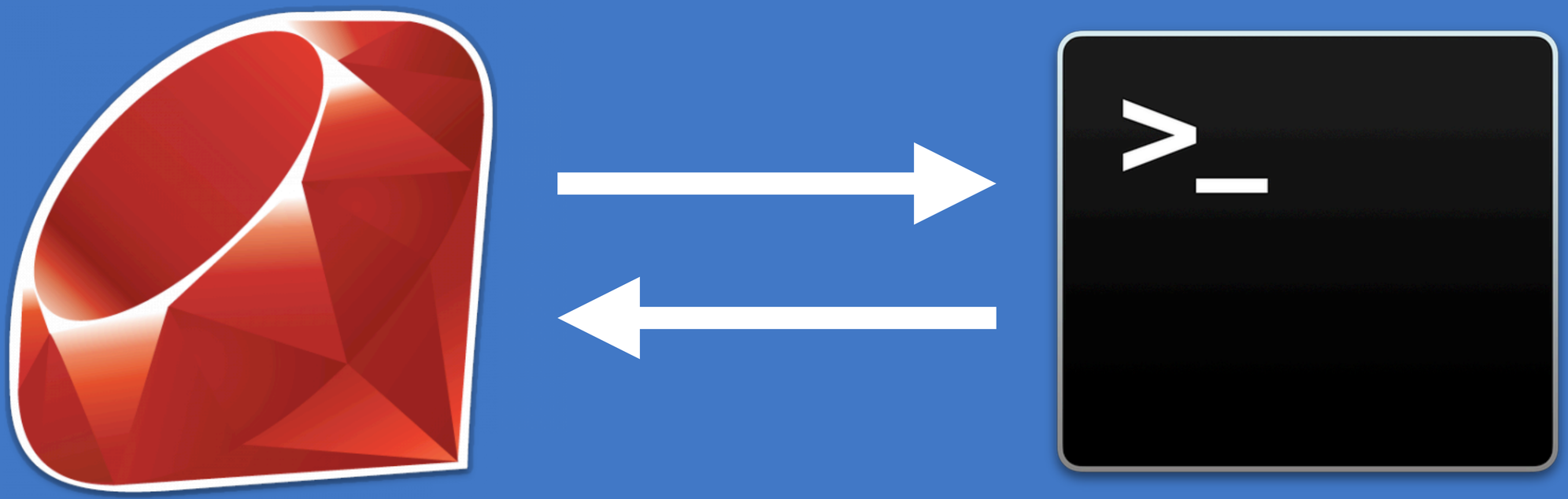
HELIX

Stanko Krtalic Rusendic

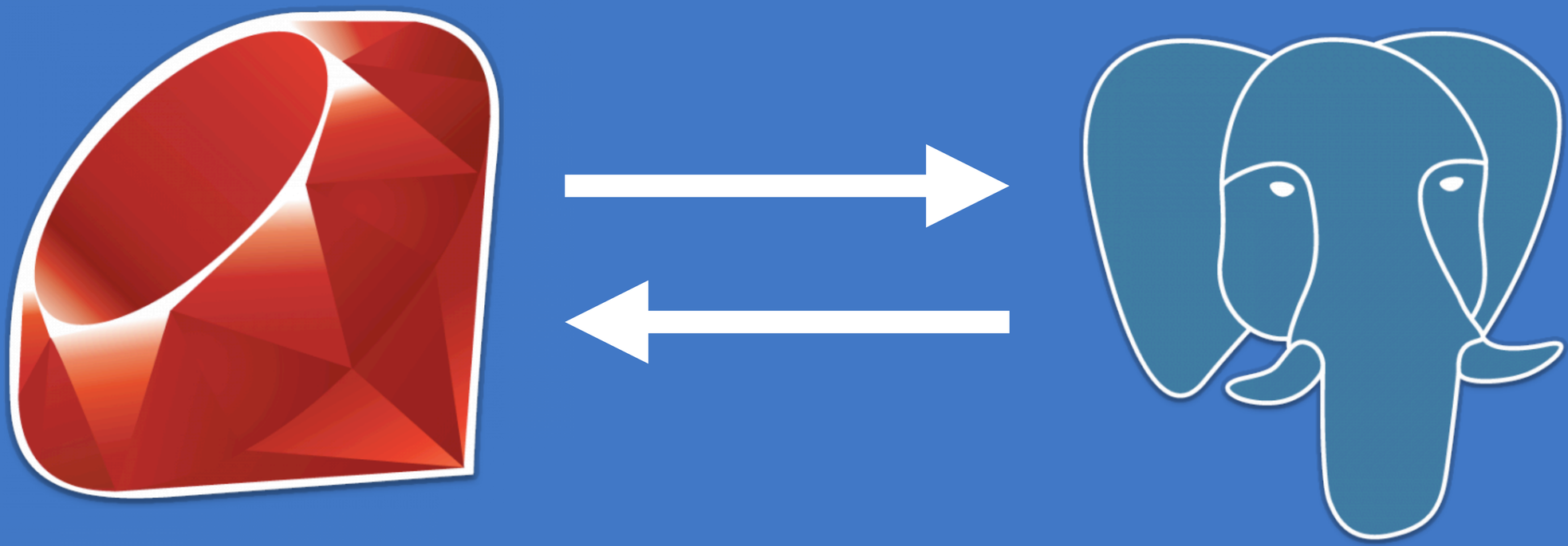
github.com/stankec
@monorkin
hi@stanko.io
stanko.io

