Embedding mruby is easy

Arek Turlewicz

Full stack engineer

Shippio

@a_turl 19-05-2023

BAYSIDE Tech Nite

International after party at RubyKaigi 2023

Money Forward × Shippio

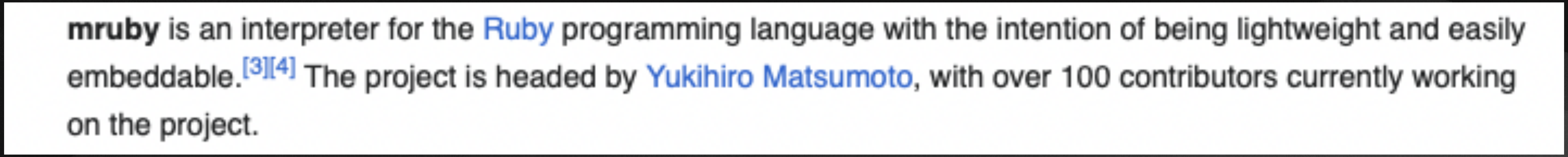
Overview

- What is mruby?
- Do we need one more Ruby implemenation?
- Customization with mruby gems
- Embedding in C programs
- Embedding mruby in Swift

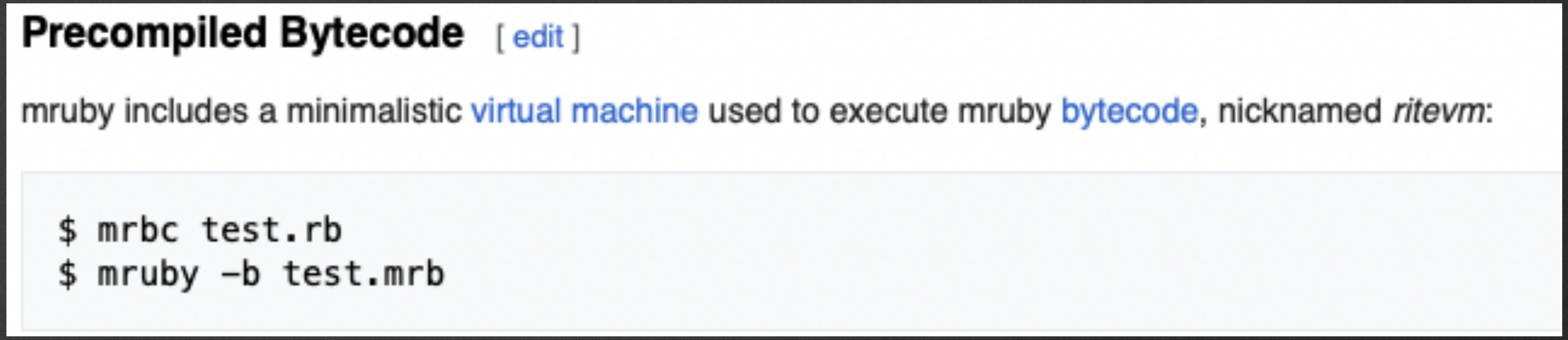
Disclaimer !!!

- I'm not C developer
- Don't use example code in production

What is mruby?



