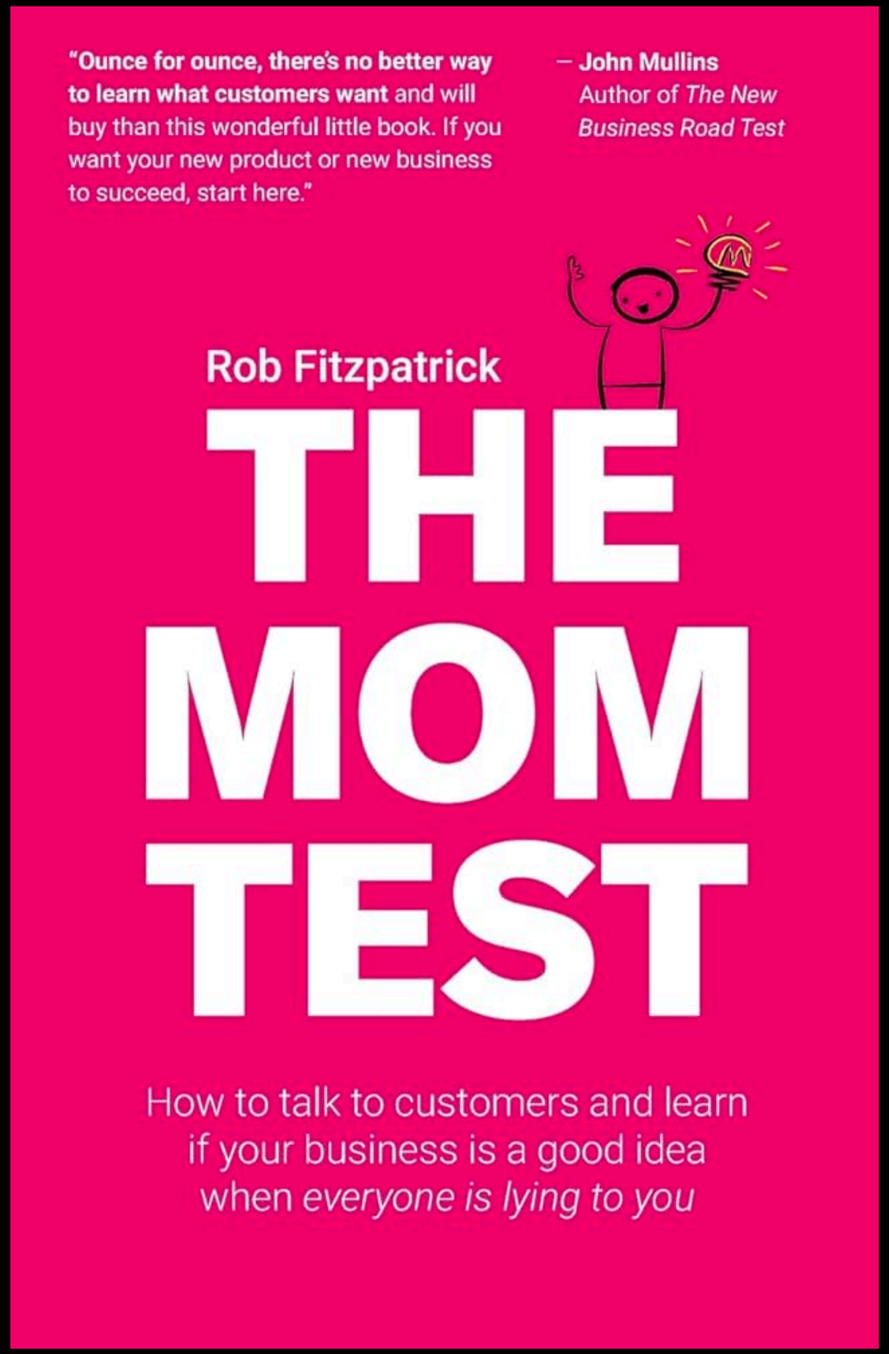
Complex and counterintuitive

At least it cannot die

Why not just ask users?

Why not just ask users?

- Encourages hypothetical answers
- Lack of self-awareness
- Social desirability bias
- Overemphasis on features over problems
- Risk of feature creep
- Difficulty in prioritising features
- Innovation limitation



Do you want generics in PHP?

Do you want generics in PHP? *

* But with a 10% performance penalty?

It's always a tradeoff

vs

It's always a tradeoff

Clever

Readable

vs

It's always a tradeoff

Clever

Readable

```
$numbers = [...];
echo array_reduce(
    array_map(fn($n) => $n * $n, $numbers),
    fn($carry, $item) => $carry + $item,
    initial: 0
);
```

vs

It's always a tradeoff

Clever

Readable

```php
$numbers = [...];
echo array_reduce(
    array_map(fn($n) => $n * $n, $numbers),
    fn($carry, $item) => $carry + $item,
    initial: 0
);
```

vs

```php
$numbers = [...];
$sum = 0;
foreach ($numbers as $number) {
    $sum += $number * $number;
}
echo $sum;
```

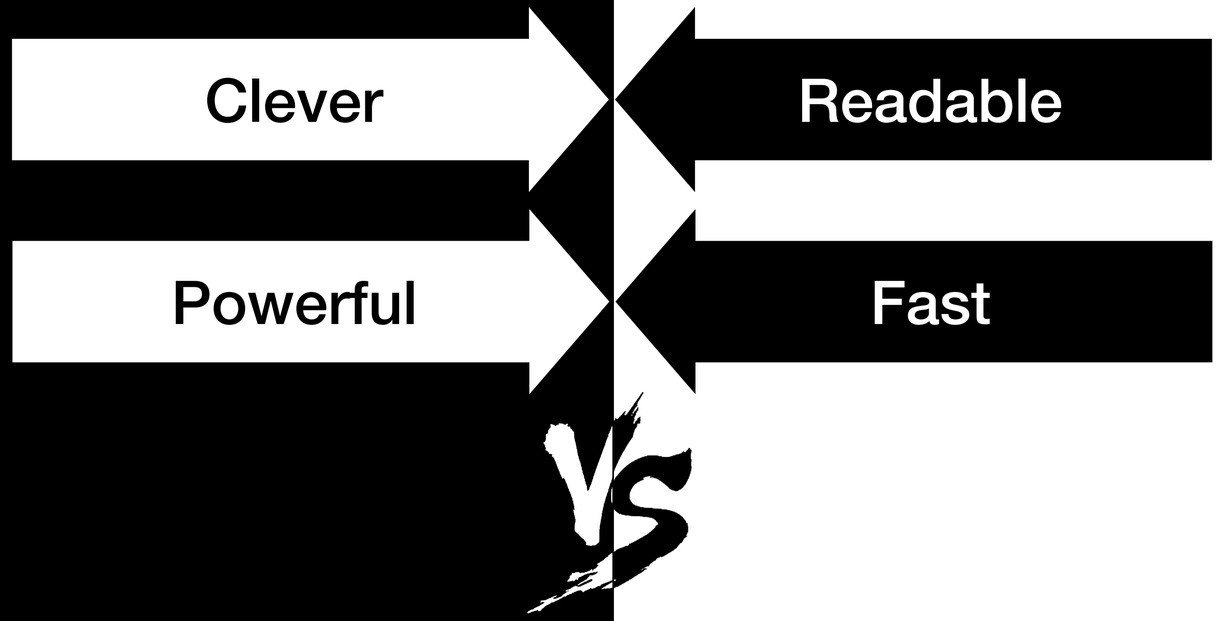
It's always a tradeoff

Clever

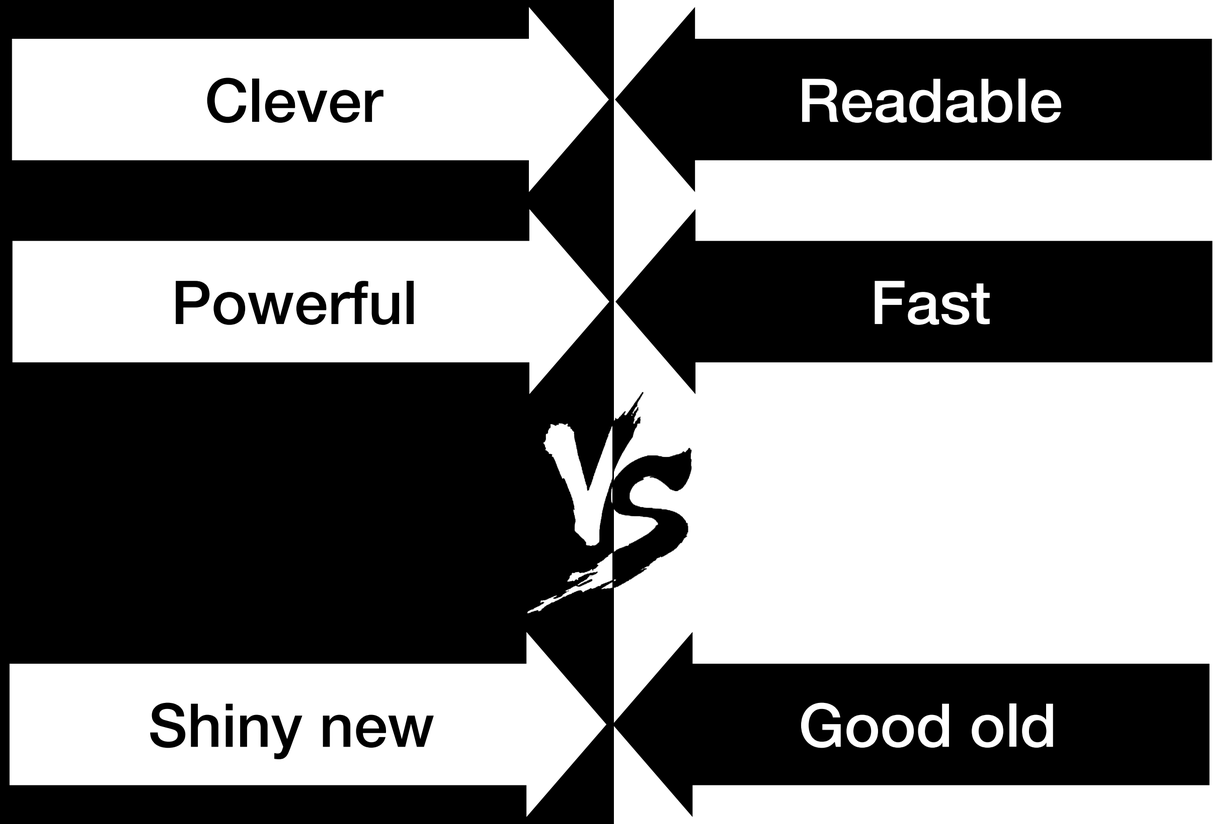
Readable

vs

It's always a tradeoff



It's always a tradeoff



It's always a tradeoff

```php
interface Logger {
    function log() { echo __METHOD__, "\n"; }
}
```

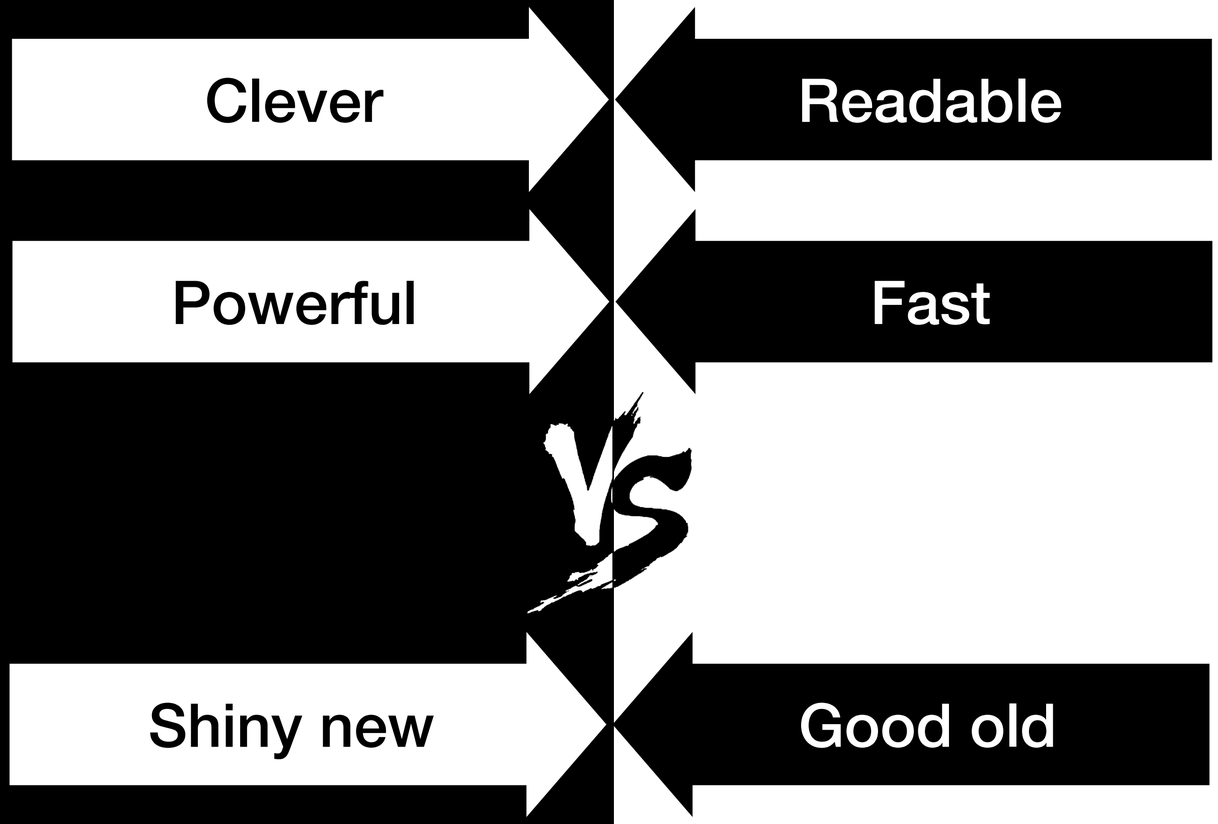
```php
interface Logger {
    function log();
}
trait LoggerTrait {
    function log() { echo __METHOD__, "\n"; }
}
```