01

EXTENSIONS

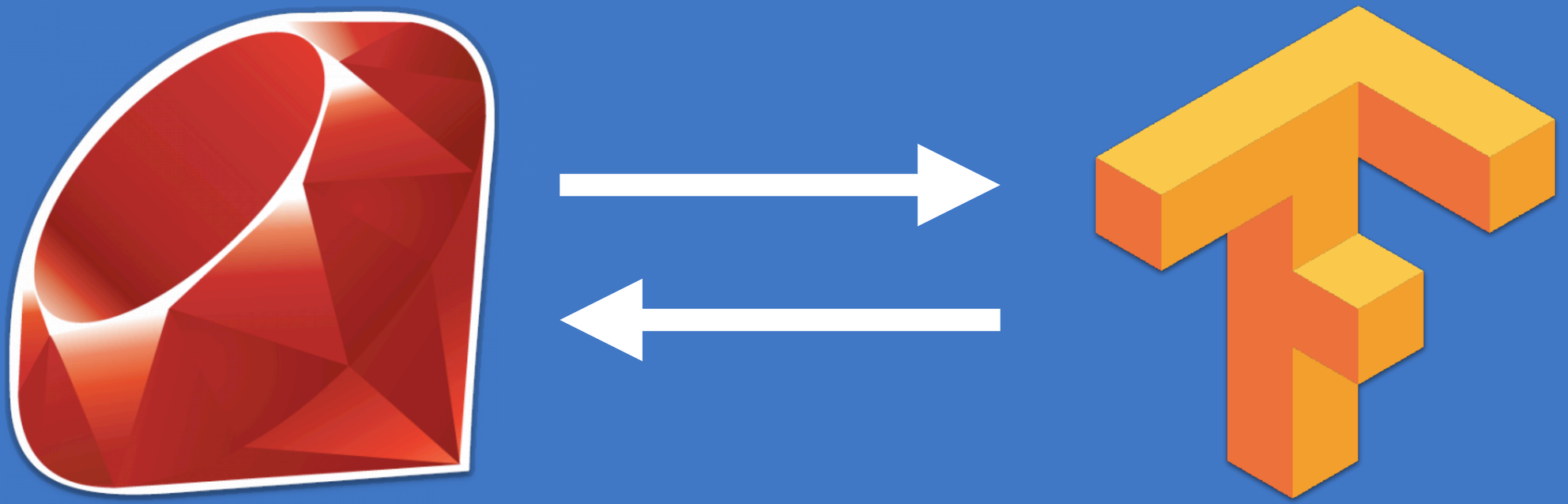


But why? 

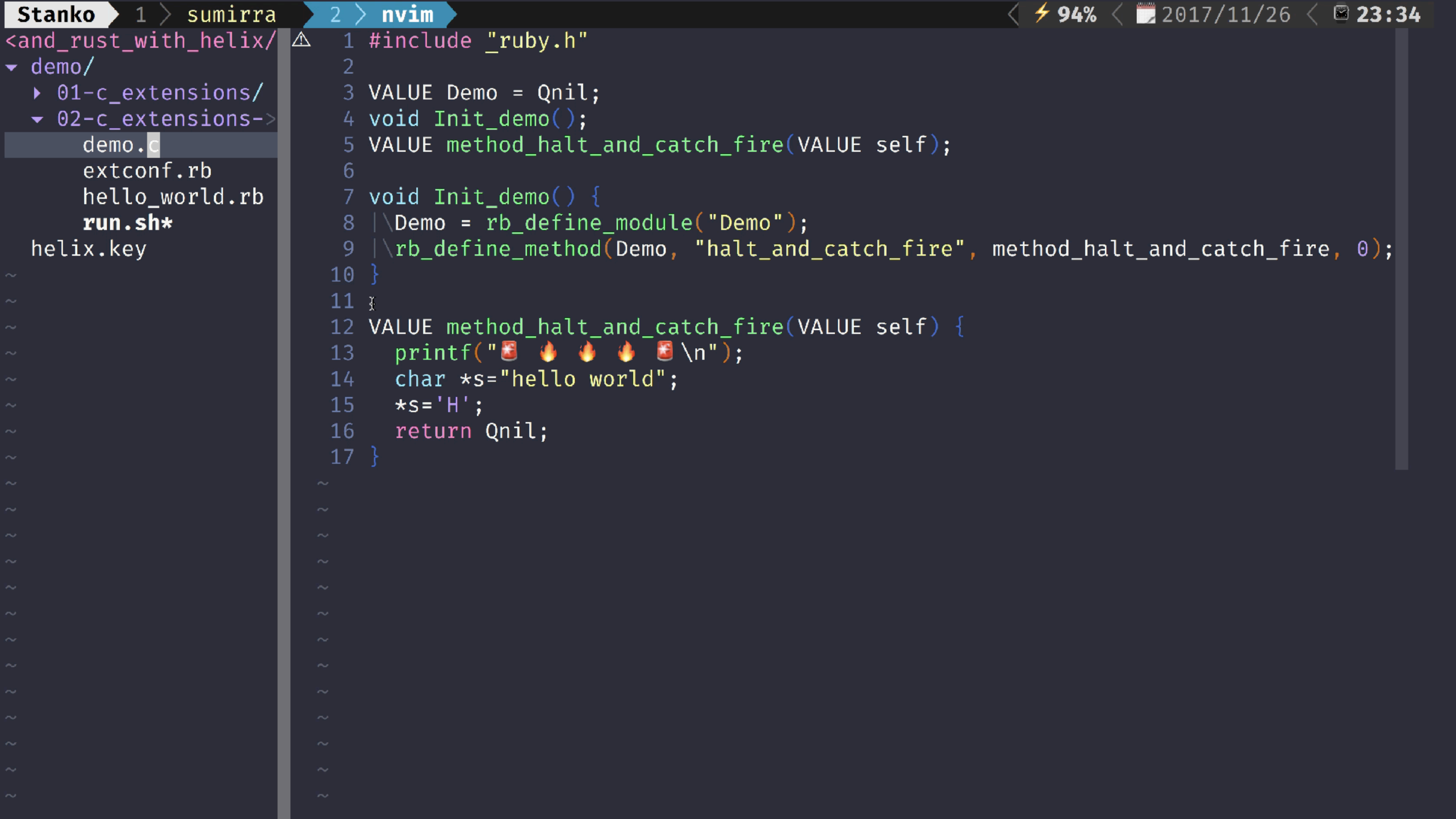


鋸

(That’s Nokogiri’s logo)



```c
#include "ruby.h"

VALUE Demo = Qnil;
void Init_demo();
VALUE method_hello_world(int argc, VALUE *argv, VALUE self);

void Init_demo() {
  Demo = rb_define_module("Demo");
  rb_define_method(Demo, "hello_world", method_hello_world, -1);
}

VALUE method_hello_world(int argc, VALUE *argv, VALUE self) {
  if (argc == 0) {
    printf("Hello World!\n");
    return Qnil;
  }

  if (argc >= 1) {
    VALUE name = argv[0];
    printf("Hello %s\n", StringValueCStr(name));
    return Qnil;
  }

  return Qnil;
}
```


