BINGO!

Shugo Maeda

Self introduction

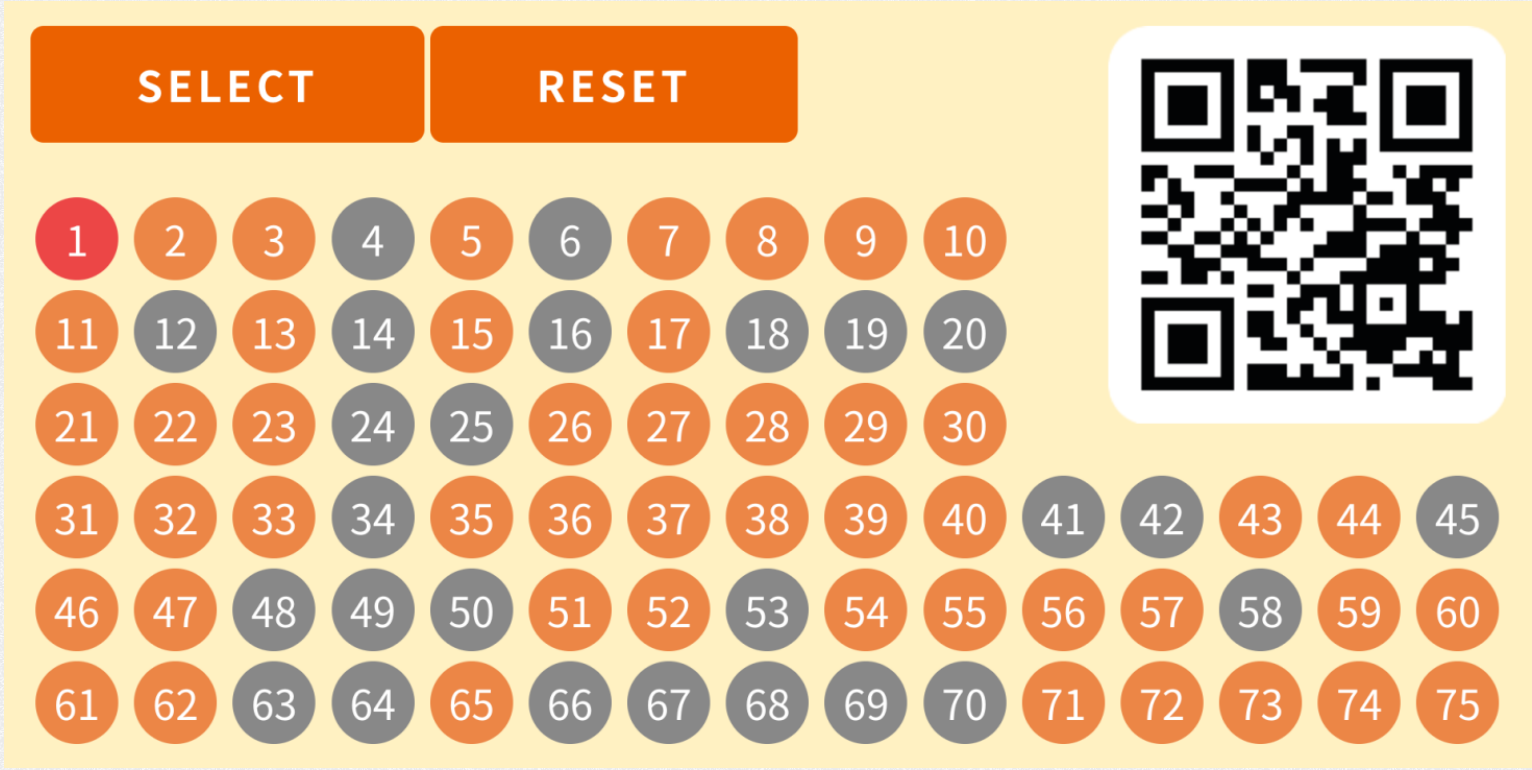
- Shugo Maeda
- The author of Textbringer
- President of NaCl (Network Applied Communication Laboratory Ltd.) since Dec 2022

Lottery at a year-end party

Bingo card

B	I	N	G	O
3	22	33	55	68
6	18	38	52	69
5	27		53	73
4	30	32	60	62
7	20	37	58	74

Bingo machine



Demo

Bingo machine implementation

- Rails
- nginx

Turbo

```
<%= turbo_stream.update "numbers" do %>
<%= render partial: "numbers", locals: { numbers: @numbers } %>
<% end %>
```

Select numbers

```
class Number
  def self.create
    synchronize do
      numbers = read_numbers rescue []
      return numbers if numbers.size >= 75
      candidates = (1..75).to_a - numbers
      numbers.push(candidates.sample)
      write_numbers(numbers)
    end
  end
end
```

Synchronization

```ruby
class Number
  LOCK_FILE_PATH = File.expand_path("tmp/numbers.lock", Rails.root)

  def self.synchronize(&block)
    File.open(LOCK_FILE_PATH, "w") do |f|
      f.flock(File::LOCK_EX)
      block.call
    end
  end
end
```

Atomic update for nginx

```ruby
class Number
  TMP_JSON_FILE_PATH = File.expand_path("tmp/numbers.json", Rails.root)
  JSON_FILE_PATH = File.expand_path("public/numbers.json", Rails.root)









  def self.write_numbers(numbers)
    File.write(TMP_JSON_FILE_PATH, { numbers: numbers }.to_json)
    File.rename(TMP_JSON_FILE_PATH, JSON_FILE_PATH)
    numbers
  end
end
```

rename(2)

If newpath already exists, it will be atomically replaced, so that there is no point at which another process attempting to access newpath will find it missing.