VS

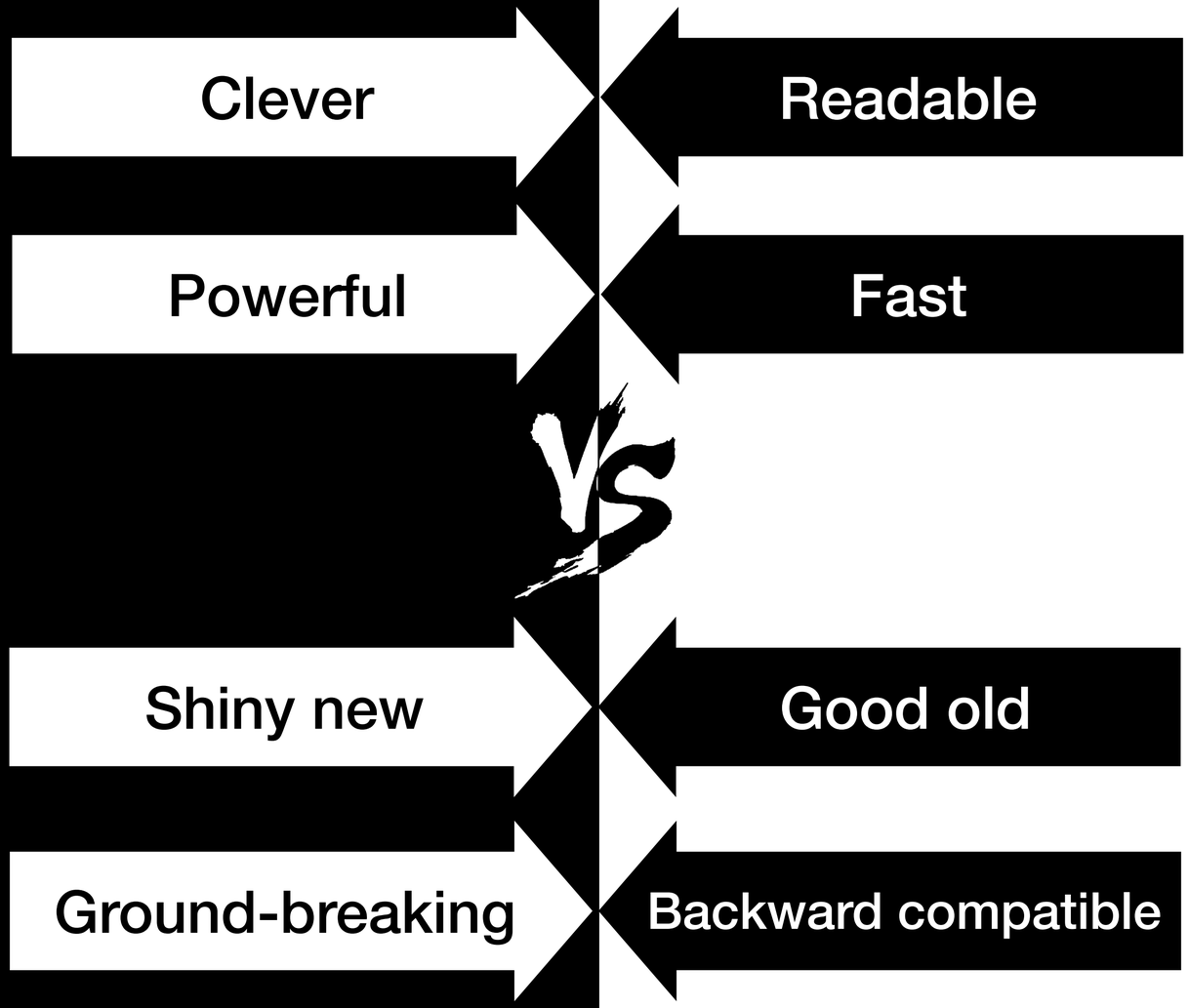
Shiny new

Good old

It's always a tradeoff



It's always a tradeoff



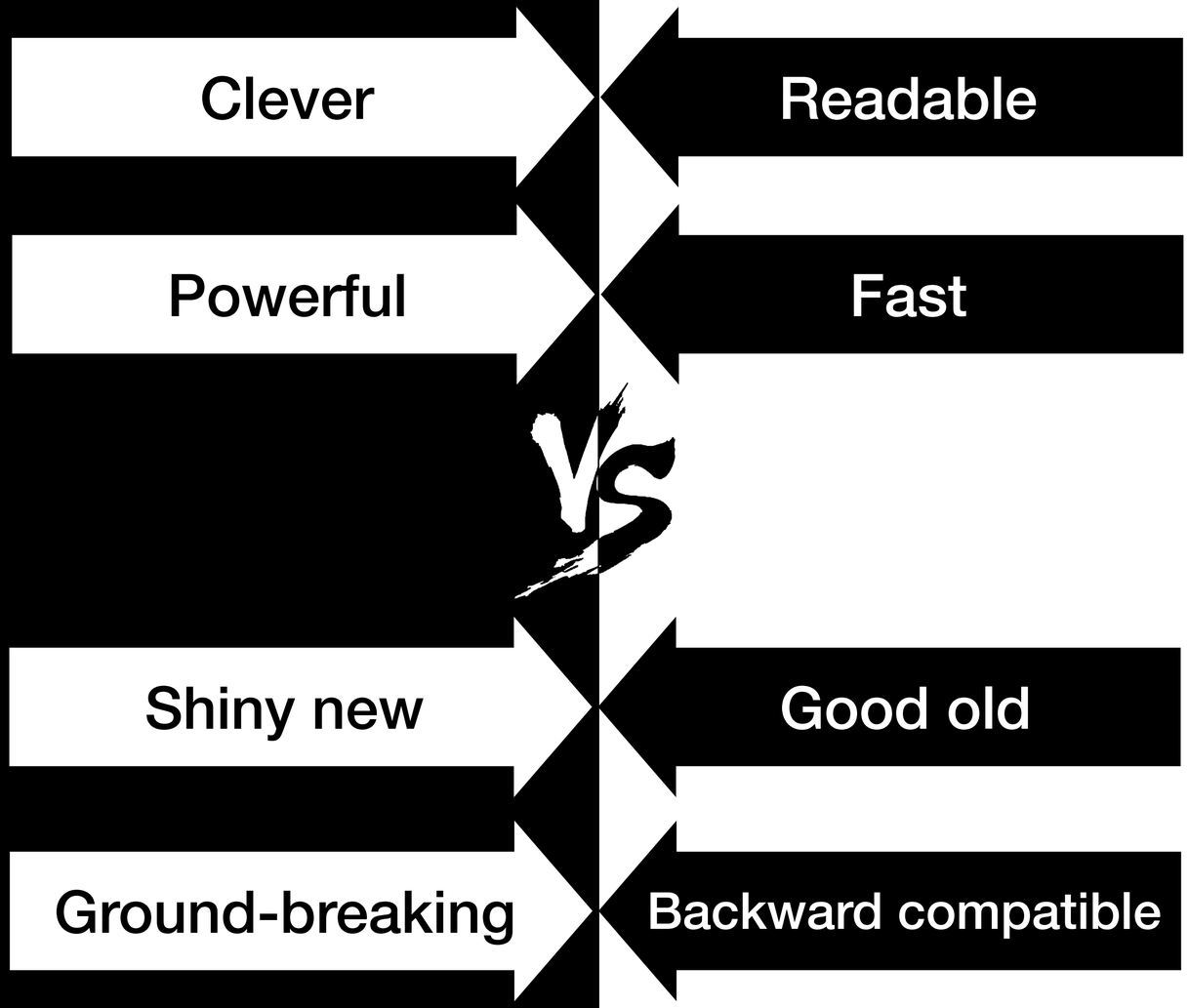
It's always a tradeoff

Clever	Readable
Powerful	Fast
Shiny new	Good old
Ground-breaking	Backward compatible

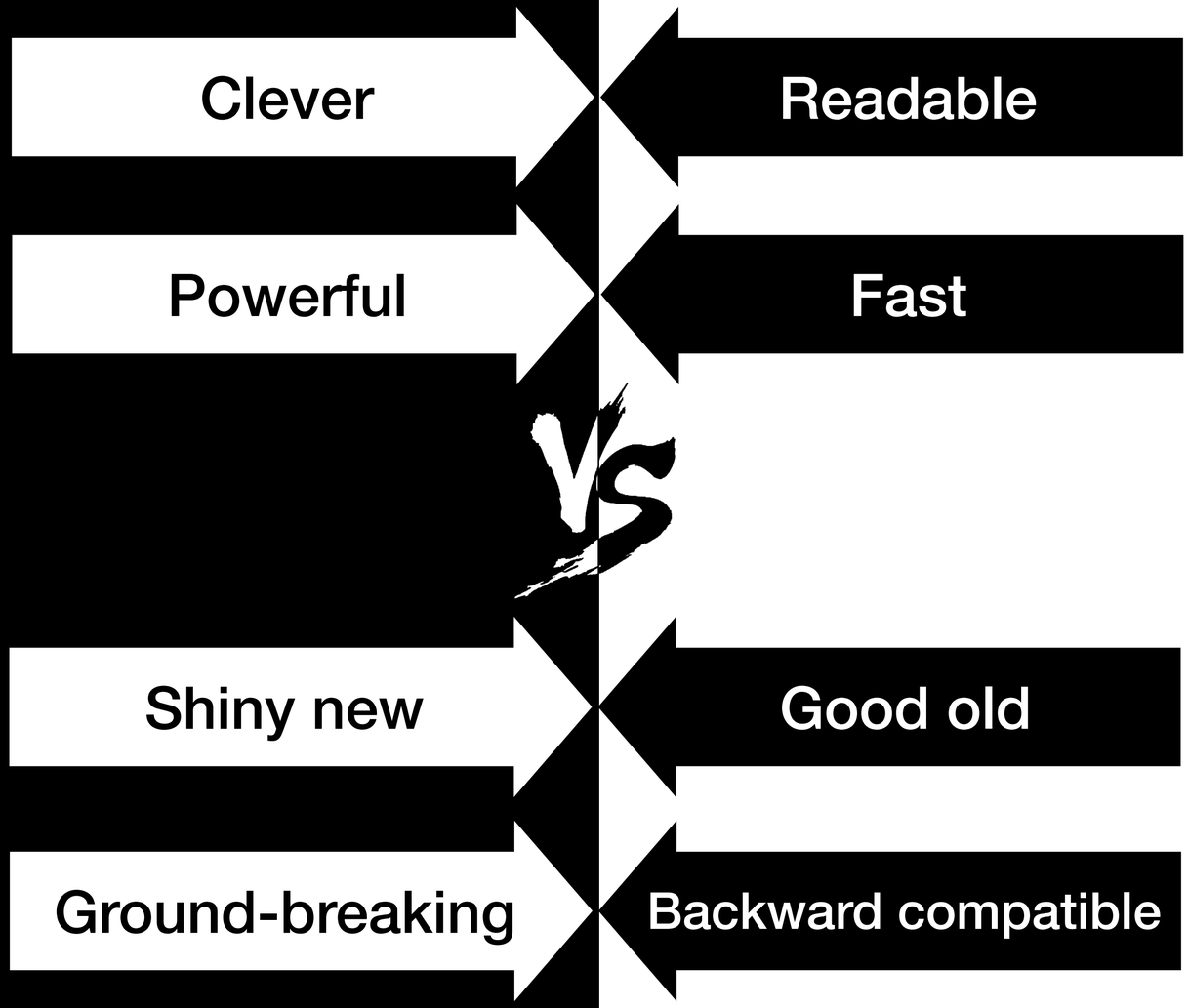
vs

Erased generics?

It's always a tradeoff



It's always a tradeoff



Applying PM tactics to PHP

1. Analyse competitors

Rust

Kotlin

Go

JavaScript

Rust

Rust

- Official toolchain management with `rustup`

Rust

- Official toolchain management with `rustup`

`rustc` — Compiler.
`cargo` — Package manager and build tool.
`clippy` — Lint tool that with extra checks.
`rustfmt` — Tool for formatting code.
`rust-docs` — Local copy of the documentation.
`rust-analyzer` — Language server for editors.

Rust

- Official toolchain management with `rustup`

Kotlin

Go

JavaScript

Go

Go

- Backward compatibility

There will not be a Go 2 that breaks Go 1 programs.

...prioritizing compatibility was the most important design decision we made...

Boring is good. Boring is stable. Boring means being able to focus on your work, not on what's different about Go.

Rust

- Official toolchain management with `rustup`

Kotlin

Go

- Backward compatibility

JavaScript

Kotlin

Kotlin

- Classes are final by default 😱

```
class Foo { }

class Bar(): Foo() { }
```

```
This type is final, so it cannot be inherited from
```

Kotlin

- Classes are final by default 😱
- Class extension (Laravel macro)

```
class Foo { }

fun Foo.myMethod() {
    println("Hello, PHPCon Poland")
}
```

Rust

- Official toolchain management with `rustup`

Kotlin

- Classes are final by default 😱
- Class extension (Laravel macro)

Go

- Backward compatibility

JavaScript

JavaScript

Node.js' stability index

Stability: 0 - Deprecated. The feature may emit warnings. Backward compatibility is not guaranteed.

Stability: 1 - Experimental. Non-backward compatible changes or removal may occur in any future release. Use of the feature is not recommended in production environments.

Stability: 2 - Stable. Compatibility with the Composer ecosystem is a high priority.

Stability: 3 - Legacy. Although this feature is unlikely to be removed, it is no longer actively maintained, and other alternatives are available.

JavaScript

- Experimental features

Rust

- Official toolchain management with `rustup`

Kotlin

- Classes are final by default 😱
- Class extension (Laravel macro)

Go

- Backward compatibility

JavaScript

- Experimental features

2. Measure: Telemetry

2. Measure: Telemetry

"If it's free then you're the product"

https://quoteinvestigator.com/2017/07/16/product/

Telemetry

Telemetry

Go (opt-in)

"I believe that open-source software projects need to explore new telemetry designs that help developers get the information they need to work efficiently and effectively, without collecting invasive traces of detailed user activity,"

https://research.swtch.com/telemetry-intro

Telemetry

Go (opt-in)

"I believe that open-source software projects need to explore new telemetry designs that help developers get the information they need to work efficiently and effectively, without collecting invasive traces of detailed user activity,"

https://research.swtch.com/telemetry-intro

Rust (removed in 2019)

https://github.com/rust-lang/rustup/pull/1642

Telemetry

Go (opt-in)

"I believe that open-source software projects need to explore new telemetry designs that help developers get the information they need to work efficiently and effectively, without collecting invasive traces of detailed user activity,"

https://research.swtch.com/telemetry-intro

Rust (removed in 2019)

https://github.com/rust-lang/rustup/pull/1642

Next.js (opt-out)

https://nextjs.org/telemetry

Telemetry

Go (opt-in)

"I believe that open-source software projects need to explore new telemetry designs that help developers get the information they need to work efficiently and effectively, without collecting invasive traces of detailed user activity,"

https://research.swtch.com/telemetry-intro

Rust (removed in 2019)

https://github.com/rust-lang/rustup/pull/1642

Next.js (opt-out)

https://nextjs.org/telemetry

PHP?

Telemetry

Go (opt-in)

"I believe that open-source software projects need to explore new telemetry designs that help developers get the information they need to work efficiently and effectively, without collecting invasive traces of detailed user activity,"

https://research.swtch.com/telemetry-intro

Rust (removed in 2019)

https://github.com/rust-lang/rustup/pull/1642

Next.js (opt-out)

https://nextjs.org/telemetry

PHP?

Composer plugin?

3. Usage statistics

How many projects use a certain new feature from PHP 8.*?

Regex search

https://sourcegraph.com/search

```
context:global file:\.php$ count:all repohascommitafter:01/08/2020 \#\[\\?AllowDynamicProperties\]
```

Aa .* []

6.1k results in 60.12s

Actions

Filters

WordPress/WordPress › wp-includes/class-wp.php ★ 18.2k

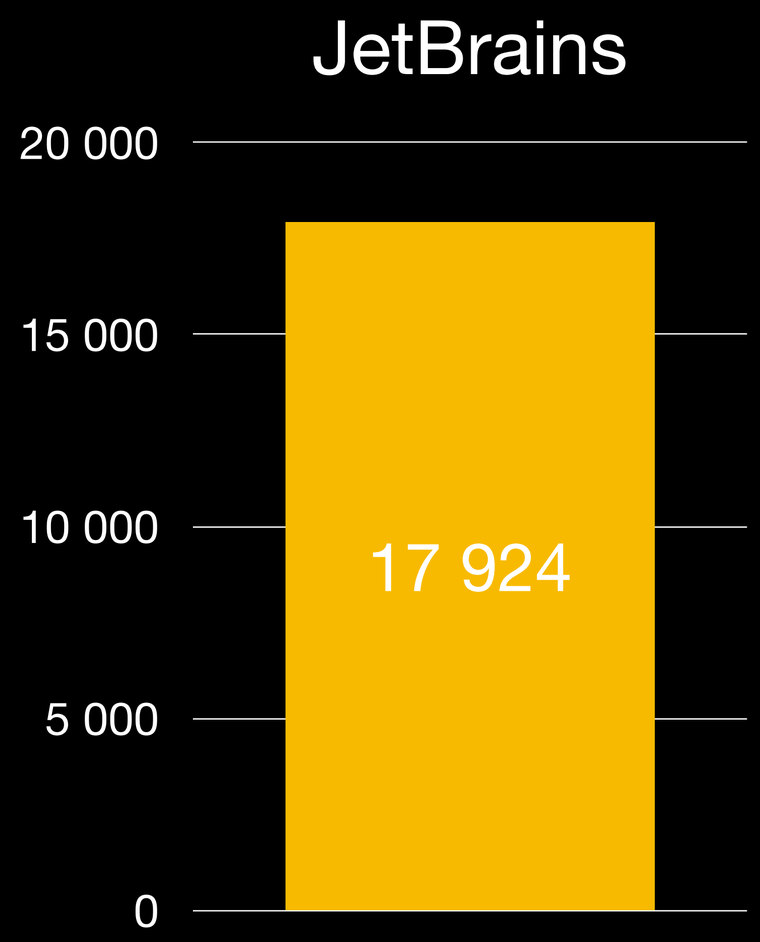
```
7   */
8   #[AllowDynamicProperties]
9   class WP {
```