Stanko 1 sumirra 2 nvim 94% 2017/11/26 23:34
<and_rust_with_helix/
demo/
01-c_extensions/
02-c_extensions->
demo.c
extconf.rb
hello_world.rb
run.sh*
helix.key
#include "ruby.h"
VALUE Demo = Qnil;
void Init_demo();
VALUE method_halt_and_catch_fire(VALUE self);
void Init_demo() {
\Demo = rb_define_module("Demo");
\rb_define_method(Demo, "halt_and_catch_fire", method_halt_and_catch_fire, 0);
}
VALUE method_halt_and_catch_fire(VALUE self) {
printf("🚨 🔥 🔥 🔥 🚨\n");
char *s="hello world";
*s='H';
return Qnil;
}

```ruby
# frozen_string_literal: true

require_relative 'demo'
include Demo

begin
  Demo.halt_and_catch_fire
rescue
  puts 'All is good'
ensure
  puts "The VM didn't crash this time"
end
```

02

RUST


hi@stanko.io
Leave feedba on:
https://goo /EXd


REVERSING LABS

Why do people use for Rust when writing FFI?

SAFETY

A program written in Rust can’t crash unexpectedly

Memory safety

(Data) Race condition safe

Performance

Results / Options

Ownership

Borrowing

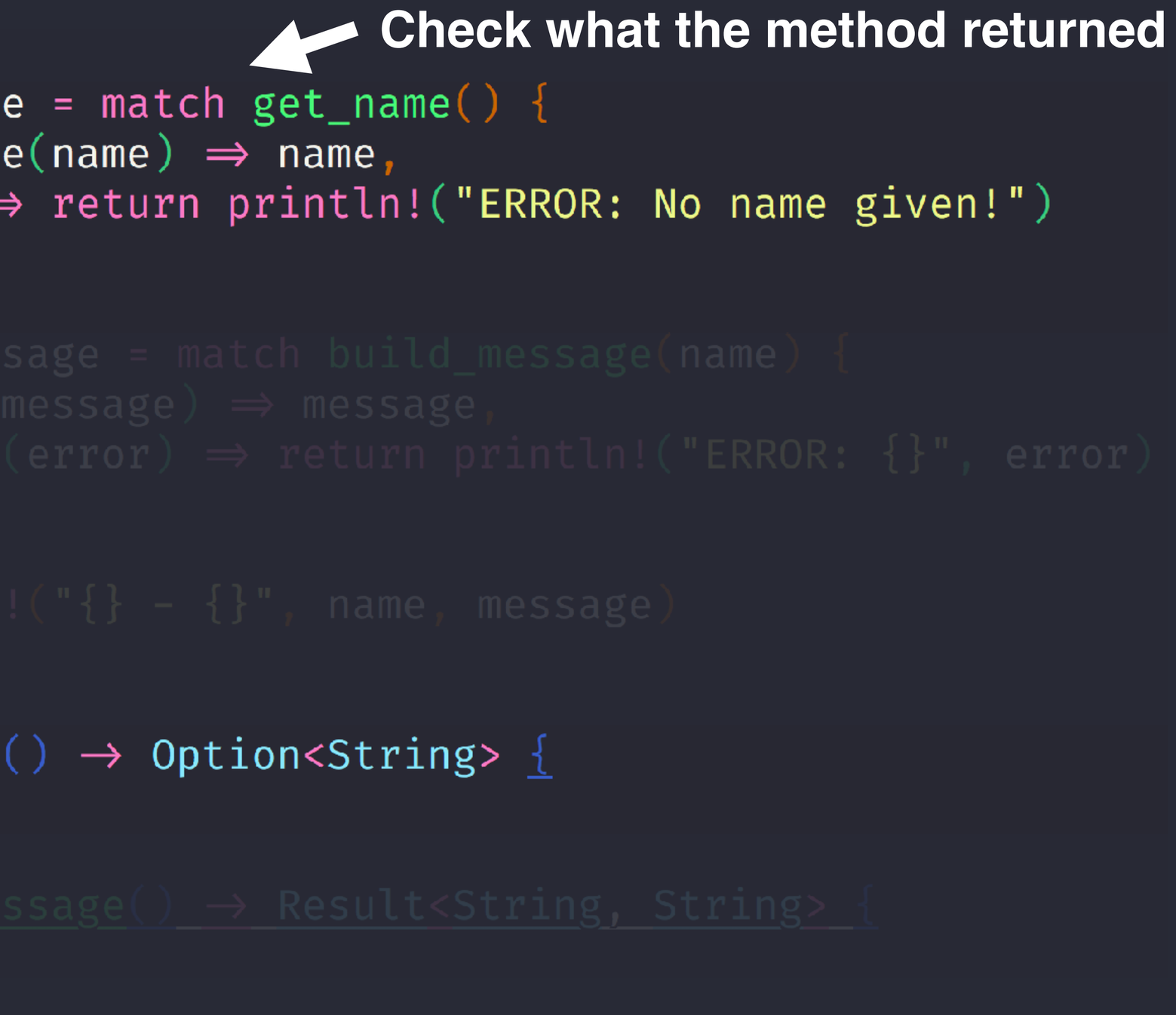
Lifetimes

```rust
pub enum Option<T> {
    None,
    Some(T),
}

pub enum Result<T, E> {
    Ok(T),
    Err(E),
}
```

```rust
fn main() {
    let name = match get_name() {
        Some(name) ⇒ name,
        _ ⇒ return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) ⇒ message,
        Err(error) ⇒ return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() → Option<String> {
}

fn build_message() → Result<String, String> {
}
```

```rust
fn main() {
    let name = match get_name() {
        Some(name) => name,
        _ => return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) => message,
        Err(error) => return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() -> Option<String> {
}

fn build_message() -> Result<String, String> {
}
```

This method can return a string or nothing


Check what the method returned
fn main() {
let name = match get_name() {
Some(name) ⇒ name,
_ ⇒ return println!("ERROR: No name given!")
};
let message = match build_message(name) {
Ok(message) ⇒ message,
Err(error) ⇒ return println!("ERROR: {}", error)
};
println!("{} - {}", name, message)
}
fn get_name() → Option<String> {
}
fn build_message() → Result<String, String> {
}

```rust
fn main() {
    let name = match get_name() {
        Some(name) => name,
        _ => return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) => message,
        Err(error) => return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() -> Option<String> {
}

fn build_message() -> Result<String, String> {
}
```

