Writing Ruby Web Server for Node.js

a.k.a. I'm as crazy as you think!

肖雪洁 / Xuejie "Rafael" Xiao / @defmacro

Scope

- What I mentioned in the talk description
- A secret announcement for Webruby

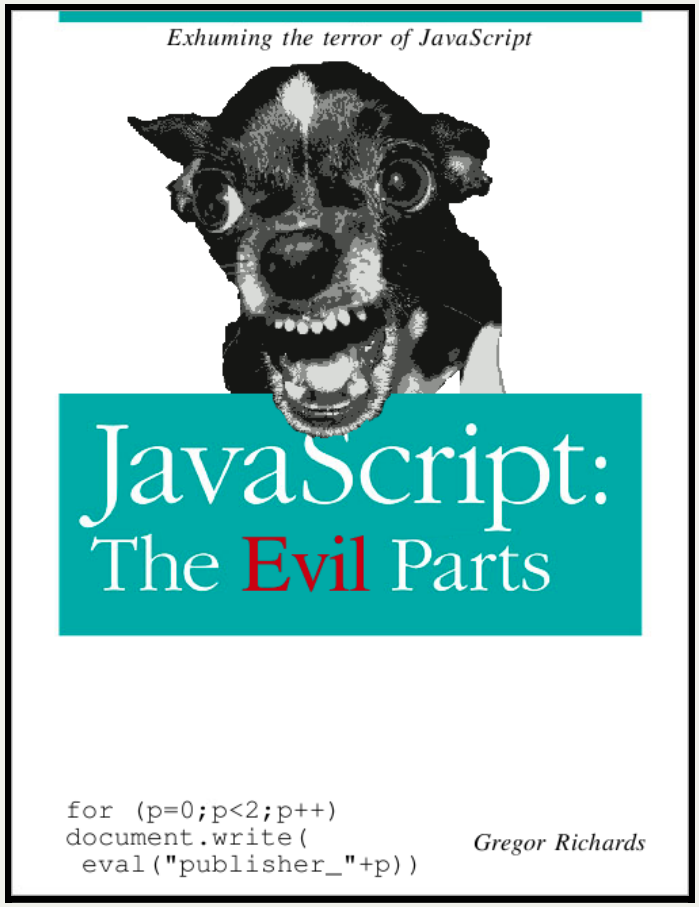
Okay, let's get to our topic!

We all love Ruby(or at least I do)

But we cannot use Ruby everywhere

In modern browsers, we can only use JavaScript

But JavaScript is ...



Do we have a solution for this?

Do we have a solution for this?

Try Webruby!

What is webruby?

- Ruby in the browser
- Based on mruby
- JavaScript Interop

Web IRB

So, how can I use webruby?

This time, let's do something different!

- Node.js instead of browser
- Web server instead of web apps

WARNING: This is just a joke!

Let's get started!

Node.js code

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('RubyConfChina 2013 is awesome!');
}).listen(1337);
console.log('Server running at http://127.0.0.1:1337/');
```

Webruby code

```
MrubyJs.global.require('http').createServer do |req, res|
  res.writeHead(200, {'Content-Type' => 'text/plain'})
  res.end("RubyConfChina 2013 is awesome!\n")
end.listen(1337)
MrubyJs.global.console.log('Server running at http://127.0.0.1:1337/')
```

Sinatra code(for comparison)

```
require 'sinatra'