WordPress/WordPress › wp-includes/class-wpdb.php ★ 18.2k

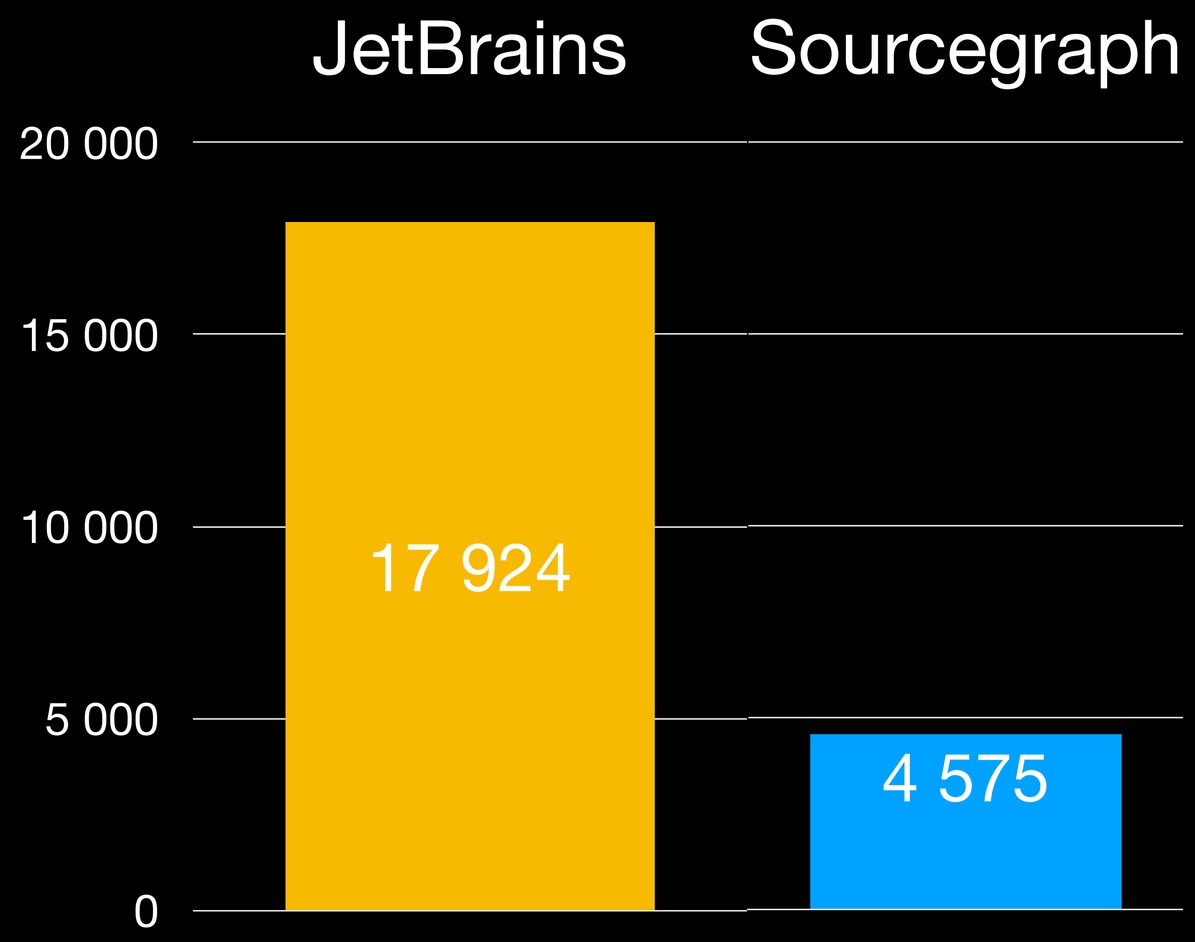
```
53  */
54  #[AllowDynamicProperties]
```