Bingo card implementation

- Rails
- ruby.wasm

Initialization

```
<script type="text/ruby">
  require "js"

  srand(<%= @seed %>)

  def document = JS.global[:document]
```

Bingo card specification

Row B	Row I	Row N	Row G	Row O
(1..15).s-ample(5)	(16..30).sampl-e(5)	(31..45).sampl-e(5)	(46..60).sam-ple(5)	(61..75).sampl-e(5)

- The center space is free

Acknowledgements

2015-03-06

Ruby -「ビンゴカード作成問題」の優秀作品ベスト3を発表します！ #codeiq

Ruby

はじめに

先月、CodeIQにビンゴカード作成問題を出題しました。

CodeIQに「ビンゴカード作成問題」を出題しました。みなさんの挑戦をお待ちしてます！ - give IT a try

このビンゴカード作成問題、ありがたいことに50人もの方が解答を送ってくれました。

挑戦してくださったみなさん、どうもありがとうございました。

Profile

伊藤淳一 (id:Junichilto) PRO

株式会社ソニックガーデンで働くRubyプログラマ。
プログラミングスクール「フィヨルドブートキャンプ」のメンター、および「プロを目指す人のためのRuby入門」の著者。

@jnchitoさんをフォロー

このブログについて

Entry Search

記事を検索

著書・訳書

Using Promise on ruby.wasm

- synchronous code by Fiber

```ruby
def sleep_ms(ms)
  JS.eval("return new Promise((f) => setTimeout(f, #{ms}))").await
end

3.times do
  # do something
  sleep_ms(5000)
end
```

rubyVM.evalAsync

```
<script type="text/ruby">
  def update_loop
    selected = Array.new(5) { [] }
    selected[2][2] = true
    while true
      selected_numbers = JS.global.fetch("/numbers.json?t=#{Time.now.to_i}").
        await.json.await[:numbers].to_s.split(/,/).map(&:to_i)

      ...
</script>
<script type="text/javascript">
  // Promise#await works only under evalAsync
  await window.rubyVM.evalAsync("update_loop")
</script>
```

SystemStackError

```
await window.rubyVM.evalAsync("p (0..3).all? { |i| i.even? }")
```

Fiber stack size is 256kb

- https://github.com/ruby/ruby.wasm/issues/133

> *evalAsync internally uses Fiber, and its stack size 256kb is smaller than main stack (16mb).*

Extend the stack size?

```
const wasi = new WASI({
    args,
    env: {
        "RUBY_FIBER_MACHINE_STACK_SIZE": String(1024 * 1024 * 20),
        ...
    },
    ...
});
```

BINGO!

- Loop unrolling by ERB

```
<%
  is_bingo = [
    *(0..4).map { |i| (0..4).map { |j| "selected[#{i}][#{j}]" }.join("&&") },
    *(0..4).map { |i| (0..4).map { |j| "selected[#{j}][#{i}]" }.join("&&") },
    (0..4).map { |i| "selected[#{i}][#{i}]" }.join("&&"),
    (0..4).map { |i| "selected[#{i}][#{4-i}]" }.join("&&")
  ].join("||")
%>
  if <%== is_bingo %>
    ...
```

Unrolled code

- No more SystemStackError

```
if selected[0][0]&&selected[0][1]&&selected[0][2]&&selected[0][3]&&selected[0][4]||
selected[1][0]&&selected[1][1]&&selected[1][2]&&selected[1][3]&&selected[1][4]||
selected[2][0]&&selected[2][1]&&selected[2][2]&&selected[2][3]&&selected[2][4]||
selected[3][0]&&selected[3][1]&&selected[3][2]&&selected[3][3]&&selected[3][4]||
selected[4][0]&&selected[4][1]&&selected[4][2]&&selected[4][3]&&selected[4][4]||
selected[0][0]&&selected[1][0]&&selected[2][0]&&selected[3][0]&&selected[4][0]||
selected[0][1]&&selected[1][1]&&selected[2][1]&&selected[3][1]&&selected[4][1]||
selected[0][2]&&selected[1][2]&&selected[2][2]&&selected[3][2]&&selected[4][2]||
selected[0][3]&&selected[1][3]&&selected[2][3]&&selected[3][3]&&selected[4][3]||
selected[0][4]&&selected[1][4]&&selected[2][4]&&selected[3][4]&&selected[4][4]||
selected[0][0]&&selected[1][1]&&selected[2][2]&&selected[3][3]&&selected[4][4]||
selected[0][4]&&selected[1][3]&&selected[2][2]&&selected[3][1]&&selected[4][0]
```

Conclusion

- ruby.wasm is awesome
- ERB is a good preprocessor for ruby.wasm