If a string was returned, unwrap it

```rust
fn main() {
    let name = match get_name() {
        Some(name) => name,
        _ => return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) => message,
        Err(error) => return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() -> Option<String> {
}

fn build_message() -> Result<String, String> {
}
```

If a string wasn't returned, Error

```rust
fn main() {
    let name = match get_name() {
        Some(name) => name,
        _ => return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) => message,
        Err(error) => return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() -> Option<String> {
}

fn build_message() -> Result<String, String> {
}
```

```rust
fn main() {
    let name = match get_name() {
        Some(name) => name,
        _ => return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) => message,
        Err(error) => return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() -> Option<String> {
}

fn build_message() -> Result<String, String> {
}
```

Results have two types

```rust
fn main() {
    let name = match get_name() {
        Some(name) => name,
        _ => return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) => message,
        Err(error) => return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() -> Option<String> {
}

fn build_message() -> Result<String, String> {
}
```

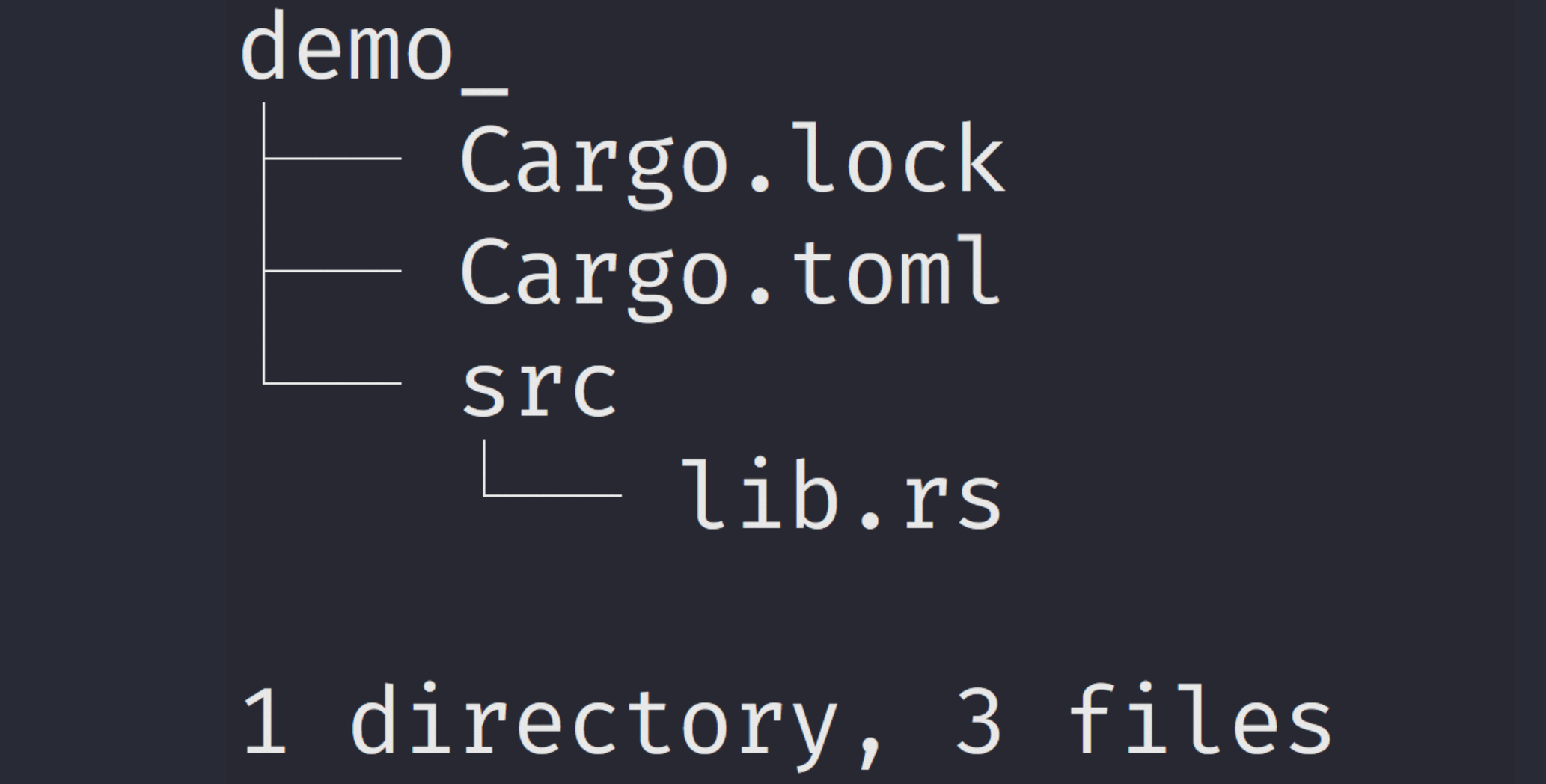
Checks if the result was Ok or an Error

```rust
fn main() {
    let name = match get_name() {
        Some(name) ⇒ name,
        _ ⇒ return println!("ERROR: No name given!")
    };

    let message = match build_message(name) {
        Ok(message) ⇒ message,
        Err(error) ⇒ return println!("ERROR: {}", error)
    };

    println!("{} - {}", name, message)
}

fn get_name() → Option<String> {
}

fn build_message() → Result<String, String> {
}
```