Repositories analysed

Repositories analysed



Repositories analysed

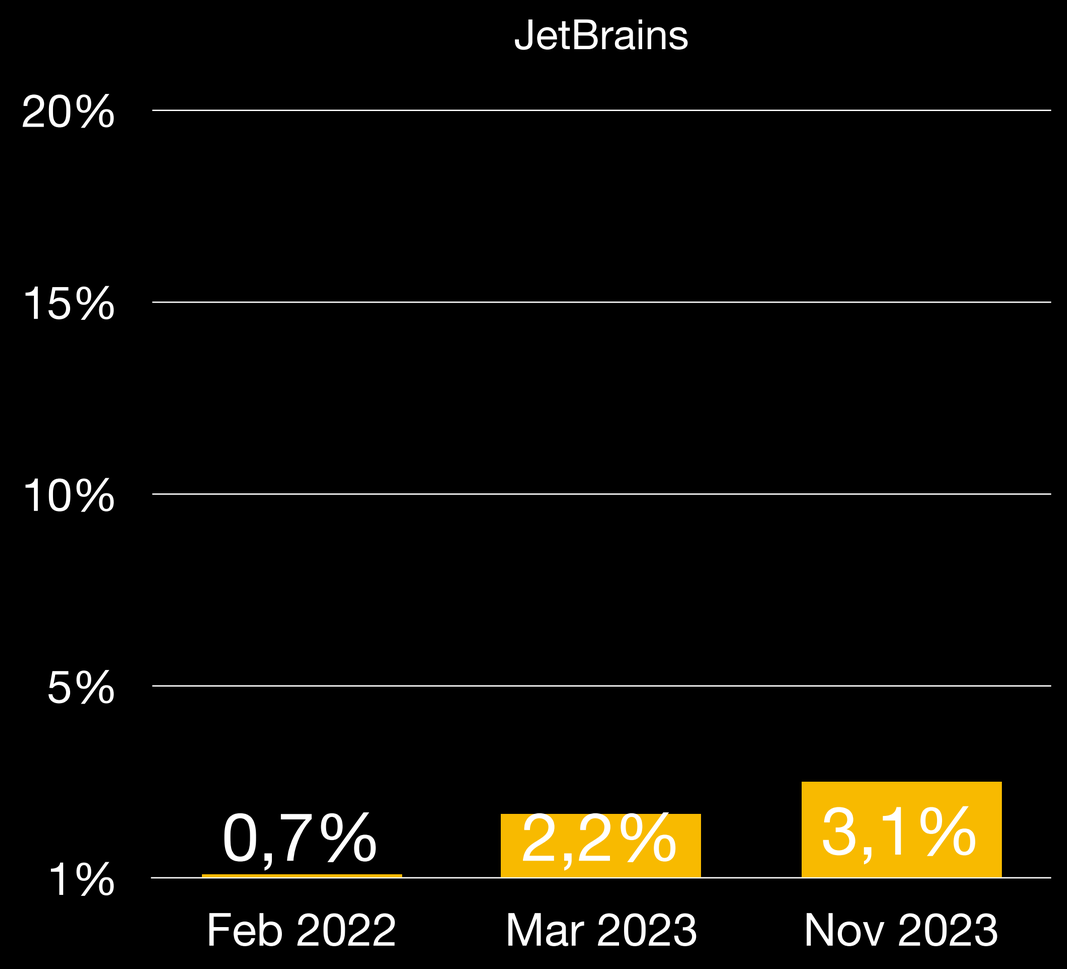


How many projects use

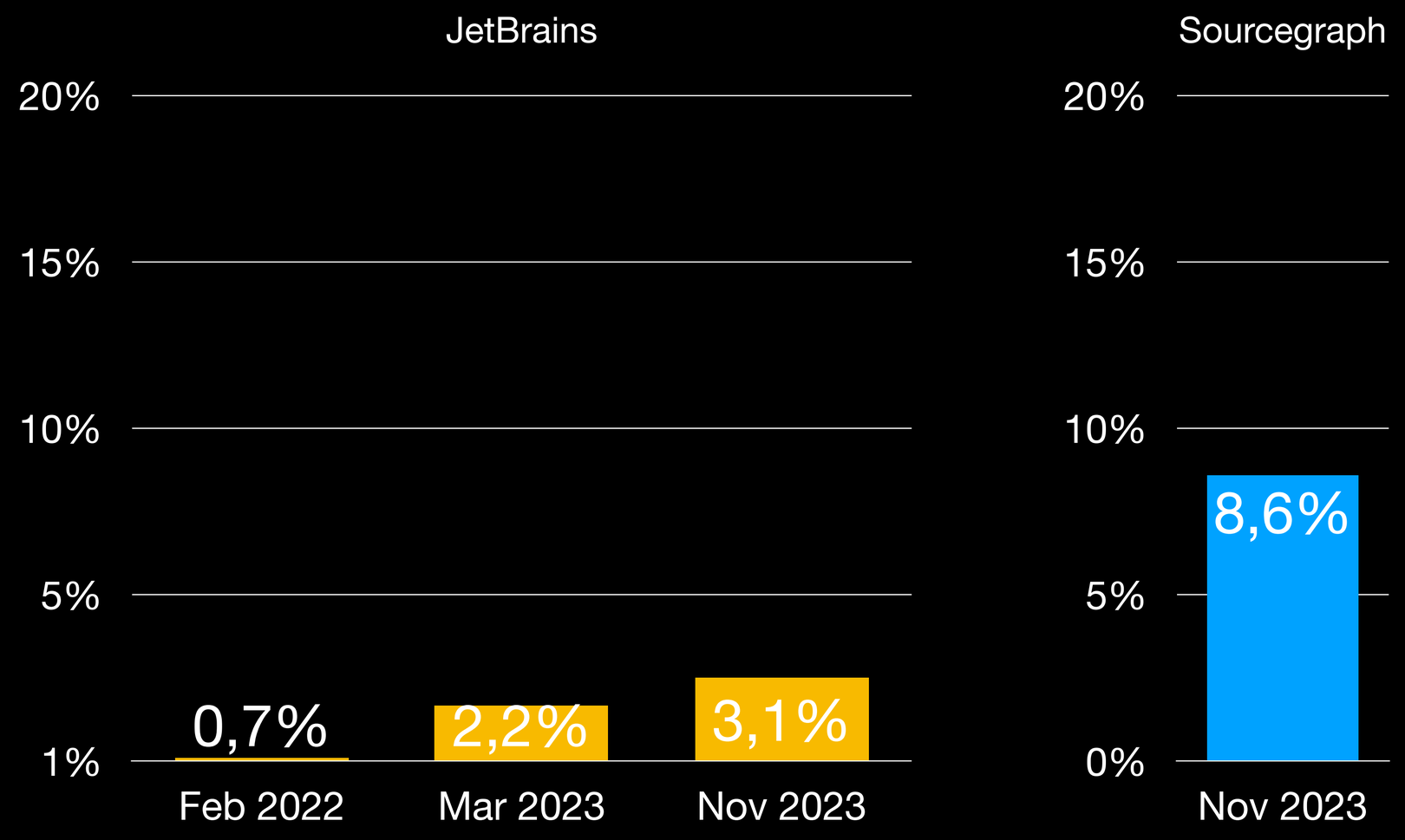
How many projects use

Enums

How many projects use Enums



How many projects use Enums

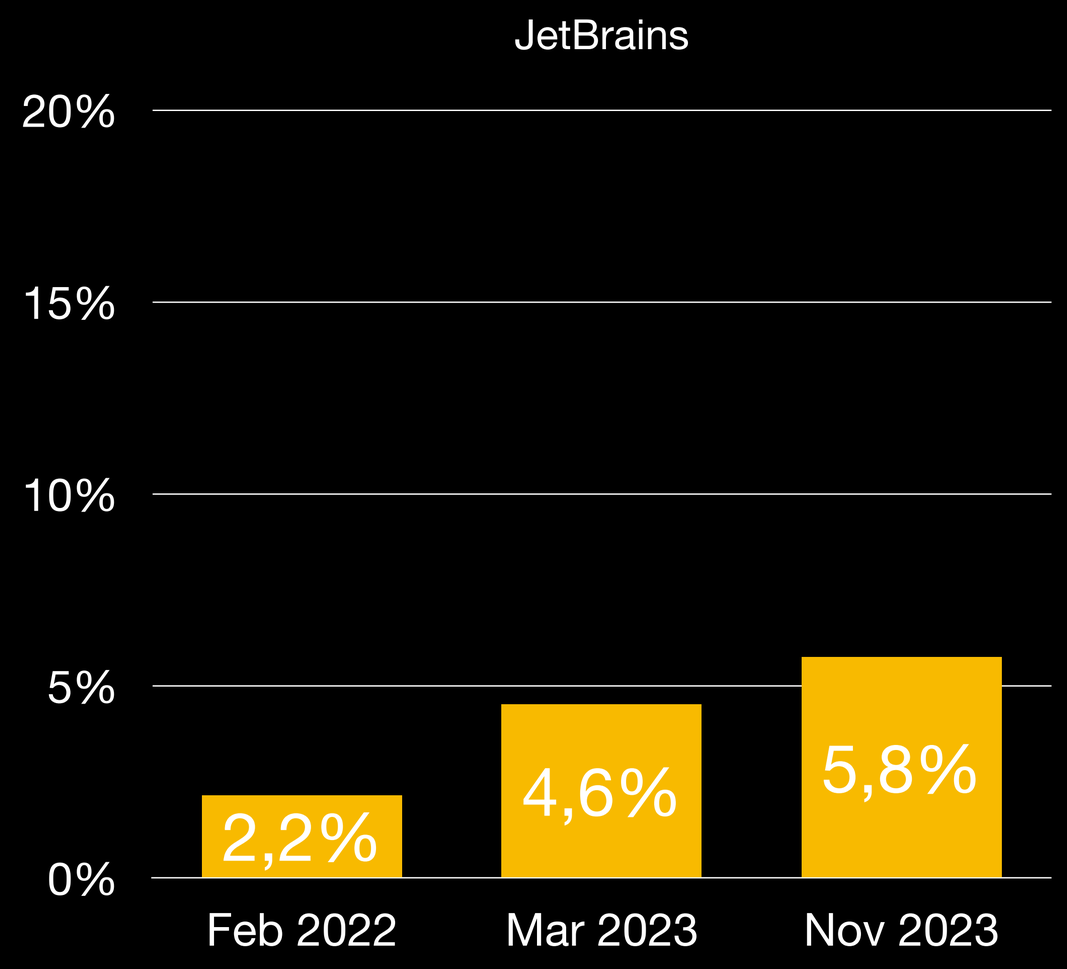


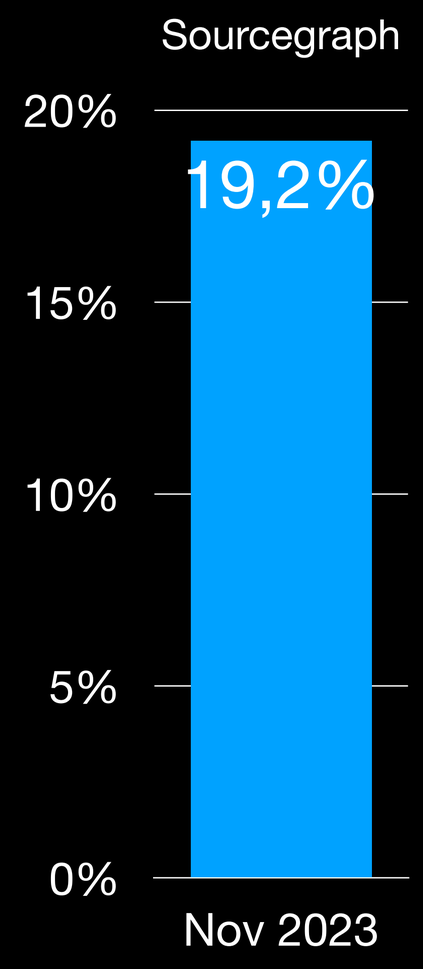
How many projects use

Readonly

How many projects use

Readonly



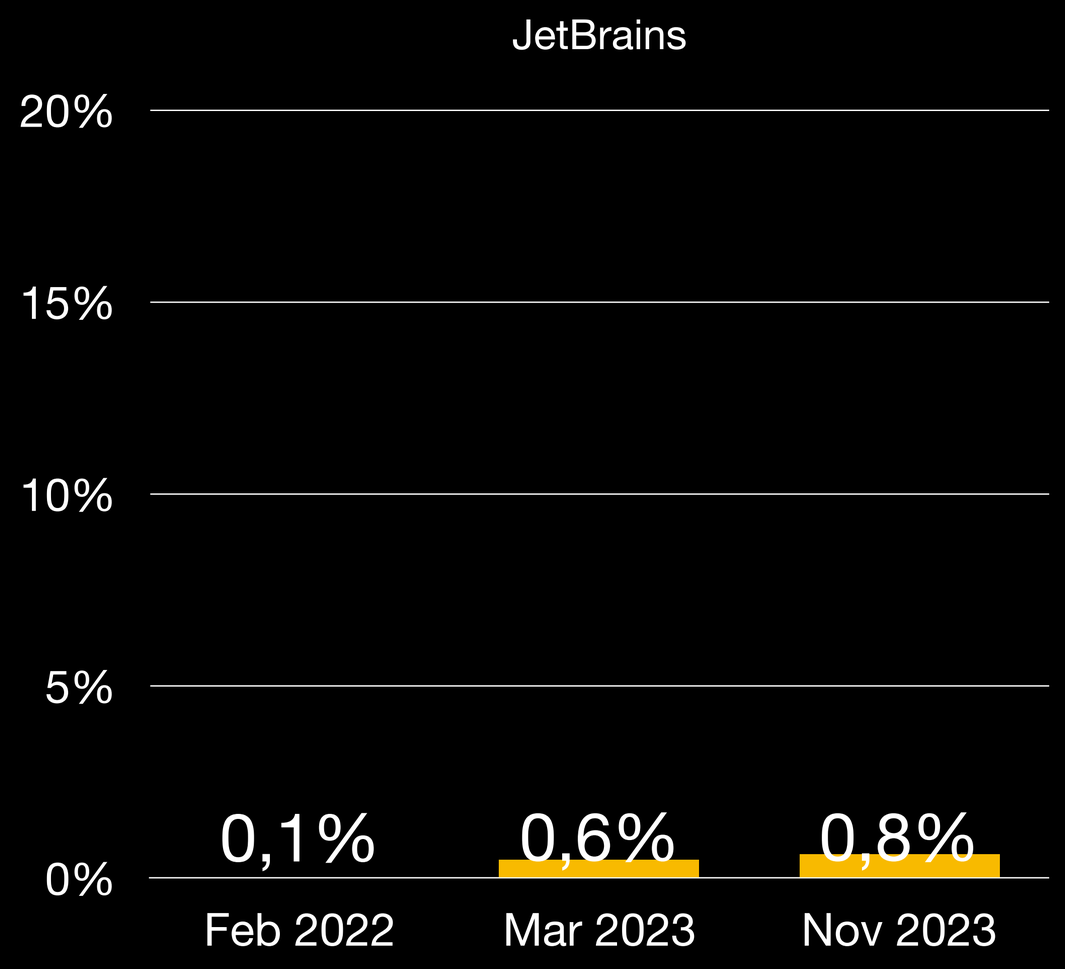


How many projects use

never
return type

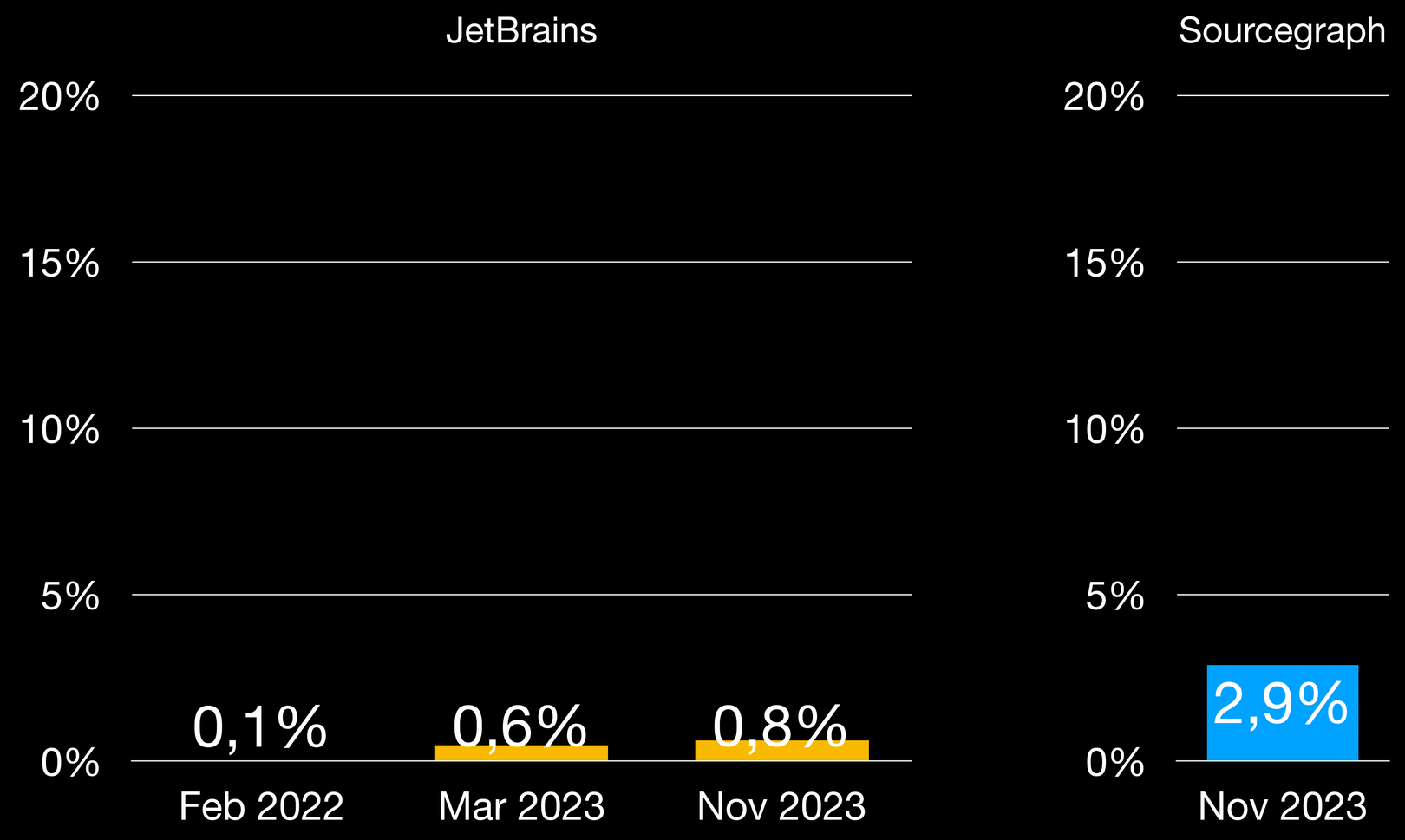
How many projects use

never return type



How many projects use

never return type

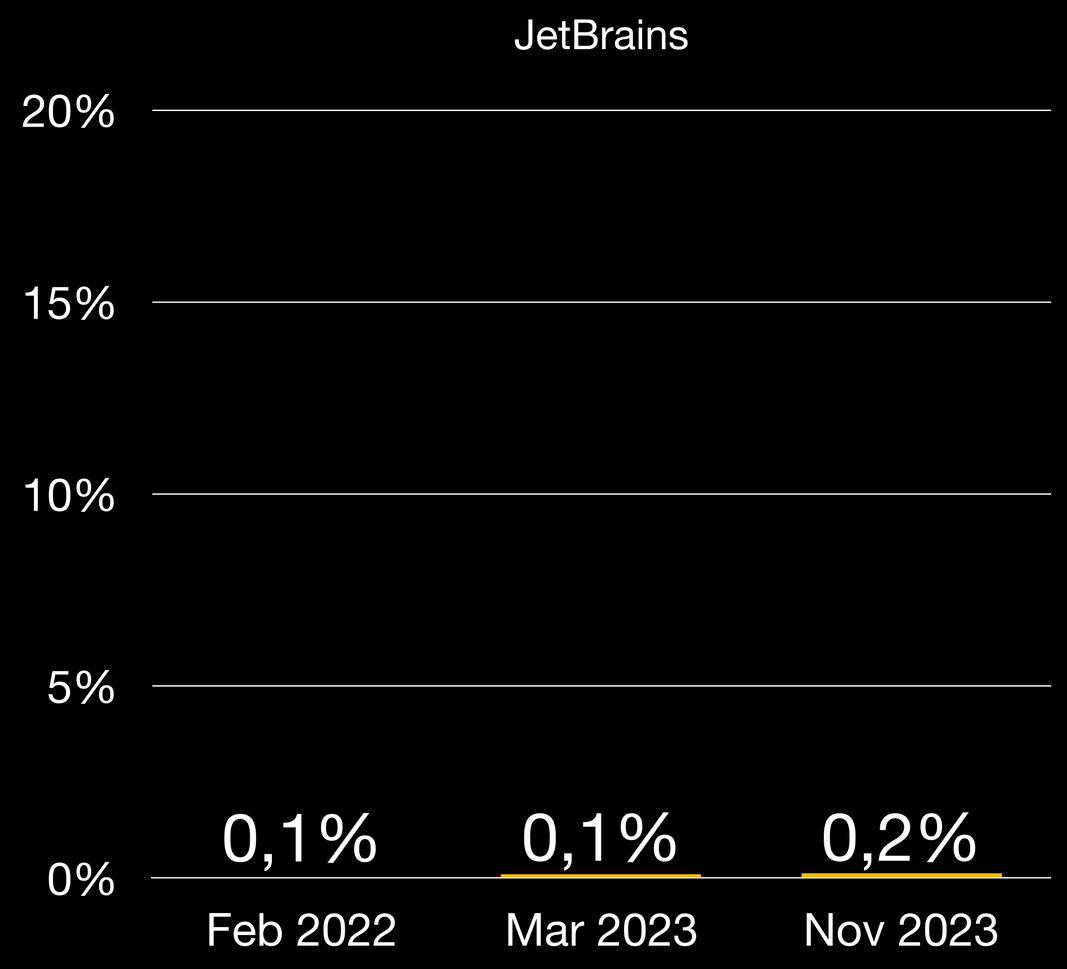


How many projects use

Fibers

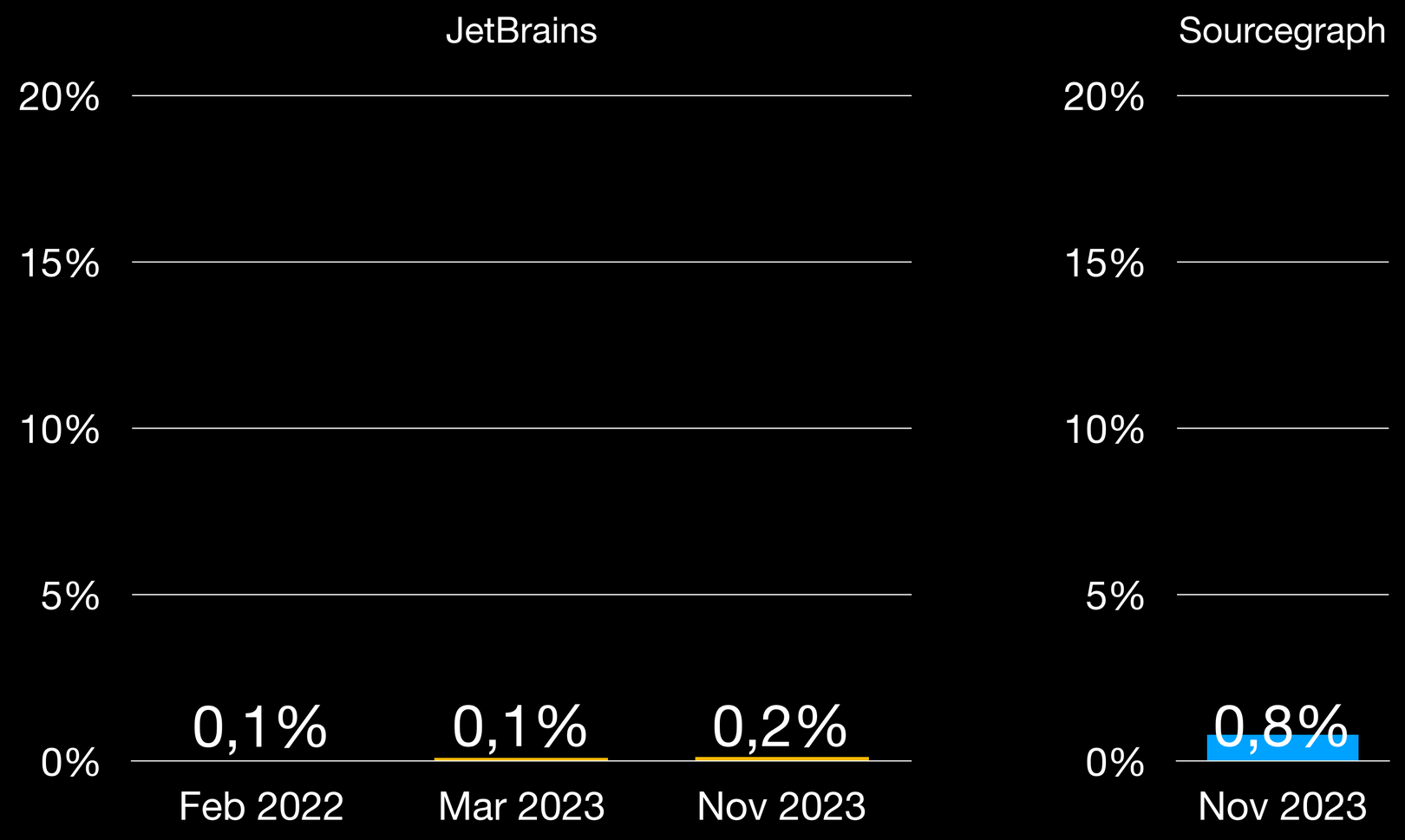
How many projects use

Fibers



How many projects use

Fibers

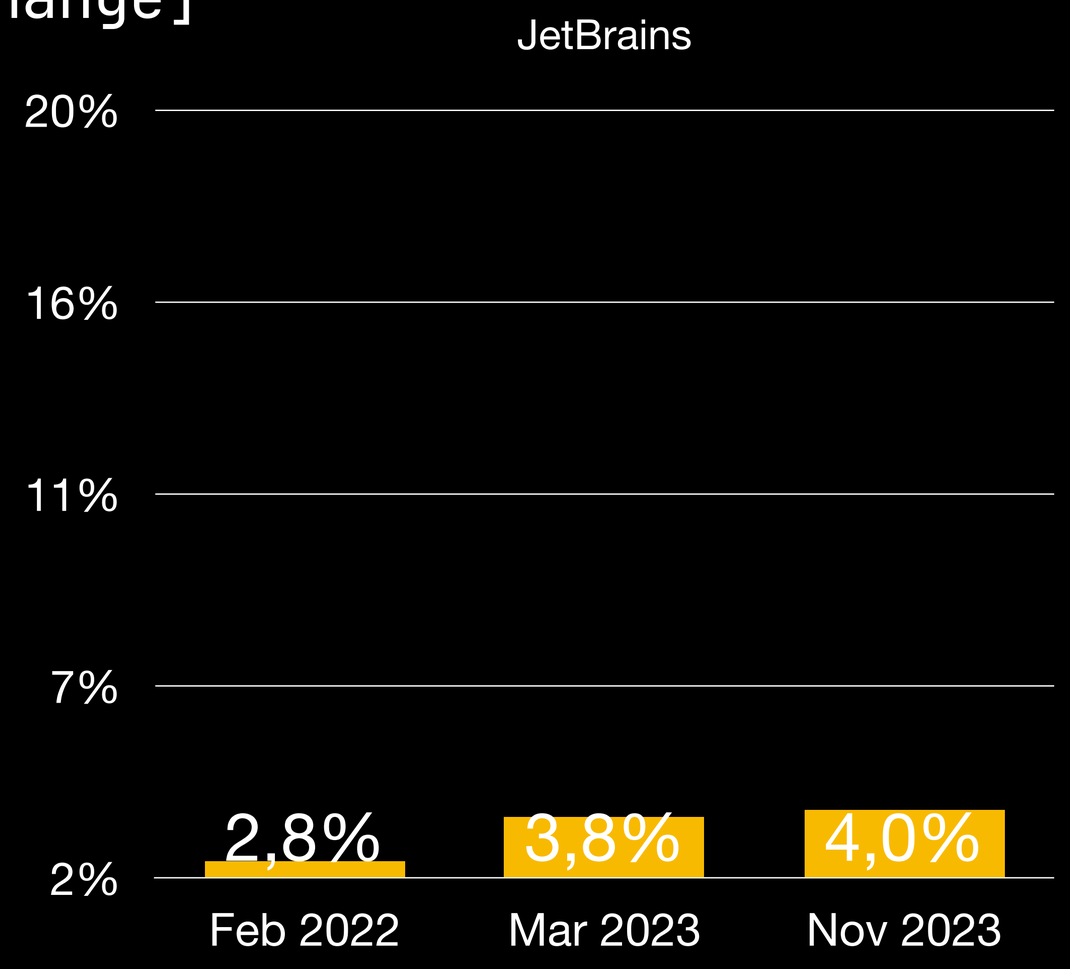


How many projects use

```
#[ReturnTypeWillChange]
```

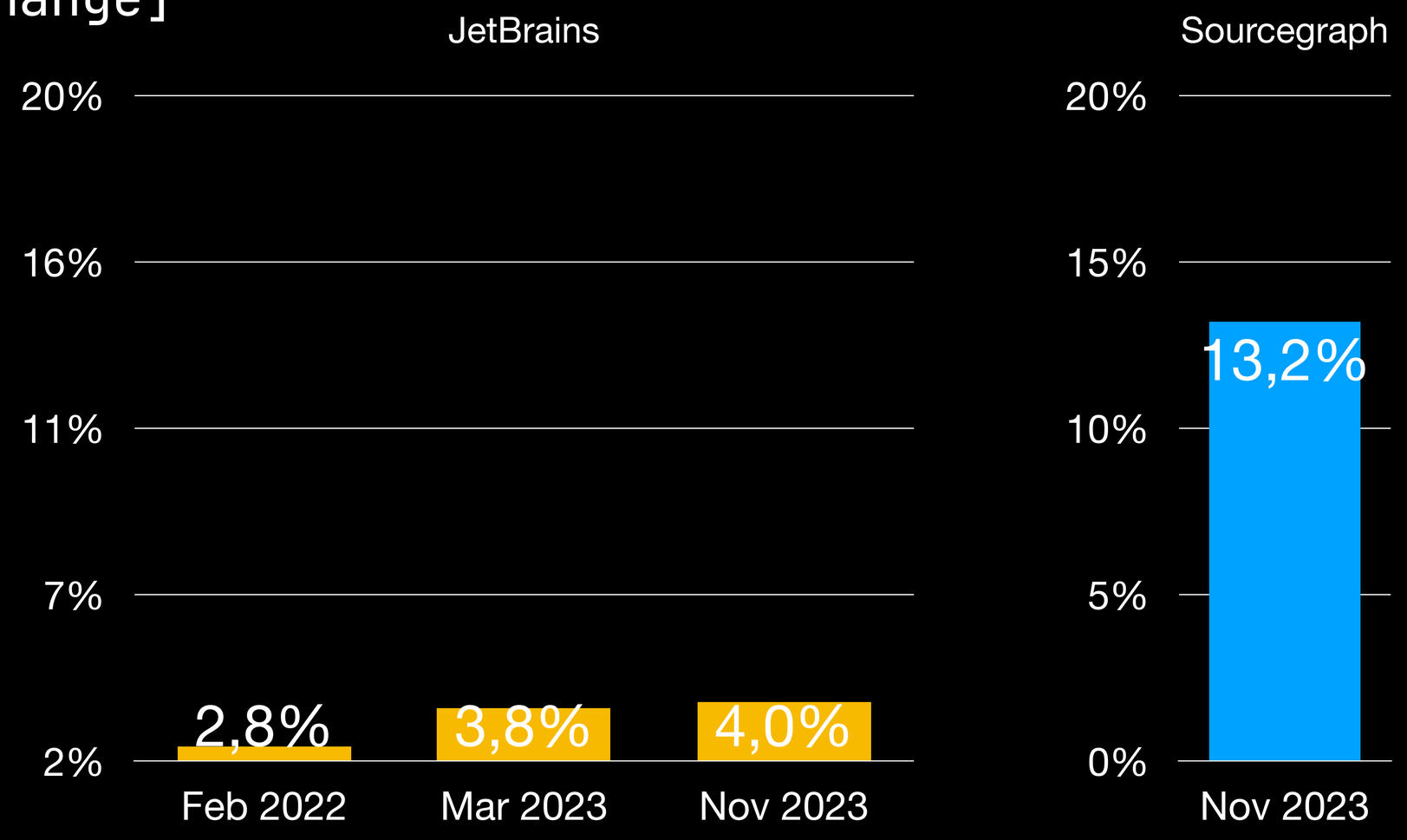
How many projects use

`#[ReturnTypeWillChange]`



How many projects use

`#[ReturnTypeWillChange]`



How many projects use

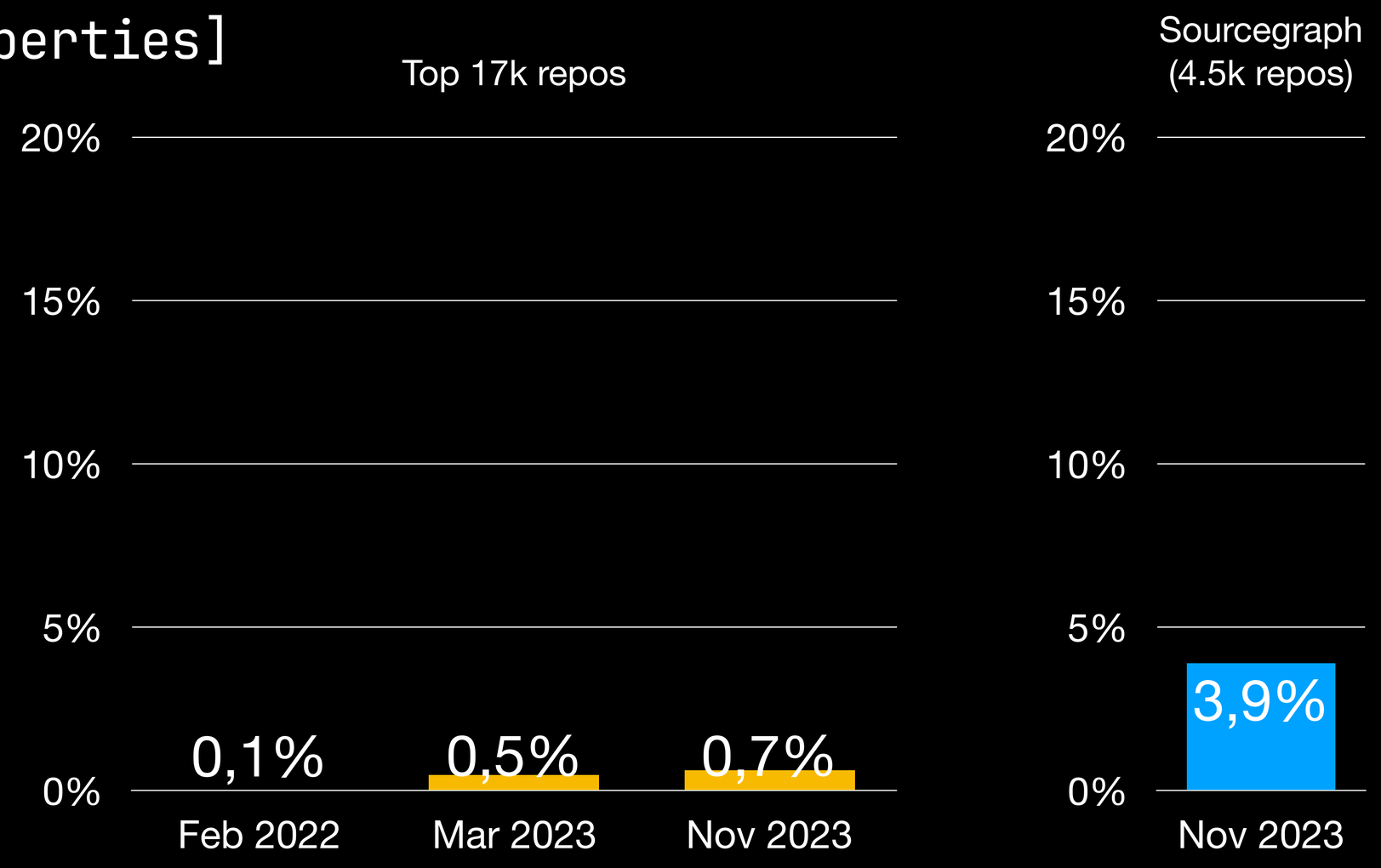
```
#[AllowDynamicProperties]
```

How many projects use

`#[AllowDynamicProperties]`

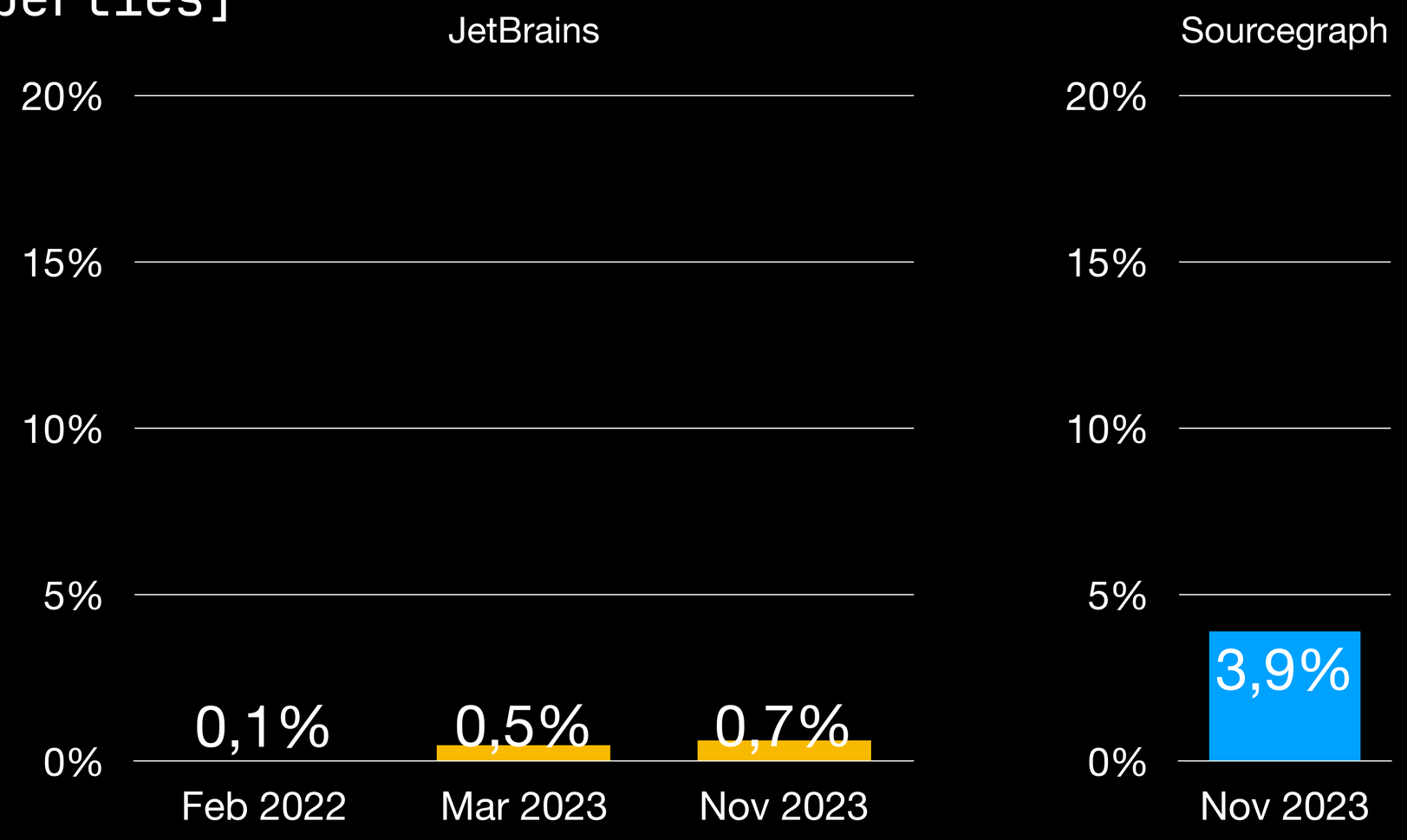
How many projects use

`#[AllowDynamicProperties]`



How many projects use

`#[AllowDynamicProperties]`

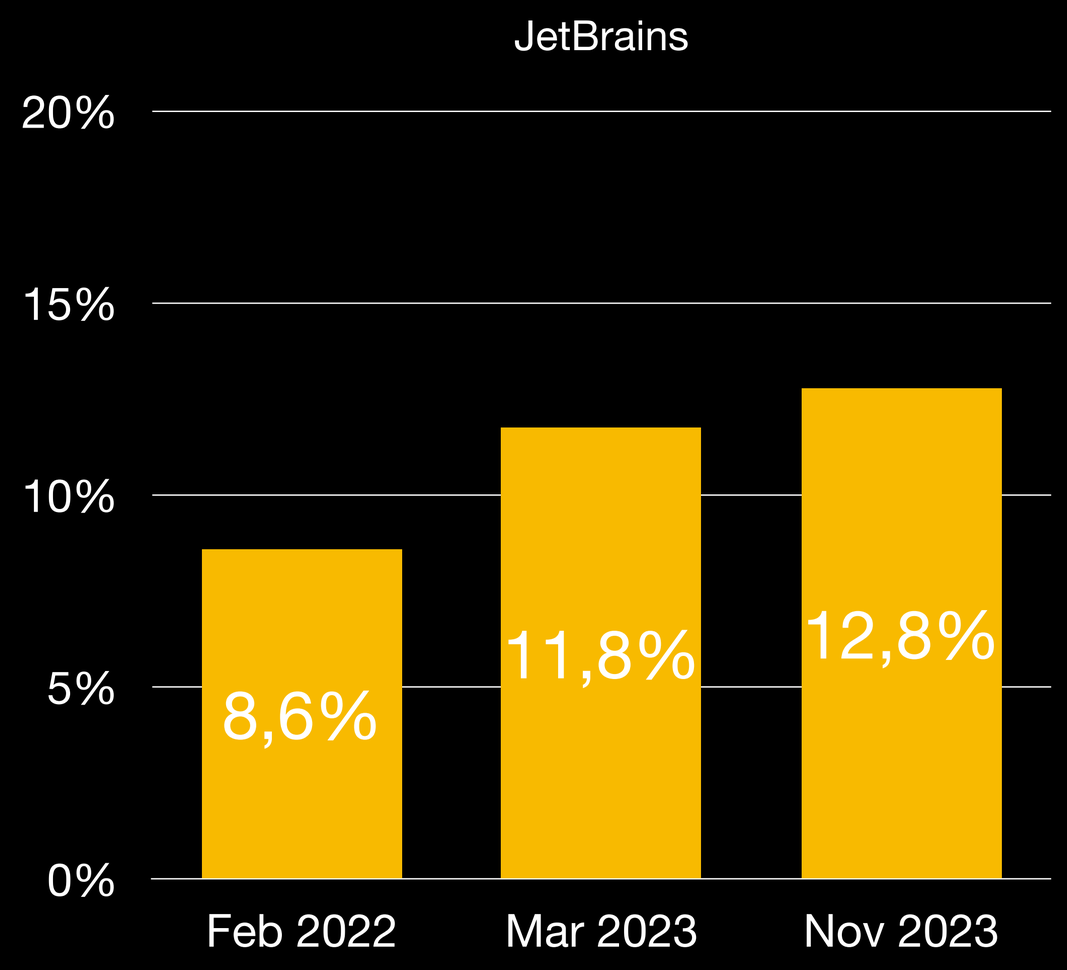


How many projects use

```
#[Attributes]
```

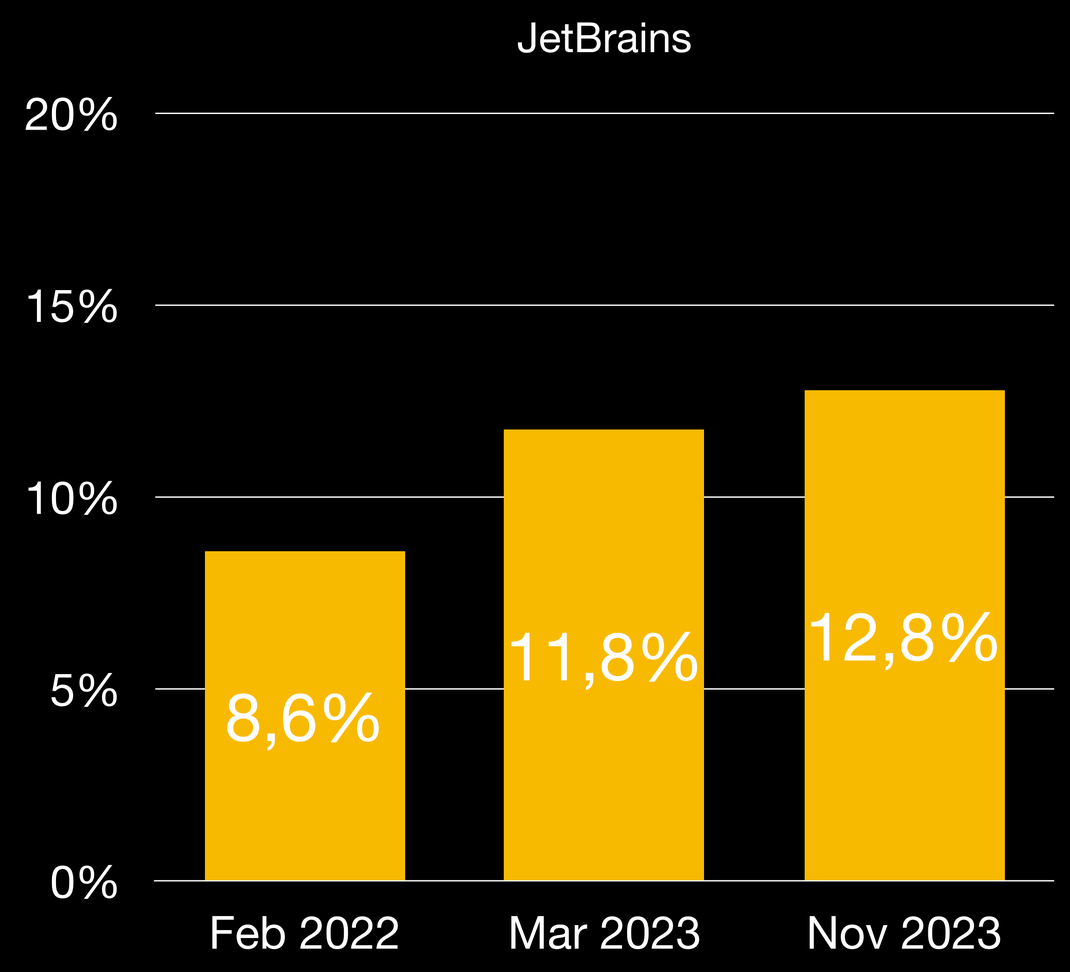
How many projects use

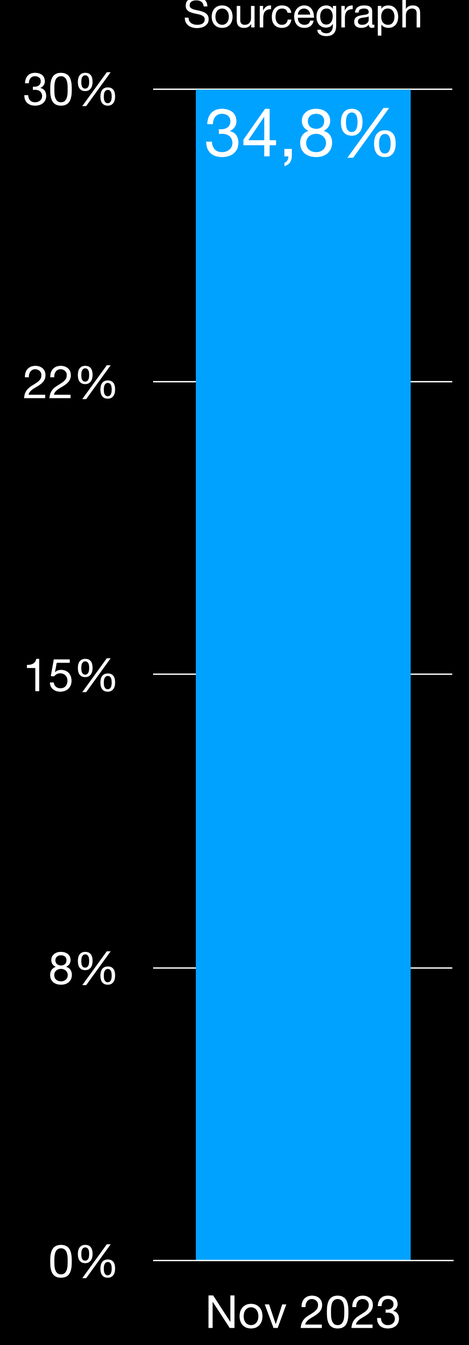
`#[Attributes]`



How many projects use

`#[Attributes]`



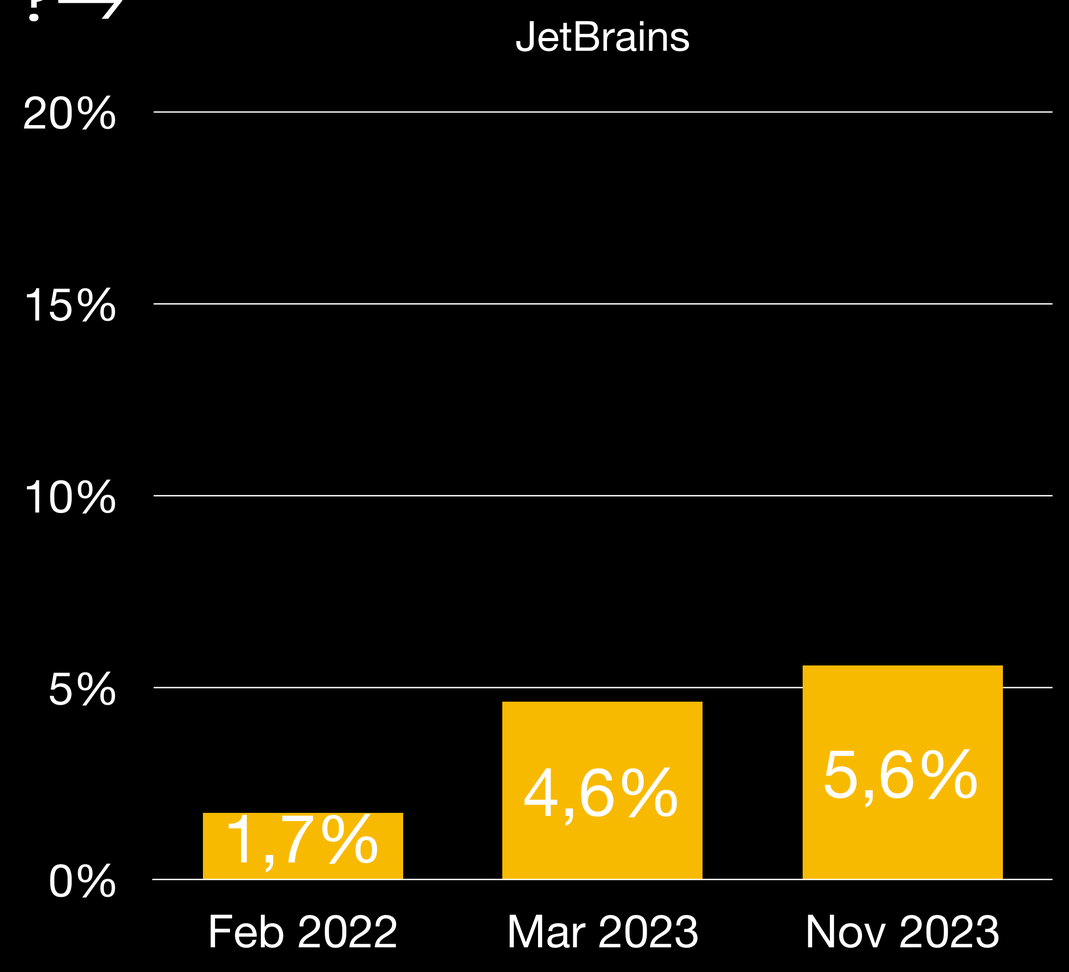