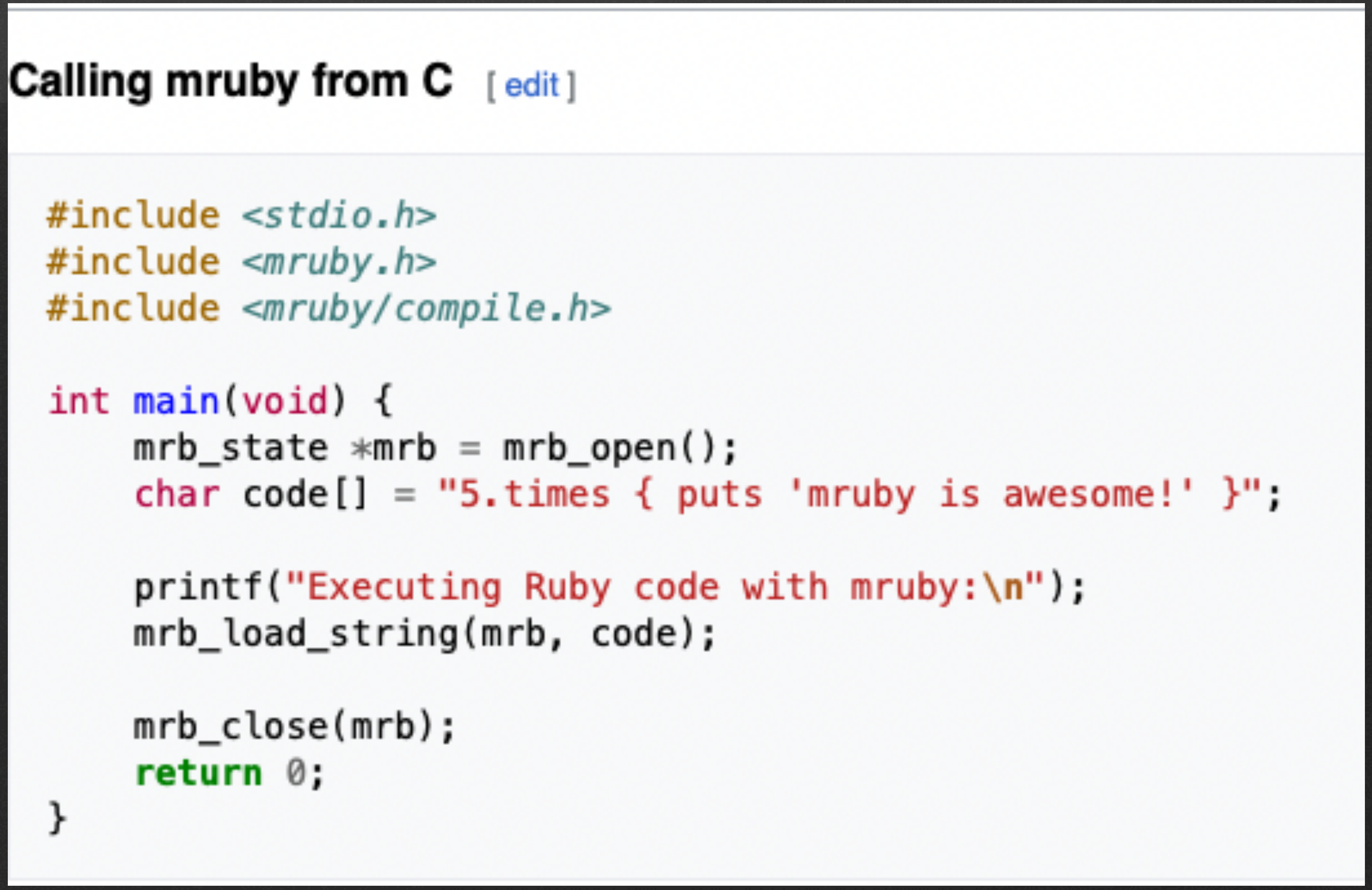
mruby is an interpreter for the Ruby programming language with the intention of being lightweight and easily embeddable.[3][4] The project is headed by Yukihiro Matsumoto, with over 100 contributors currently working on the project.



Precompiled Bytecode [edit]

mruby includes a minimalistic virtual machine used to execute mruby bytecode, nicknamed *ritevm*:

```
$ mrbc test.rb
$ mruby -b test.mrb
```