03

HELIX

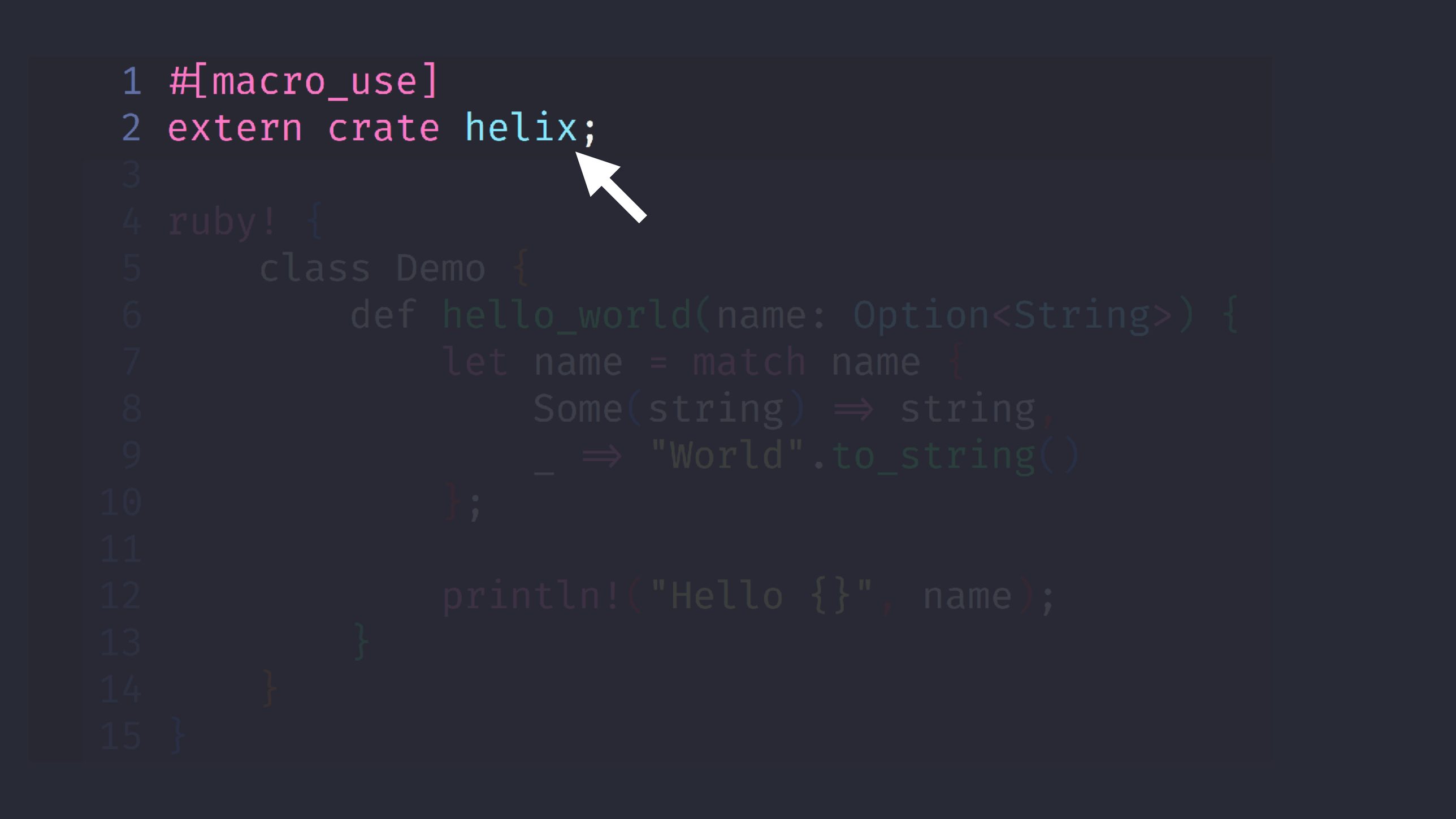
```
cargo new demo
```


demo_
Cargo.lock
Cargo.toml
src
lib.rs
1 directory, 3 files

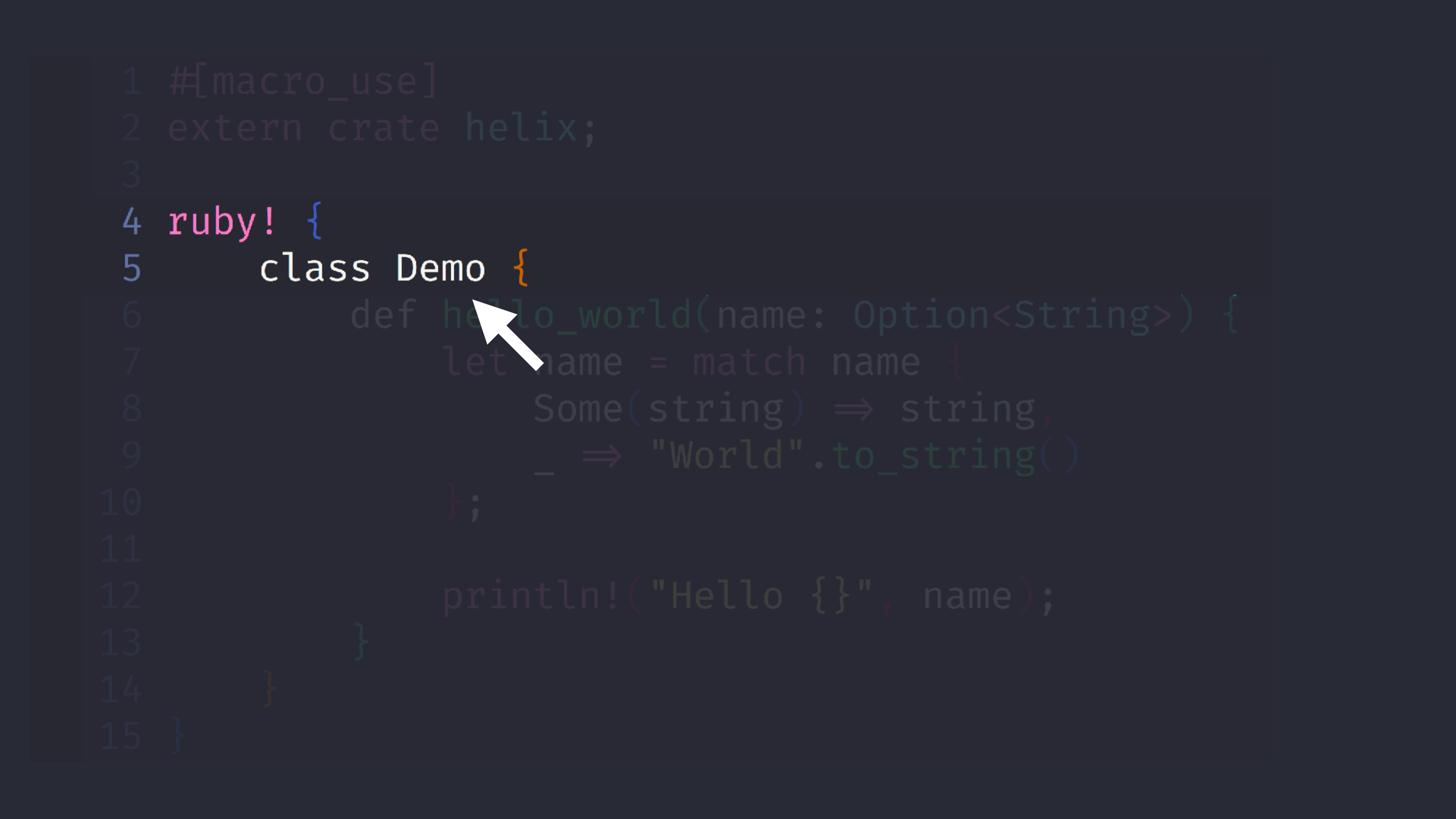
```toml
[package]
name = "demo"
version = "0.1.0"
authors = ["Stanko Krtalić <stanko.krtalic@gmail.com>"]

[lib]
crate-type = ["cdylib"]

[dependencies]
helix = "0.7.2"
```

```toml
[package]
name = "demo"
version = "0.1.0"
authors = ["Stanko Krtalić <stanko.krtalic@gmail.com>"]

[lib]
crate-type = ["cdylib"]

[dependencies]
helix = "0.7.2"
```