How many projects use

`Nullsafe operator ?→`

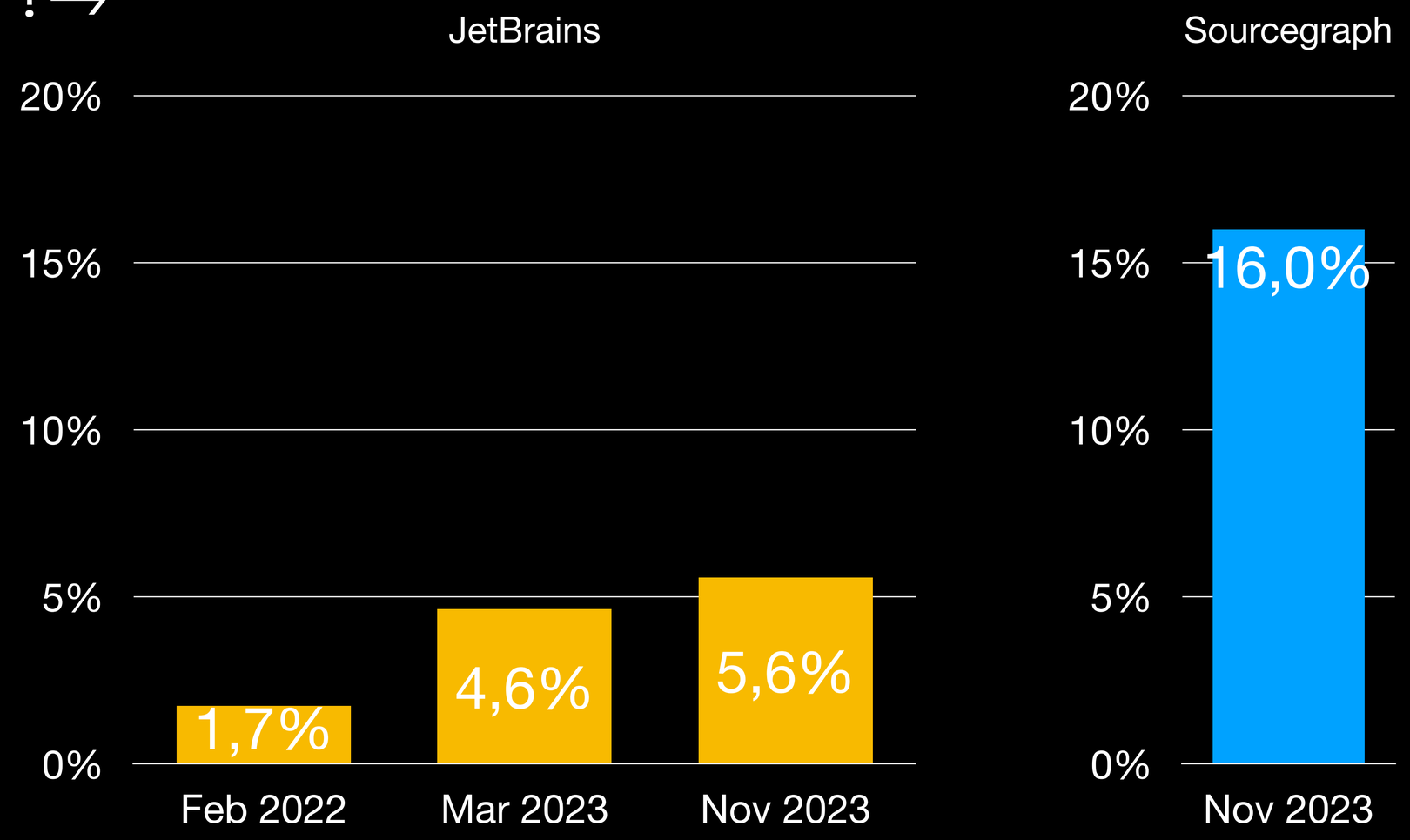
How many projects use

`Nullsafe operator ?→`



How many projects use

`Nullsafe operator ?→`

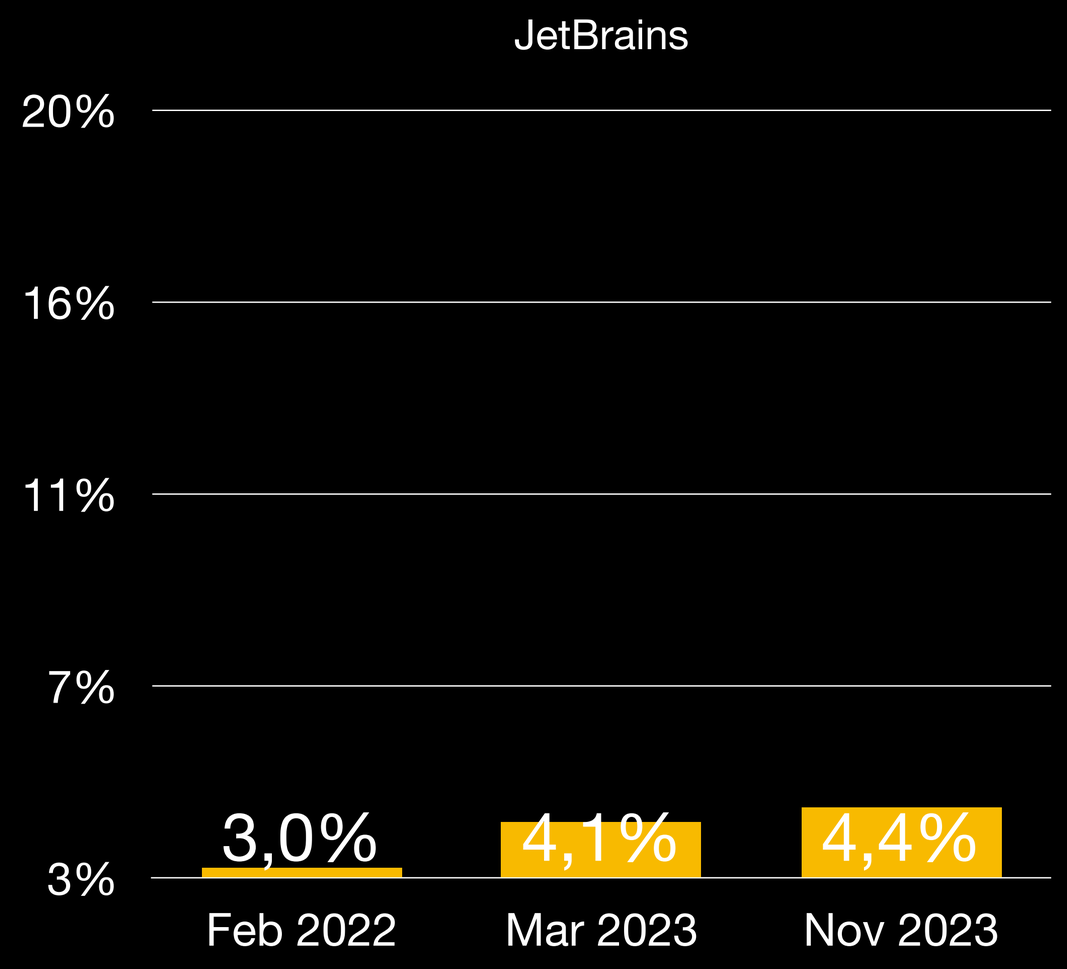


How many projects use

Generics

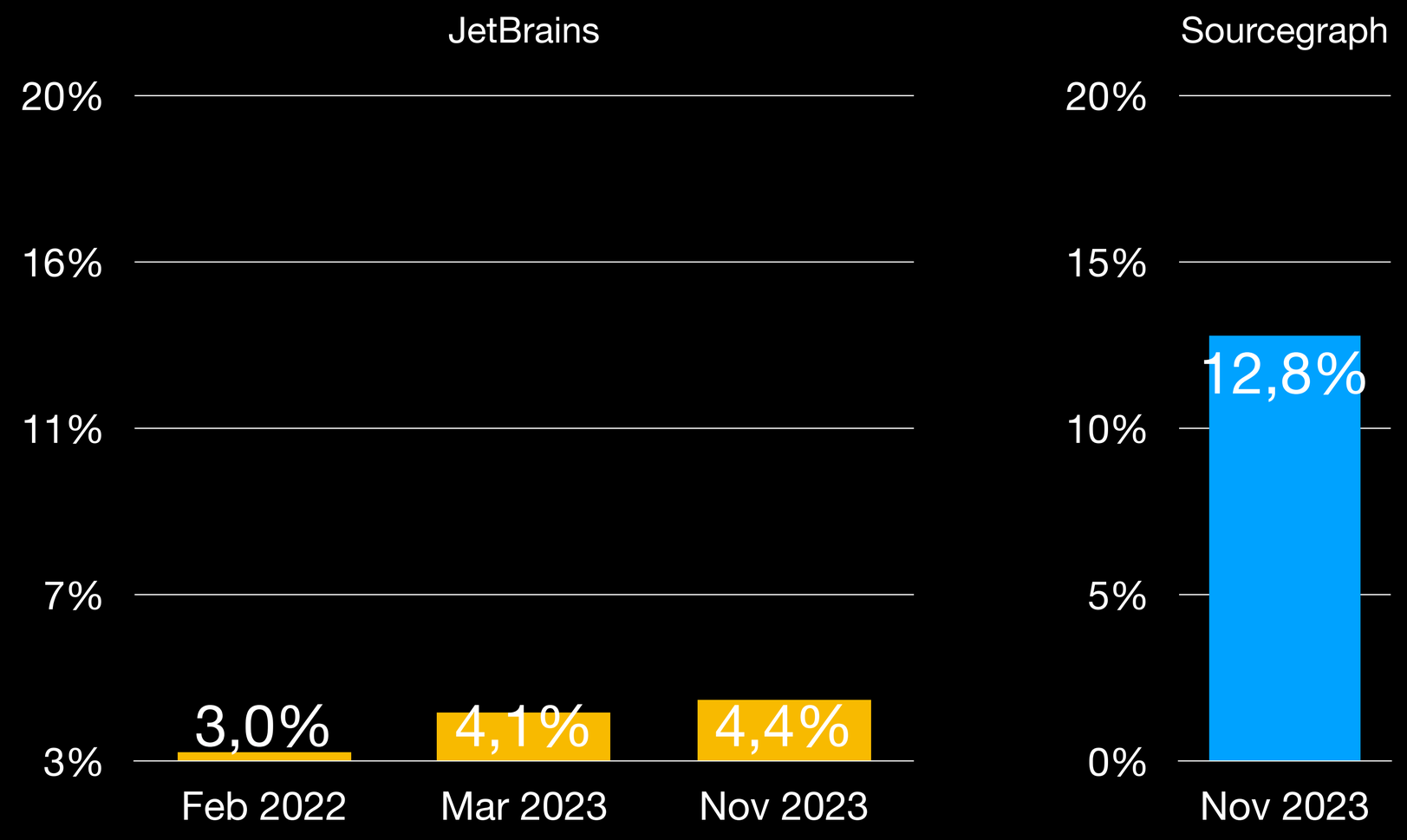
How many projects use

Generics



How many projects use

`Generics`

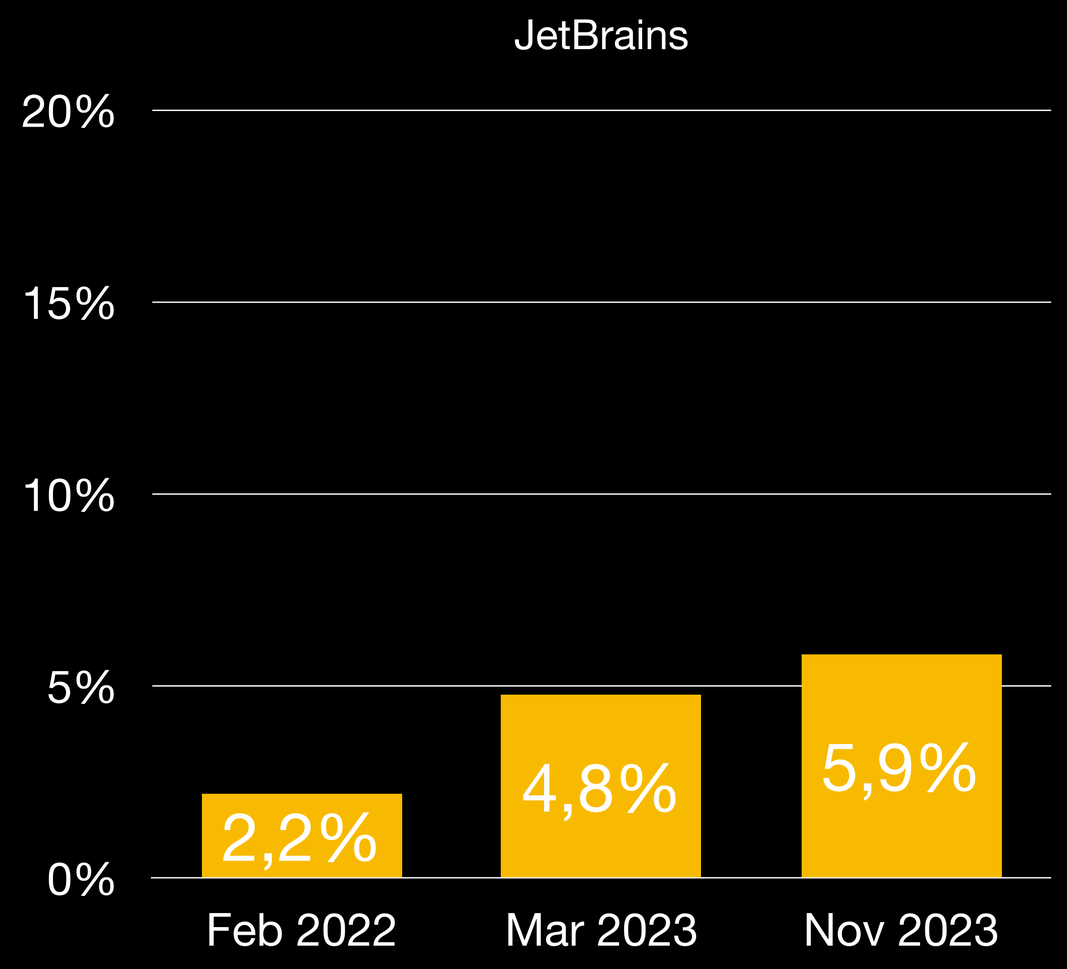


How many projects use

Match Expression

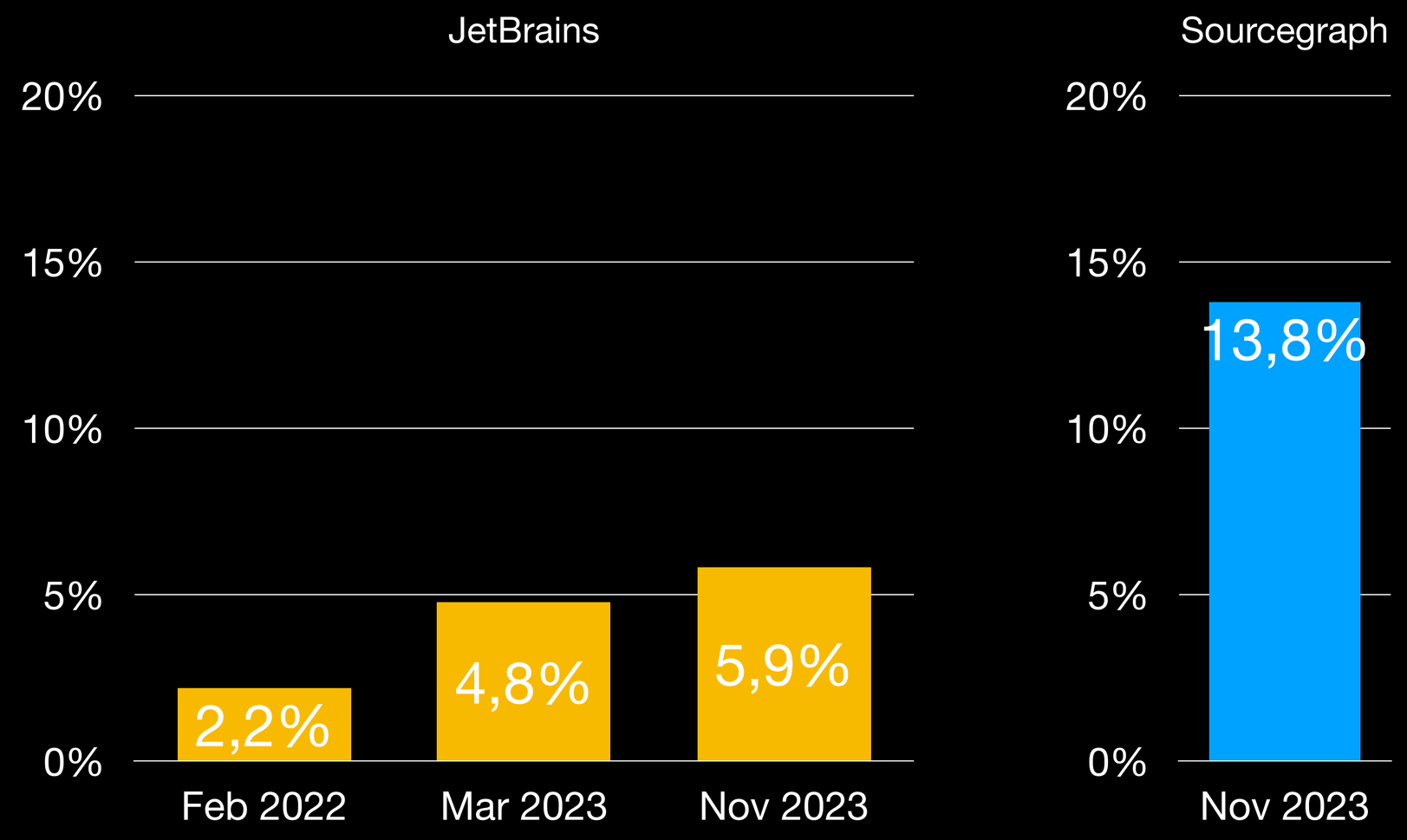
How many projects use

`Match Expression`



How many projects use

`Match Expression`

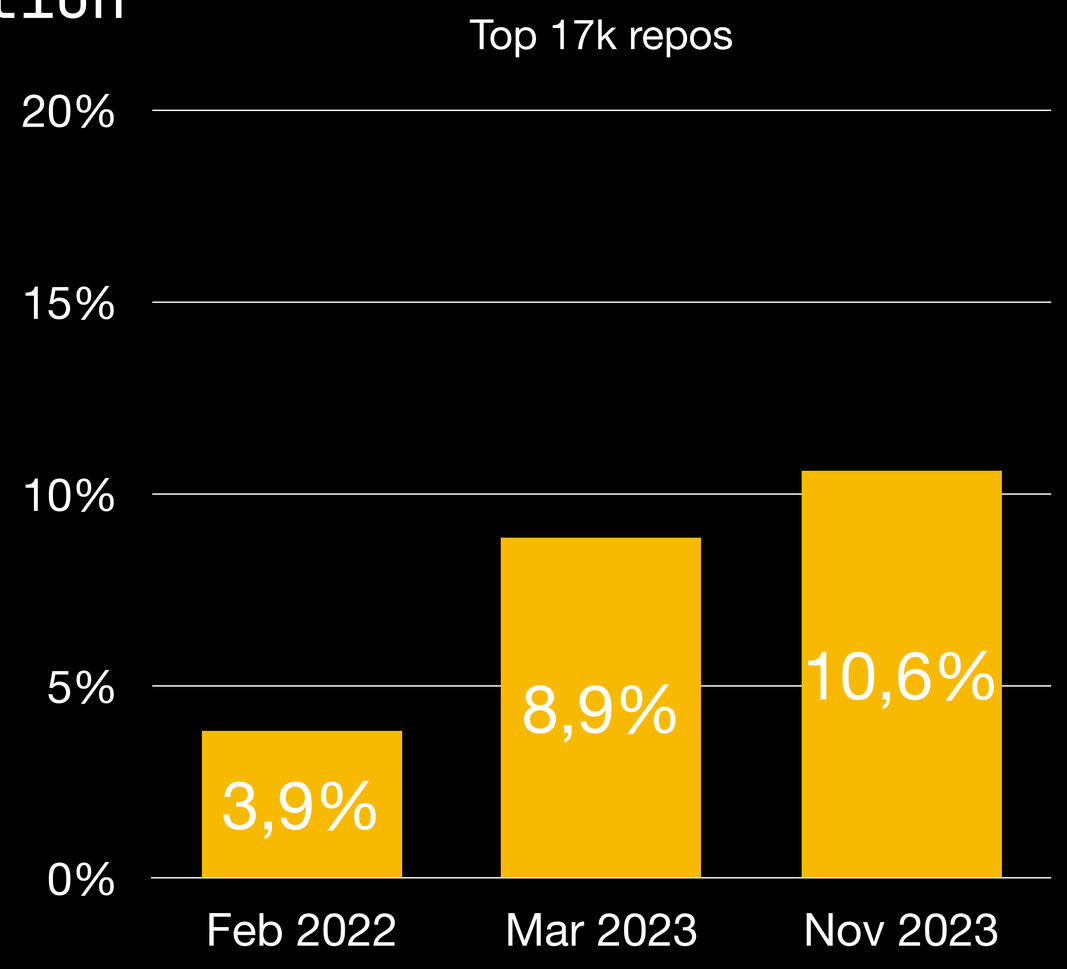


How many projects use

Constructor Promotion

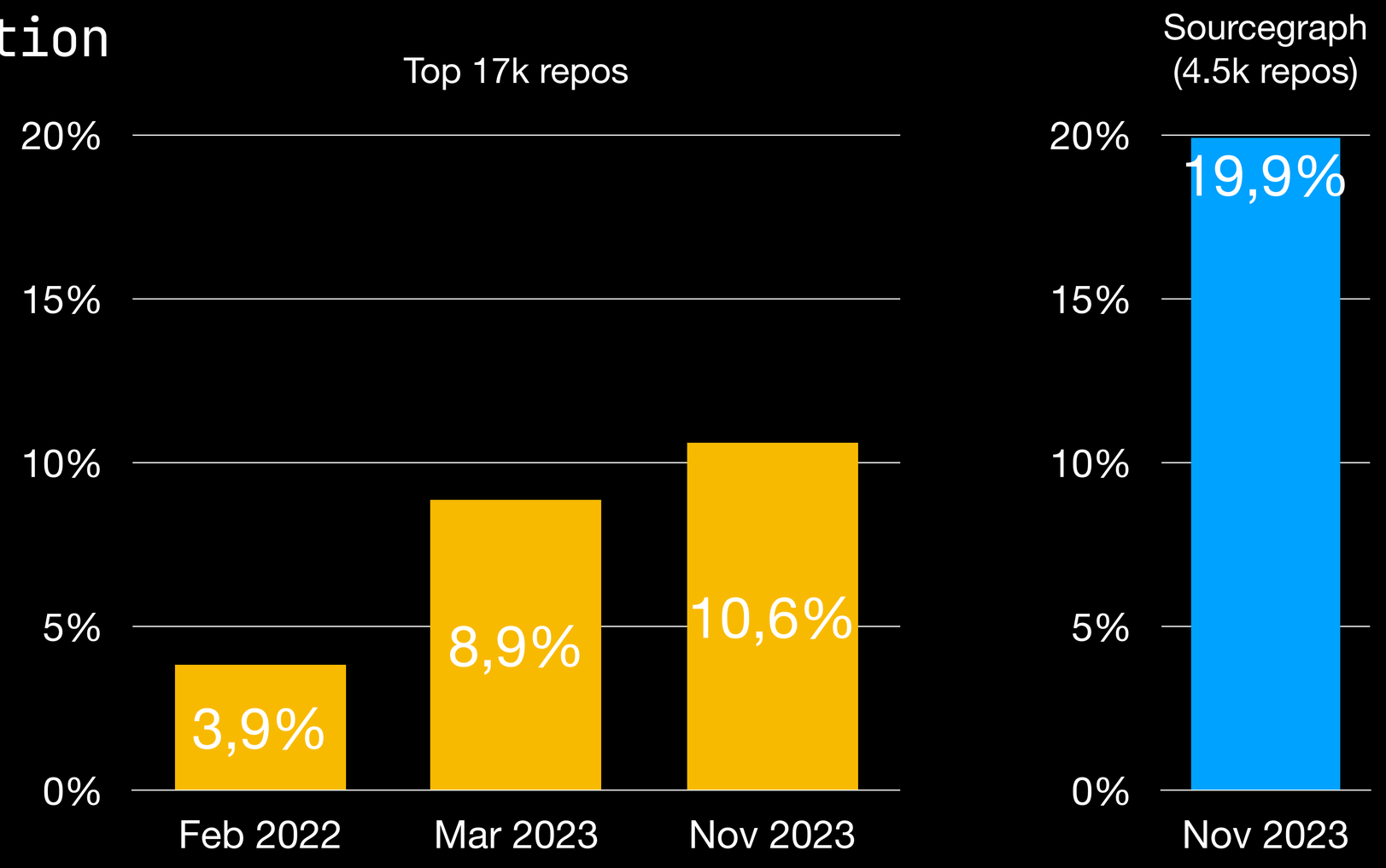
How many projects use

Constructor Promotion



How many projects use

`Constructor Promotion`

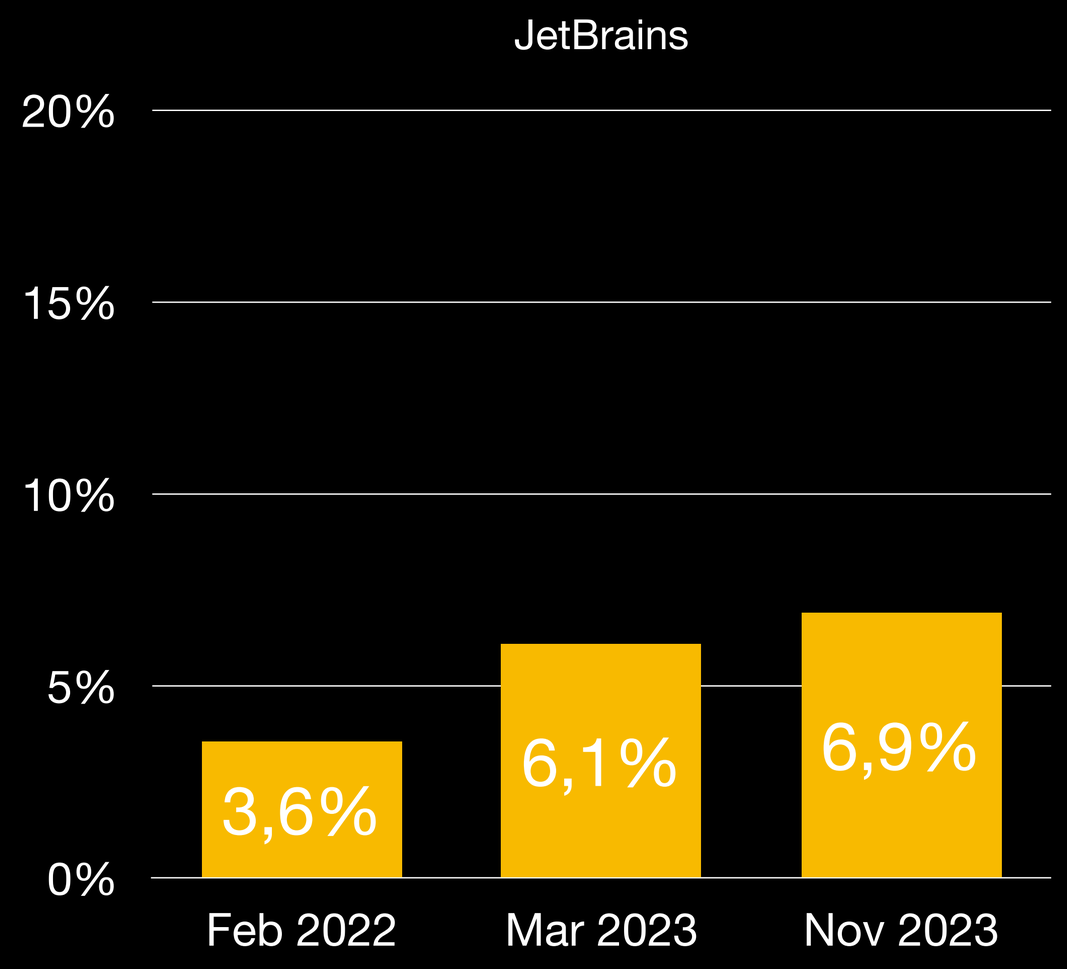


How many projects use

`str_contains()`

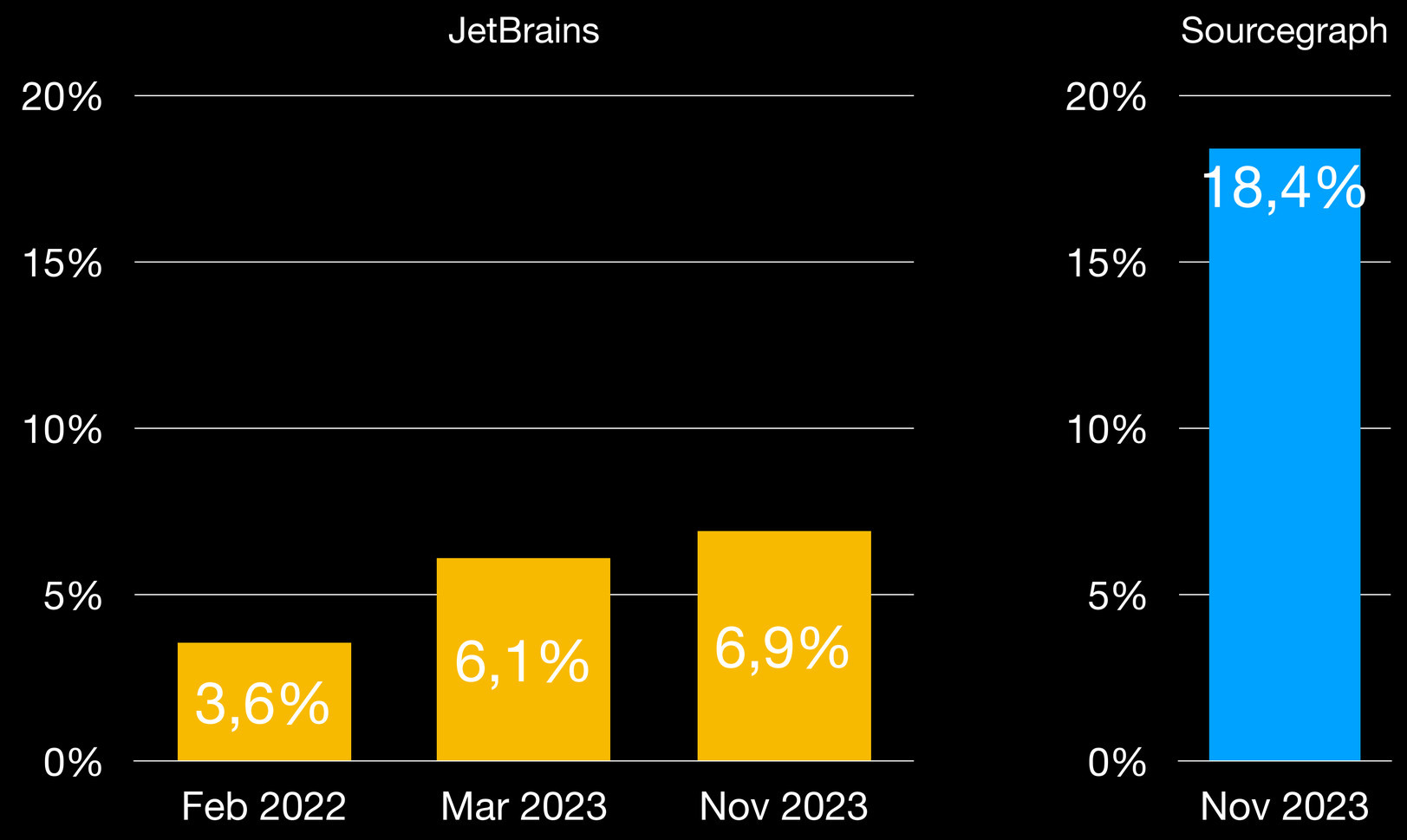
How many projects use

`str_contains()`



How many projects use

`str_contains()`



How do I check if a string contains a specific word?

Asked 12 years, 11 months ago Modified 3 months ago Viewed 6.6m times Part of PHP Collective

2657

Consider:

```
$a = 'How are you?';

if ($a contains 'are')
    echo 'true';
```

Suppose I have the code above, what is the correct way to write the statement `if ($a contains 'are')`?

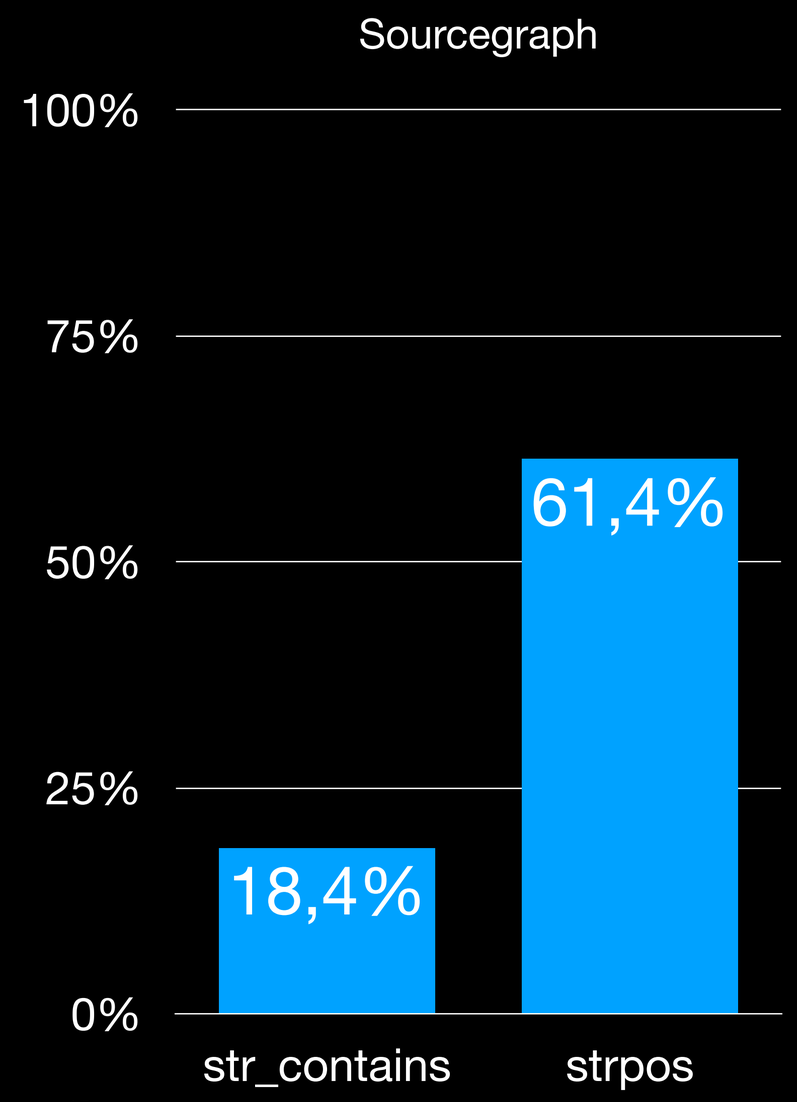
str_contains() vs strpos() ≢ false

How many projects use

```
str_contains()
vs.
strpos() ≢ false
```

How many projects use

```
str_contains()
vs.
strpos() ≢ false
```



4. Brainstorm

What features do we need?

