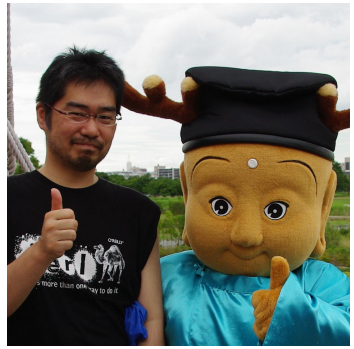
Ruby on Browser

RubyWorld Conference

2024-12-05

とみたまさひろ / @tmtms

自己紹介



- とみたまさひろ
  - https://twitter.com/tmtms
  - https://tmtms.net
- 長野県北部在住
- ９年ぶりに登壇
- 趣味 Ruby / MySQL / メール / 文字化け
- 勤務先 株式会社SmartHR

Ruby歴27年

前世紀から使ってる

今日は仕事の話ではなく趣味の話

普段はRubyで誰の役にも立たないものを作ってる

MySQL/Ruby (mysql)

https://github.com/tmtm/mysql-ruby

- libmysqlclientというライブラリを使って RubyからMySQLを使えるようにしたやつ
- Cで書かれてる
- 要コンパイル
- もうメンテしてない

ruby-mysql

https://gitlab.com/tmtms/ruby-mysql

- Cで書くのがいやになったんでlibmysqlclientを使わずに全部Rubyで作り直したやつ
- コンパイル不要
- だけど遅い
- 最近はみんなmysql2使ってるし
- 誰も使わない
- 自分でも使ってない
- と思ったけど稀に使ってる人がいるっぽい

ruby-mysql2

https://gitlab.com/tmtms/ruby-mysql2

- ruby-mysqlを使ってmysql2互換APIのライブラリを作ってみた
- これも作ってみただけで自分でも使ってない
- 互換はあるはずなんだけどRailsでは使えない
- RailsはGemfile以外にコード中に `gem "mysql2"` って書いてあるので
- `active_record/connection_adapters/mysql2_adapter.rb` を書き換えれば動きそう

車輪の再発明は楽しいですね！

LSP Router

https://gitlab.com/tmtms/lsp_router

- LSP サーバーとして Rubocop LSP と Solargprah を一緒に使いたい
- Emacs は LSP サーバーをひとつしか使えなので作った
- 各 LSP サーバーの Capability に応じてリクエストを振り分ける
- 便利に使ってる

```
server :rubocop do
  command 'rubocop --lsp 2> /tmp/rubocop.err'
end
server :solargraph do
  command 'solargraph stdio 2> /tmp/solargraph.err'
end
```

mrubyを使ってみたやつ

ハードウェア組み込みじゃなくてソフトウェア組み込み言語として

postfix-mruby

https://github.com/tmtm/postfix-mruby

Postfixというメールサーバーに組み込めるやつ
独自のlookupテーブルを作れる

```
class Upcase
  def lookup(key)
    Log.info "input: #{key}" if Log.verbose?
    return key.upcase
  end
end
Upcase.new
```

```
% postmap -q abc mruby:/path/to/upcase.rb
ABC
```

- できたら面白いかなーと思って作ってみたらできた
- 使ってない

mrubyudf

https://github.com/tmtm/mrubyudf

MySQLのユーザー定義関数をRubyで書けるやつ

```ruby
FIXNUM_MAX = 2**62-1
def fib(n)
  a, b = 1, 0
  n.times { a, b = b, a + b }
  raise 'Overflow' if b > FIXNUM_MAX
  b
end
```

```
% mrubyudf fib.spec
% cp fib.so $(mysql_config --plugindir)
% mysql -uroot
mysql> create function fib returns int soname 'fib.so';
mysql> select fib(10);
+---------+
| fib(10) |
+---------+
|      55 |
+---------+
```

- できたら面白いかなーと思って作ってみたらできた
- 使ってない

Linuxデスクトップ環境まわりのやつ

rkremap

https://gitlab.com/tmtms/rkremap

- Linux用のキーリマッパー
- ブラウザ等のテキスト編集にEmacsのキーバインドを実現するために作った
- 自分ではちゃんと使ってる
- xremapという最強のキーリマッパーがあるので普通はそれを使うのがいい

車輪の再発明は楽しいですね！

wmadd

https://gitlab.com/tmtms/wmadd

- Linuxデスクトップでウィンドウが重なったときにアクティブウィンドウの上にあるウィンドウだけを半透明化するやつ
- 最近Kubuntuを使い始めたんだけどKWinだとこの機能がなかったので
- 自分ではちゃんと使ってる