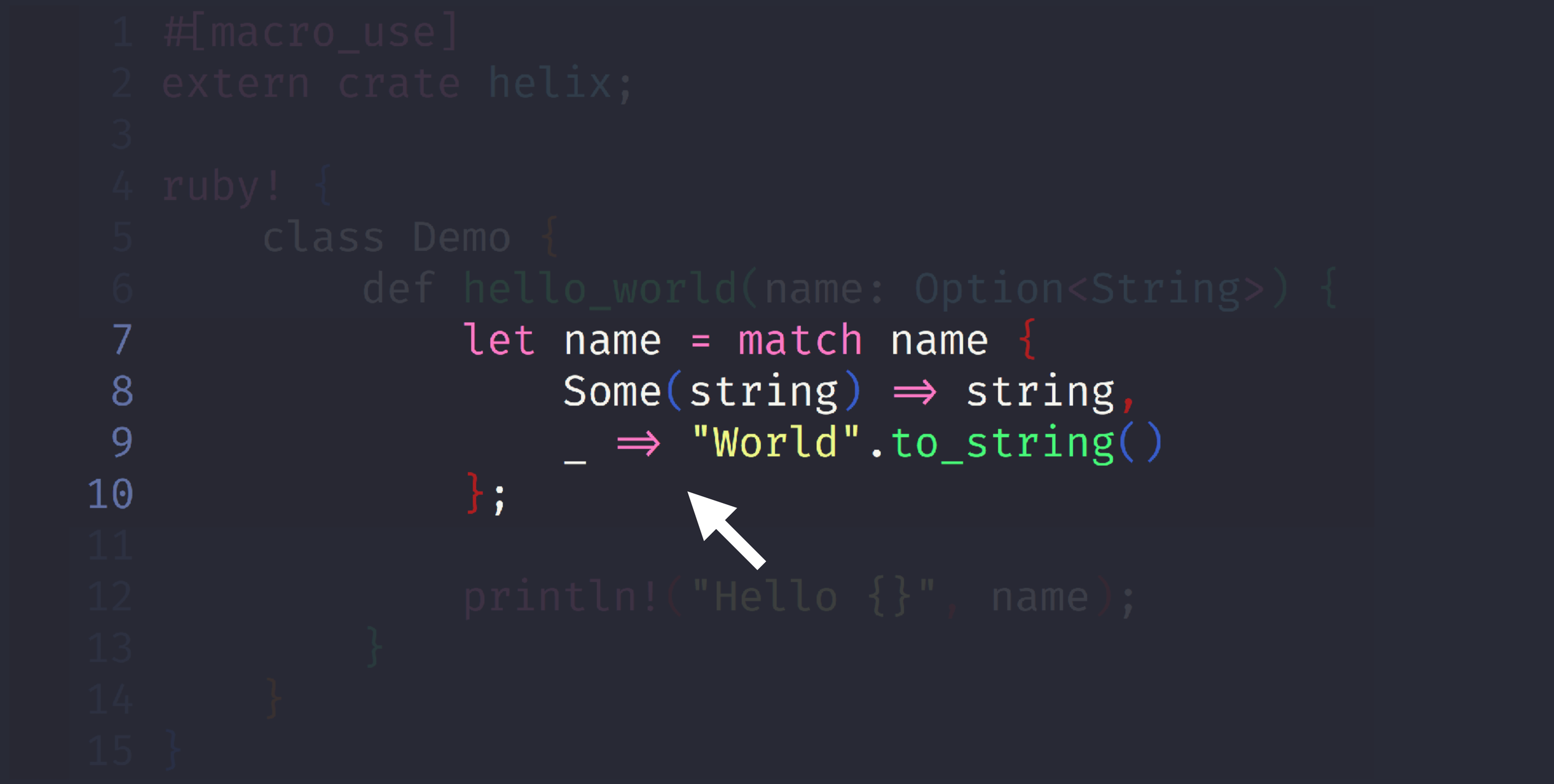
```rust
#[macro_use]
extern crate helix;

ruby! {
    class Demo {
        def hello_world(name: Option<String>) {
            let name = match name {
                Some(string) => string,
                _ => "World".to_string()
            };

            println!("Hello {}", name);
        }
    }
}
```


#[macro_use]
extern crate helix;
ruby! {
class Demo {
def hello_world(name: Option<String>) {
let name = match name {
Some(string) => string,
_ => "World".to_string()
};
println!("Hello {}", name);
}
}
}