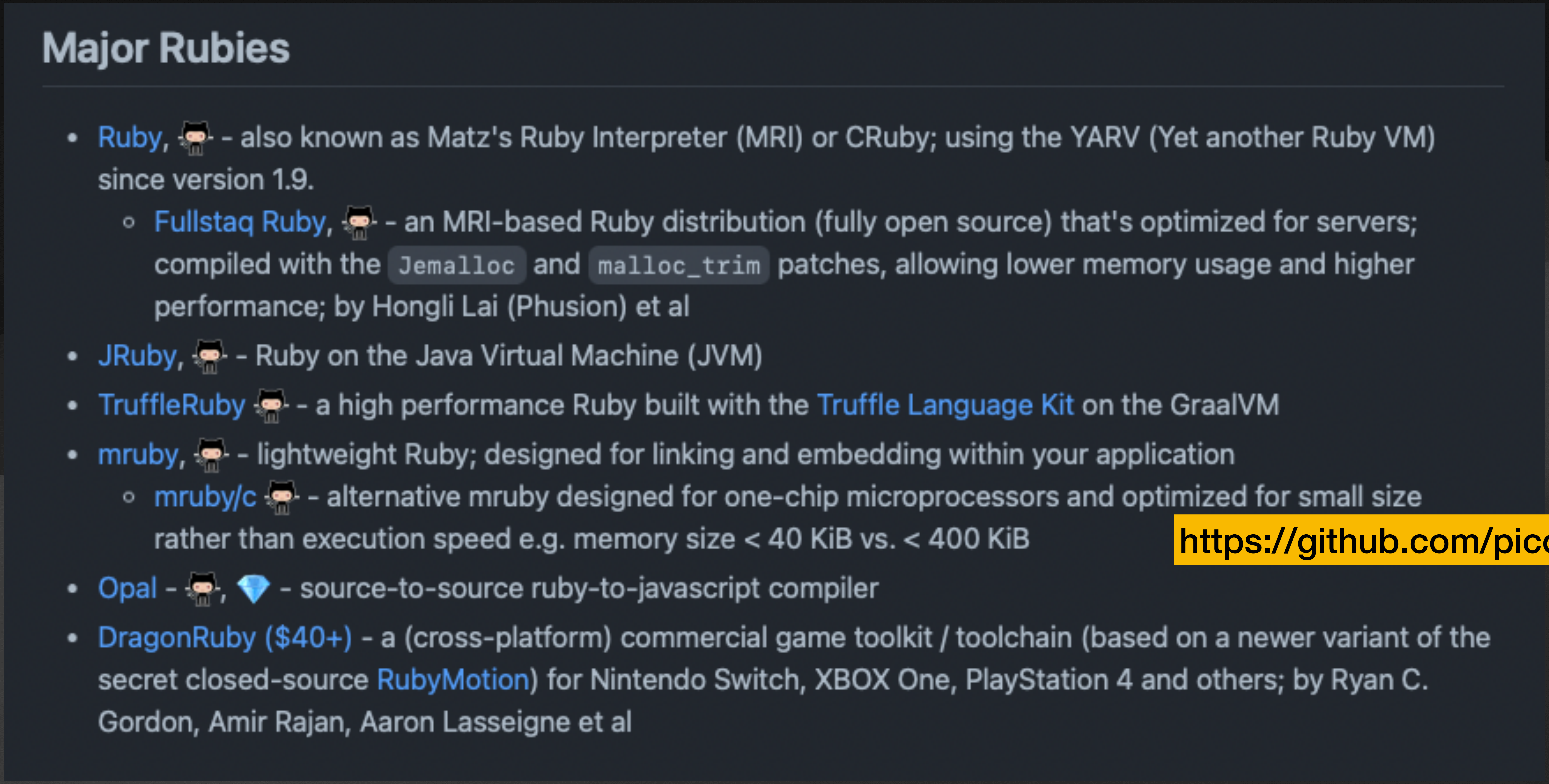


Calling mruby from C [edit]

```
#include <stdio.h>
#include <mruby.h>
#include <mruby/compile.h>

int main(void) {
    mrb_state *mrb = mrb_open();
    char code[] = "5.times { puts 'mruby is awesome!' }";

    printf("Executing Ruby code with mruby:\n");
    mrb_load_string(mrb, code);

    mrb_close(mrb);
    return 0;
}
```