get '/' do
  'RubyConfChina 2013 is awesome!'
end
```

Result using "ab -n 1000"(this is a joke!)

Version	Time per request(mean)
Webruby	1.718 ms
Node.js	0.241 ms
Ruby(sinatra)	0.890 ms

Okay, I've seen this, but what's inside webruby?

mruby

- Minimal Ruby implementation
- Targets embedded devices
- Highly modular

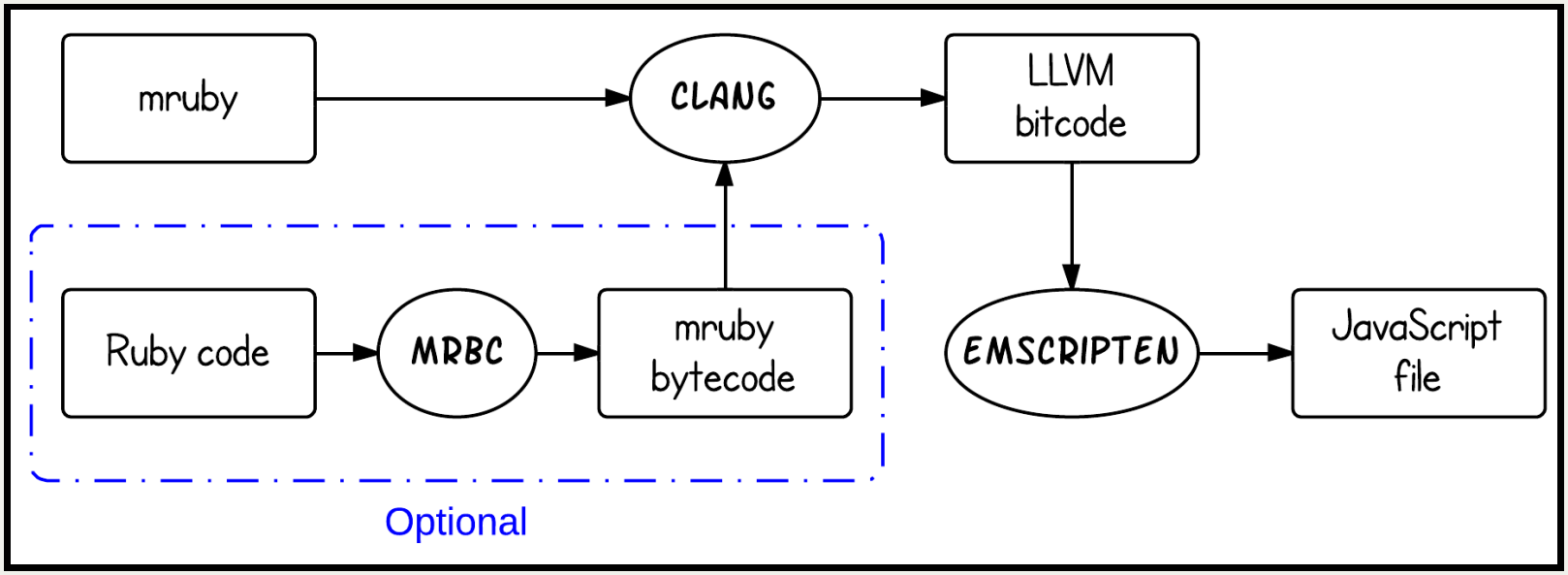
emscripten

- LLVM-to-JavaScript compiler
- Incorporates with clang
- Emits asm.js

asm.js

An optimizable subset of JavaScript, intended primary as a compiler target

The whole flow



JS file size

Version	Raw	gzip -1
O0 with parsing code	3.3M	780K
O0 without parsing code	2.3M	541K
O2 with parsing code	1.4M	379K
O2 without parsing code	863K	252K

Use cases

- Games
  - Replacement for lua
  - Native mruby for iOS(Android), webruby for the web
- Web Apps
  - Ruby for both client and server(sounds familiar?)
  - Or maybe Ruby client + node.js/go server

I strongly believe:

People should have the freedom to use whatever programming languages they enjoy on the browser!

I strongly believe:

People should have the freedom to use whatever programming languages they enjoy on the browser!

(The end of my original slides)

Announcement for Webruby

Demo

Webruby now supports require!

Webruby now supports require!

How does this work?

1: Loop through all Ruby files in the directory.

```
tree app
app
├── another.rb
├── app.rb
└── foo
    └── bar.rb

1 directory, 3 files
```

2: Compile each Ruby file into C array of bytecodes

```
const uint8_t another_irep[] = {
0x52,0x49,0x54,0x45,0x30,0x30,0x30,0x31,0xf1,0x94,0x00,0x00,0x00,0xa6,0x4d,0x
...
const uint8_t app_irep[] = {
0x52,0x49,0x54,0x45,0x30,0x30,0x30,0x31,0xbc,0x55,0x00,0x00,0x00,0x98,0x4d,0x
...
const uint8_t foo_bar_irep[] = {
0x52,0x49,0x54,0x45,0x30,0x30,0x30,0x31,0xb5,0xea,0x00,0x00,0x00,0xa6,0x4d,0x
...
```

3: Attach code for dynamically loading bytecode

```
switch (idx) {
  case 0:
    result = mrb_load_irep(mrb, another_irep);
    break;
  case 1:
    result = mrb_load_irep(mrb, app_irep);
    break;
  case 2:
    result = mrb_load_irep(mrb, foo_bar_irep);
    break;
}
```

And the Ruby part!

```
module Kernel
  @@REQUIRED_MODULES = {
    "another" => 0,
    "app" => 1,
    "foo/bar" => 2
  }

  def require(name)
    return false unless @@REQUIRED_MODULES.include?(name)
    require_internal(@@REQUIRED_MODULES[name])
    @@REQUIRED_MODULES.delete(name)
    true
  end
end
```

This could also work for native mruby!

- The pre-processing result is just a single C file.
- Only one setup function needed.

Limitation

- Only works with a single directory(but can have countless sub-folders)
- No require_relative support yet(adding soon!)

Thank you!
Questions?

- Webruby: https://github.com/xxuejie/webruby