Maybe we don't need some at all?

Generics?

```
/** @template T */
class Map
{
    /** @param T $item */
    public function __construct(mixed $item) {...}

    /** @return T|null */
    public function first(): mixed {...}
}

$map = (new Map(new User));
$map->first()->;
```

Generics?

```
/** @template T */
class Map
{
    /** @param T $item */
    public function __construct(mixed $item) {...}

    /** @return T|null */
    public function first(): mixed {...}
}

$map = (new Map(new User));
$map->first()->;
```

Generics?

Generics?

tl;dr: How about using generics in PHP attributes?

```
#[<T>]
class Stack
{
    public function push(#[<T>] mixed $item): void
    {
    }

    public function pop(): #[<T>] mixed
    {
    }
}
```

https://pronskiy.com/blog/generics-via-attributes-in-php/

Variables without $?

```php
<?php

class User
{
    public string $name;

    public function changeName($newName)
    {
        name = $newName;
    }
}
```

https://plugins.jetbrains.com/plugin/10611-php-foldings

Variables without $?

```php
<?php

class User
{
    public string $name;

    public function changeName($newName)
    {
        name = $newName;
    }
}
```

https://plugins.jetbrains.com/plugin/10611-php-foldings

My PHP Wishlist 🎅

My PHP Wishlist 🎅

- Core PHP packages

My PHP Wishlist 🎅

- Core PHP packages
- Composer built-in
- Static analyser built-in

My PHP Wishlist

- Core PHP packages
- Optional runtime type checks

My PHP Wishlist 🎅🏻

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc

How do you get the correct length of a string with emojis?

`strlen('👩‍❤️‍👩');` returns 25.