https://en.wikipedia.org/wiki/Mruby

Do we need one more Ruby implementation



Major Rubies

- Ruby, - also known as Matz's Ruby Interpreter (MRI) or CRuby; using the YARV (Yet another Ruby VM) since version 1.9.
  - Fullstaq Ruby, - an MRI-based Ruby distribution (fully open source) that's optimized for servers; compiled with the `Jemalloc` and `malloc_trim` patches, allowing lower memory usage and higher performance; by Hongli Lai (Phusion) et al
- JRuby, - Ruby on the Java Virtual Machine (JVM)
- TruffleRuby - a high performance Ruby built with the Truffle Language Kit on the GraalVM
- mruby, - lightweight Ruby; designed for linking and embedding within your application
  - mruby/c - alternative mruby designed for one-chip microprocessors and optimized for small size rather than execution speed e.g. memory size < 40 KiB vs. < 400 KiB
- Opal - , 💎 - source-to-source ruby-to-javascript compiler
- DragonRuby ($40+) - a (cross-platform) commercial game toolkit / toolchain (based on a newer variant of the secret closed-source RubyMotion) for Nintendo Switch, XBOX One, PlayStation 4 and others; by Ryan C. Gordon, Amir Rajan, Aaron Lasseigne et al

https://github.com/picoruby/picoruby

https://github.com/planetruby/awesome-rubies

Do we need one more Ruby implementation?

- mruby is very modular (you can easily add or remove functionality)
- what you select as a gem is actually compiled into main libruby.a
- there is no **"require"**, everything is build-in into main interpreter binary

Quick start

```
➜ git clone https://github.com/mruby/mruby.git
➜ cd mruby
➜ make
➜ ./bin/mirb
mirb - Embeddable Interactive Ruby Shell

> puts "hello..."
hello...
 => nil
```

Customization with Gems

open build_config/default.rb

```
diff --git a/build_config/default.rb b/build_config/
default.rb
index 9770e5732..a9bad8751 100644
--- a/build_config/default.rb
+++ b/build_config/default.rb
@@ -7,6 +7,7 @@ MRuby::Build.new do |conf|
   conf.gem 'examples/mrbgems/c_extension_example' do |g|
     g.cc.flags << '-g' # append cflags in this gem
   end
+  conf.gem 'examples/mrbgems/ruby_extension_example'
```

uncomment ruby_extension_example gem

Customization with Gems

```
➜ git show
mruby git:(master) ✗
➜ cat examples/mrbgems/ruby_extension_example/mrblib/example.rb
class RubyExtension
  def RubyExtension.ruby_method
    puts "------------ >>>> #{self}: A Ruby Extension"
  end
end
```

```
➜ make
rake
...[build output is here]
➜ ./bin/mirb
mirb - Embeddable Interactive Ruby Shell

> RubyExtension.ruby_method
------------ >>>> RubyExtension: A Ruby Extension
 => nil
```

Embedding in C programs

```c
#include <mruby.h>
#include <mruby/compile.h>

int
main(void)
{
  mrb_state *mrb = mrb_open();
  if (!mrb) { /* handle error */ }

  mrb_load_string(mrb, "puts 'hello world'");
  mrb_load_string(mrb, "a = 42");
  mrb_load_string(mrb, "puts a");
  mrb_close(mrb);
  return 0;
}
```

Embedding in C programs

What will be the output?

Embedding in C programs

If you create folder inside your mruby installation you can just point to ".."

```
➜ gcc -std=c99 -I../include -L../build/host/lib
-lmruby quiz.c -o quiz
➜ ./quiz
hello world
```

Embedding mruby in Swift

- We need to tell Swift how to access c methods
- We need to link mruby library and wrapper function (with interface we will use in Swift)