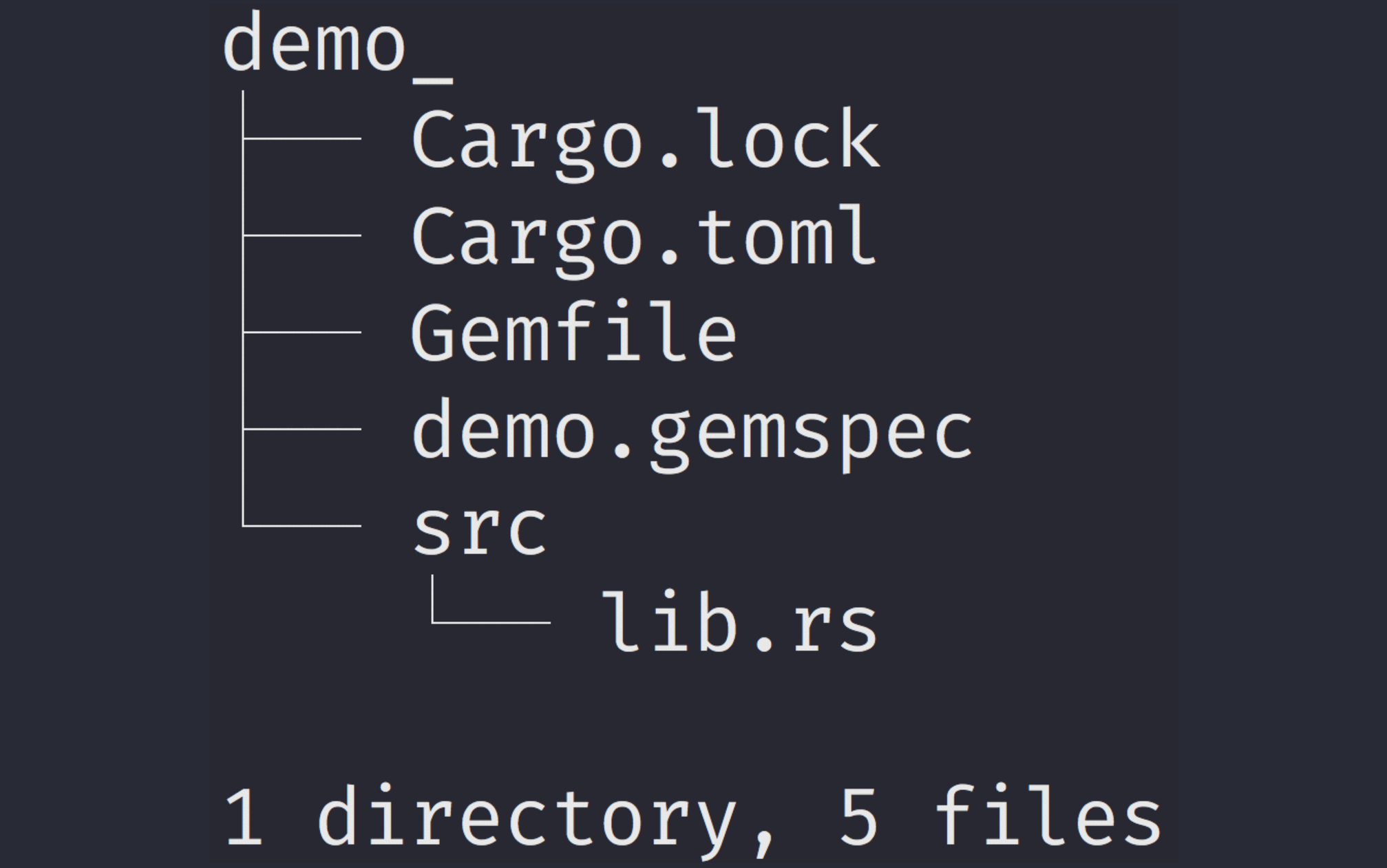
```rust
#[macro_use]
extern crate helix;

ruby! {
    class Demo {
        def hello_world(name: Option<String>) {
            let name = match name {
                Some(string) => string,
                _ => "World".to_string()
            };

            println!("Hello {}", name);
        }
    }
}
```


#[macro_use]
extern crate helix;
ruby! {
class Demo {
def hello_world(name: Option<String>) {
let name = match name {
Some(string) ⇒ string,
_ ⇒ "World".to_string()
};
println!("Hello {}", name);
}
}
}


#[macro_use]
extern crate helix;
ruby! {
class Demo {
def hello_world(name: Option<String>) {
let name = match name {
Some(string) => string,
_ => "World".to_string()
};
println!("Hello {}", name);
}
}
}


#[macro_use]
extern crate helix;
ruby! {
class Demo {
def hello_world(name: Option<String>) {
let name = match name {
Some(string) ⇒ string,
_ ⇒ "World".to_string()
};
println!("Hello {}", name);
}
}
}


#[macro_use]
extern crate helix;
ruby! {
class Demo {
def hello_world(name: Option<String>) {
let name = match name {
Some(string) => string,
_ => "World".to_string()
};
println!("Hello {}", name);
}
}
}


demo_
Cargo.lock
Cargo.toml
Gemfile
demo.gemspec
src
lib.rs
1 directory, 5 files


demo_
Cargo.lock
Cargo.toml
Gemfile
demo.gemspec
src
lib.rs
1 directory, 5 files

```ruby
Gem::Specification.new do |s|
  s.name = 'demo'
  s.version = '1.0.0'
  s.authors = ['Stanko Krtalic Rusendic <stanko.krtalic@gmail.com>']
  s.summary = "A Helix demo project"
  s.files = Dir['{lib/**/*,[A-Z]*}']

  s.platform = Gem::Platform::RUBY
  s.require_path = 'lib'

  s.add_dependency 'helix_runtime', '~> 0.7.2'
end
```

```ruby
Gem::Specification.new do |s|
  s.name = 'demo'
  s.version = '1.0.0'
  s.authors = ['Stanko Krtalic Rusendic <stanko.krtalic@gmail.com>']
  s.summary = "A Helix demo project"
  s.files = Dir['{lib/**/*,[A-Z]*}']

  s.platform = Gem::Platform::RUBY
  s.require_path = 'lib'

  s.add_dependency 'helix_runtime', '~> 0.7.2'
end
```

```ruby
Gem::Specification.new do |s|
  s.name = 'demo'
  s.version = '1.0.0'
  s.authors = ['Stanko Krtalic Rusendic <stanko.krtalic@gmail.com>']
  s.summary = "A Helix demo project"
  s.files = Dir['{lib/**/*,[A-Z]*}']

  s.platform = Gem::Platform::RUBY
  s.require_path = 'lib'

  s.add_dependency 'helix_runtime', '~> 0.7.2'
end
```

```
rake build
```

```ruby
# Optional and KV arguments are not yet supported
Demo.hello_world(nil)
Demo.hello_world('Ruby Zagreb!')

begin
  Demo.hello_world(Object.new)
rescue TypeError
  puts 'Rescued from TypeError'
end
```