`mb_strlen('👩‍❤️‍👩');` returns 7.
`grapheme_strlen('👩‍❤️‍👩');` is what you need, and it returns 1.

My PHP Wishlist

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions

My PHP Wishlist 🎅

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Swoole / swow

```
$http = new Swoole\Http\Server('127.0.0.1', 9501);
$http->set(['hook_flags' => SWOOLE_HOOK_ALL]);

$http->on('request', function ($request, $response) {
    $result = [];
    Co::join([
        go(function () use (&$result) {
            $result['google'] = file_get_contents( filename: "https://www.google.c
        }),
        go(function () use (&$result) {
            $result['bing'] = file_get_contents( filename: "https://www.bing.com/"
        })
    ]);
    $response->end(json_encode($result));
});

$http->start();
```

My PHP Wishlist 🎅

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Swoole / swow

My PHP Wishlist 🎅

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Swoole / swow
- Scalar objects

```php
class StringHandler {
    public static function length($self) {
        return grapheme_strlen($self);
    }
}

register_primitive_type_handler('string', StringHandler::class);

$string = "abc";
var_dump($string->length()); // int(3)
```

My PHP Wishlist

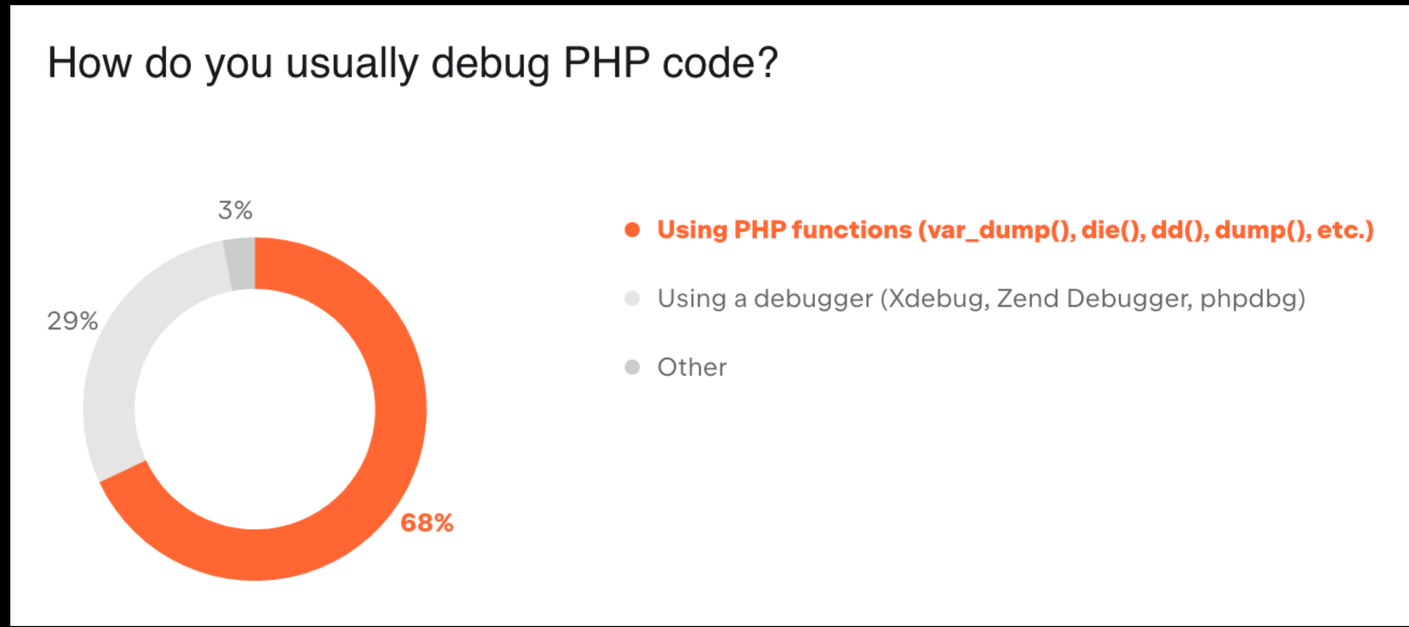
- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions

My PHP Wishlist 🎅

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Built-in debugger

My PHP Wishlist 🎅🏻

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Built-in debugger



My PHP Wishlist 🎅

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Built-in debugger