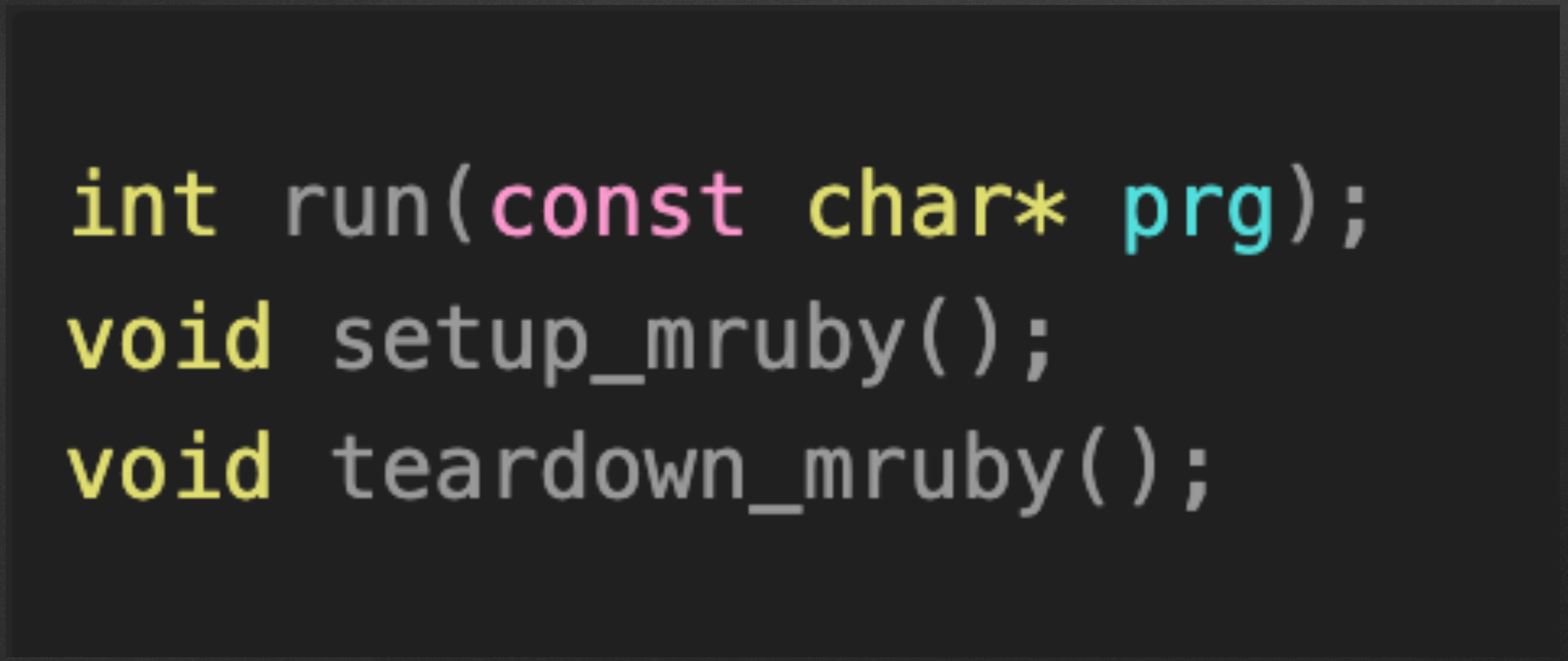
Embedding mruby in Swift

- wrapper.c

```c
#include <mruby.h>
#include <mruby/compile.h>

static mrb_state *mrb;
void setup_mruby()
{
  mrb = mrb_open();
}
void teardown_mruby()
{
  mrb_close(mrb);
}
int run(const char *prg)
{
  if (!mrb) { printf("mrb is not initialized\n"); return 1; }

  mrb_load_string(mrb, prg);
  return 0;
}
```