" Press ? for help
.. (up a dir)
<d_rust_with_helix/
[✗]demo/
01-c_extension>
02-c_extension>
03-rust_exampl>
[✗]04-helix/
[✗]crates/de>
lib/
src/
lib.rs
[✗]target/
Cargo.lock
Cargo.toml
demo.gemsp>
Gemfile
Gemfile.lo>
Rakefile
tags
NERD
#[macro_use]
extern crate helix;
ruby! {
class Demo {
def hello_world(name: Option<String>) {
let name = match name {
Some(string) ⇒ string,
_ ⇒ "World".to_string()
};
println!("Hello {}", name);
}
}
}
N…
←helix/crates/demo/src/lib.rs
6% ≡
1: 1

```
rails generate helix:crate demo
```

04

WHEN TO USE IT

WORDCOUNT

WordCount uses the `word_count` crate, which defines a very simple class that implements a single class method: `search`. This methods takes a path to a file and a search string as input, performing a count of search string in the source file.

Select a Dataset:
(•) Shakespeare
( ) Upload: Browse... No file selected.

Search Text:

Count

Source

- Crate: Class / Tests
- Controller
- View

INLINE CSS

InlineCSS uses the `inline_css` crate, which uses Servo's `cssparser` and `html5ever` crates to inline a CSS file into an HTML file.

HTML

```
<html id="facebook" class="" lang="en"><head><meta charset="utf-8"><meta name="referrer"
content="origin-when-crossorigin" id="meta_referrer"><script>window._cstart=+new Date();
</script>
<script>function envFlush(a){function b(c){for(var d in
a)c[d]=a[d]}if(window.requireLazy)window.requireLazy(["Env"],b);else{window.Env=window.Env||{};
b(window.Env)}}envFlush({"ajaxpipe_token":"AXjzeWeAHBiMjj1k","timeslice_heartbeat_config":
{"pollIntervalMs":33,"idleGapThresholdMs":60,"ignoredTimesliceNames":
{"requestAnimationFrame":true,"Event listenHandler mousemove":true,"Event listenHandler
mouseover":true,"Event listenHandler mouseout":true,"Event listenHandler
```

CSS

```
.fbEmuTracking{position:absolute;visibility:hidden}
.sp_fWm8gQH0FqP_2x.sx_0c5538{background-position:0
-376px}.sp_fWm8gQH0FqP_2x.sx_41c7c8{background-position:-17px
-376px}.sp_fWm8gQH0FqP_2x.sx_358d9e{background-position:-34px
-376px}.sp_fWm8gQH0FqP_2x.sx_83afbe{background-position:0
-393px}.sp_fWm8gQH0FqP_2x.sx_4dc3ee{background-position:-17px
-393px}.sp_fWm8gQH0FqP_2x.sx_85fa58{width:20px;height:20px;background-position:0
-321px}.sp_fWm8gQH0FqP_2x.sx_44115a{background-position:-34px
-393px}.sp_fWm8gQH0FqP_2x.sx_80a83f{width:20px;height:20px;background-position:-21px
```

Inline

No KV or optional arguments

Can’t write Modules

No cross-compilation

Non-standard compiler*

Safety and speed

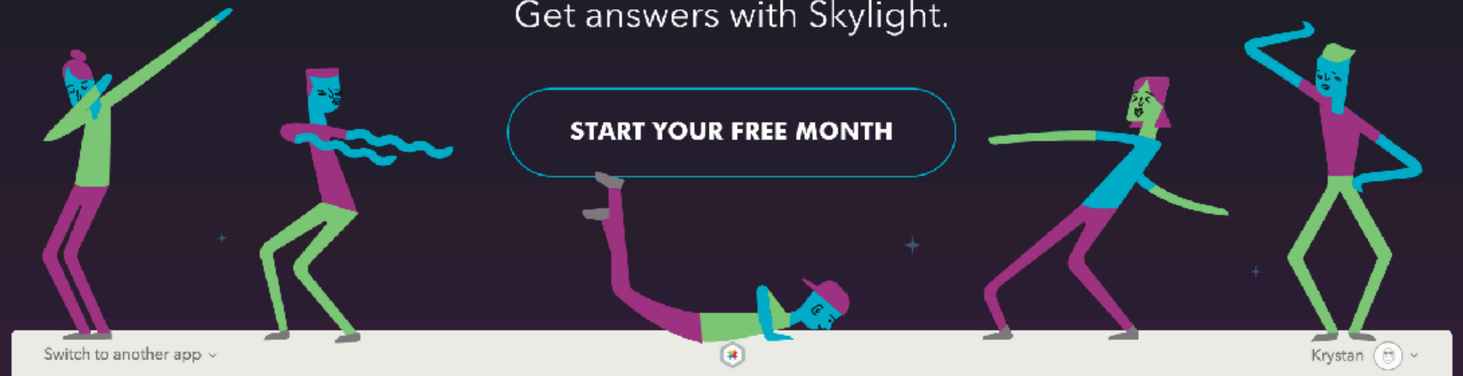

CANONICAL
wire
Dropbox
npm
SENTRY
redhat
mozilla
coursera
Braintree

DO THE ANSWER DANCE.

Don't struggle to learn why your app is slow.

Get answers with Skylight.

START YOUR FREE MONTH



QUESTIONS

 github.com/stankec

 @monorkin

 hi@stanko.io

05

EXTRAS

```
1
2  fn sign_in(username: String, password: String) → Result<User, String> {
3      if username == "user".to_string() && password == "password".to_string() {
4          return Ok(
5              User::new(username, password)
6          );
7      }
8
9      return Err("No such user found");
10 }
```

Username and password are deallocated

```
fn main {
    let name = "petar".to_string();
    let amne = shuffle(name); // ⟵ The function shuffle became the owner of
                              //     the name variable

    println!("Original: {}", name); // ⟵ Errors during compilation
                                    //  We tried to use a variable we don't own
    println!("Shuffled: {}", amne);
}
```