My PHP Wishlist 🎅

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Built-in debugger
- Local readonly variables

```
function foobar()
{
    readonly $local_const = '💜';

    echo $local_const;

    $local_const = '⚠️'; // Compile time error
}
```

My PHP Wishlist 🎅🏻

- Core PHP packages
- Optional runtime type checks
- Modern JSON, String, etc
- Bundled extensions
- Built-in debugger
- Local readonly variables
- Back tic for string templates instead of eval

```php
<?php

$output = `ls -al`;
echo "<pre>$output</pre>";

// vs

$template_literal = `
    Hello ${getName()}
`;
```

My PHP Wishlist 🎅

My PHP Wishlist 🎅

- Experimental features

Stability: 0 - Deprecated. The feature may emit warnings. Backward compatibility is not guaranteed.

Stability: 1 - Experimental. Non-backward compatible changes or removal may occur in any future release. Use of the feature is not recommended in production environments.

Stability: 2 - Stable. Compatibility with the Composer ecosystem is a high priority.

Stability: 3 - Legacy. Although this feature is unlikely to be removed, it is no longer actively maintained, and other alternatives are available.

Get involved! Give your feedback

Property Hooks

READ THE RFC | EXTERNALS | 32 | 47 YES | 31 NO

```
class User
{
    public string $name {
        set {
            if (strlen($value) === 0) {
                throw new ValueError("Name must be non-empty");
            }
            $field = $value;
        }
    }

    public function __construct(string $name) {
        $this->name = $name;
    }
}
```

60% 40%

Take a survey

Thank you

Roman Pronskiy
@pronskiy