Embedding mruby in Swift

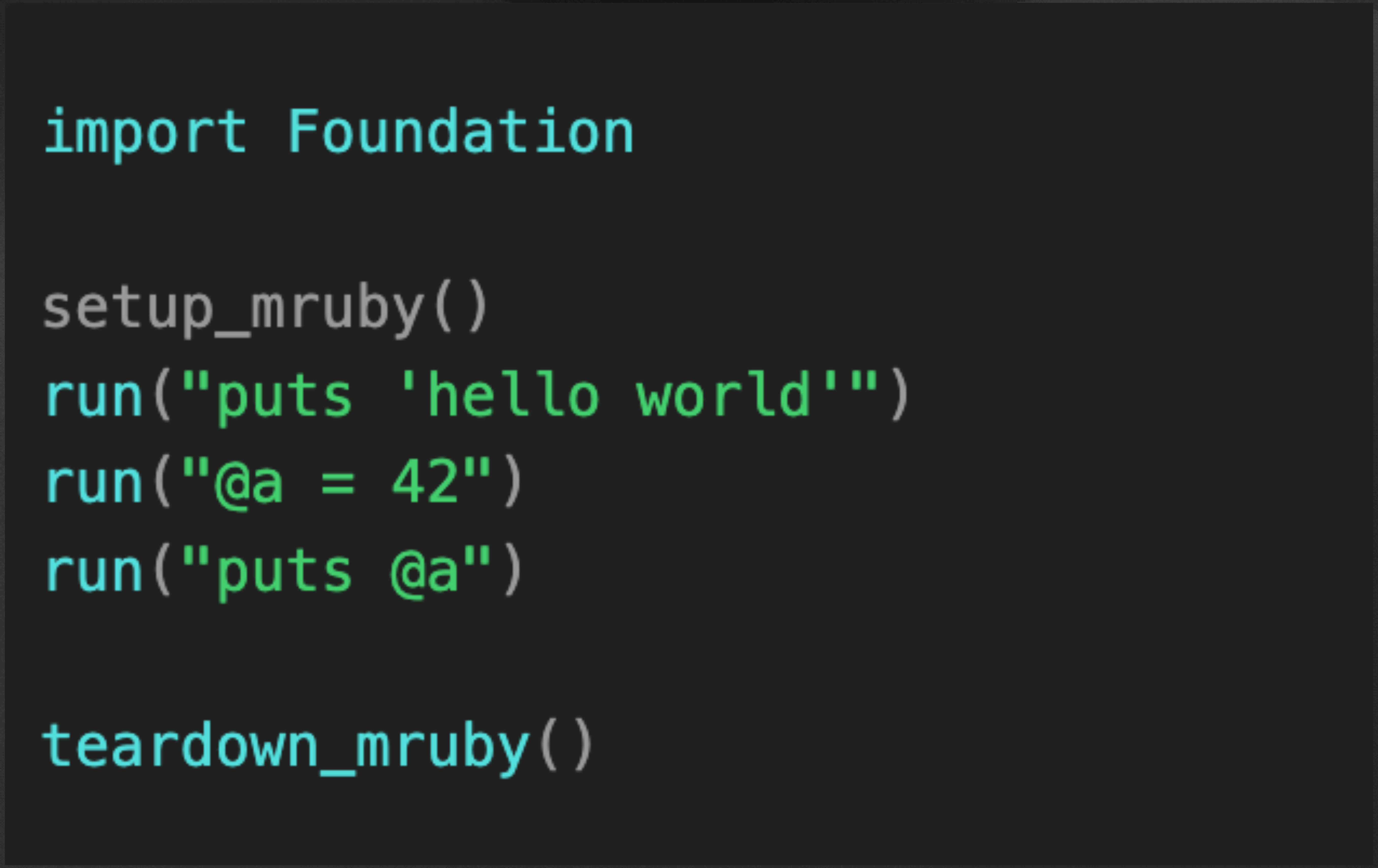
- wrapper-Bridging-Header.h



```c
int run(const char* prg);
void setup_mruby();
void teardown_mruby();
```

Embedding mruby in Swift

- foo.swift



```swift
import Foundation

setup_mruby()
run("puts 'hello world'")
run("@a = 42")
run("puts @a")

teardown_mruby()
```

Embedding mruby in Swift

- linking everything together

```
➜ gcc -c -std=c99 -I../include wrapper.c
swiftc -import-objc-header wrapper-Bridging-Header.h \
  ../build/host/lib/libmruby.a wrapper.o foo.swift -o
foo
➜ ./foo
hello world
42
```

Thank you!