Rubyべんり

個人的に趣味で作ってるやつでも
データベースを使うためのライブラリとか
プログラムに組み込むやつとか
キーリマッパーとかデスクトップツールとか

いろんな用途でRubyを使える

でも

- ブラウザ上でちょっとしたツールを作りたいと思ってもJavaScriptだけ
- またはJavaScriptにトランスパイルする何か
- RubyもOpalというJavaScriptへのトランスパイラはある
- コンパイルとかトランスパイルとかめんどくさい
- .rbファイルをそのまま置くだけで動いてほしい

数年前にruby.wasmが登場してRubyがそのままブラウザ上で動くようになった！

やったー！

https://github.com/ruby/ruby.wasm

Ruby on Browser

ruby.wasm で作ったものをいくつか

文字化けを復元するよ

https://tmtms.net/mojibake/

ruby.wasm で最初に作ったページ
文字化けを復元したり
文字列を文字化けさせたりする

（余裕があったらデモ）

前からこういうのを作りたかったけど
このためにわざわざサーバーは動かしたくなかったし
JavaScriptで実装するのはムリだと思ったので諦めてた

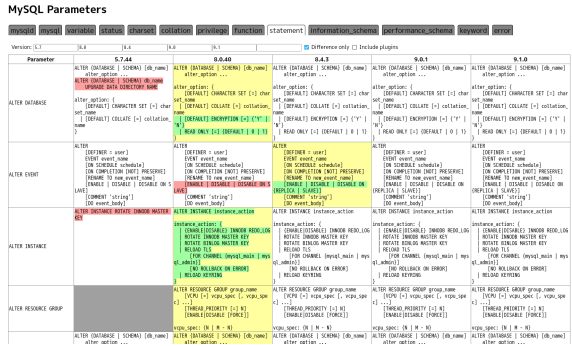
Rubyは文字コード変換処理を内部に持ってるのでできた
Rubyすばらしい

MySQL Parameters

https://mysql-params.tmtms.net/statement/

- めずらしく実用的なやつ
- MySQLのバージョン間の差分を表示する
- ずっと前にVue.jsの練習で作ったやつ
- ruby.wasmで作り直した

（余裕があったらデモ）



Rabbit on Firefox

https://tmtms.net/rabbit/

- Rabbit(https://rabbit-shocker.org) ~~パクリ~~ インスパイア
- Firefox で Reveal.js / PDF / Speaker Deck のスライドを表示しているときにブックマークレットを実行するとウサギが表示される
- もともと JavaScript で作ったやつを作り直した

```
javascript:(()=>{if(typeof rubyVM!='undefined'){rubyVM.eval('start');return};
var d=document;var h=d.getElementsByTagName('head')[0];var s=d.createElement('script');
s.src='https://cdn.jsdelivr.net/npm/@ruby/3.3-wasm-wasi@2.6.2/dist/browser.script.iife.js';
h.appendChild(s);s=d.createElement('script');s.src='https://tmtms.net/rabbit/rabbit.rb';
s.type='text/ruby';h.appendChild(s);})()
```

ブックマークレットはJavaScriptで書かないといけない。残念。

余裕があったら PDF と Speaker Deck もデモ

PDF はこの辺

Speaker Deck はこの辺

ruby.wasm の使い方

HTML内でruby.wasmを読み込む

```
<!DOCTYPE html>
<html>
  <script src="https://cdn.jsdelivr.net/npm/@ruby/3.3-wasm-wasi@2.6.2/dist/browser.script.iife.js">
  </script>
  <script type="text/ruby">
    # ここがRubyスクリプト
    p Time.now  #=> 出力はブラウザのコンソール
    puts "Hello, world!"
  </script>
</html>
```

`<script type="text/ruby">`にRubyを書く

```
<!DOCTYPE html>
<html>
  <script src="https://cdn.jsdelivr.net/npm/@ruby/3.3-wasm-wasi@2.6.2/dist/browser.script.iife.js">
  </script>
  <script type="text/ruby">
    # ここがRubyスクリプト
    p Time.now  #=> 出力はブラウザのコンソール
    puts "Hello, world!"
  </script>
</html>
```

簡単！

`<script type="text/ruby" src=〜>`で別ファイルにできる

```
<!DOCTYPE html>
<html>
  <script src="https://cdn.jsdelivr.net/npm/@ruby/3.3-wasm-wasi@2.6.2/dist/browser.script.iife.js">
  </script>
  <script type="text/ruby" src="hogehoge.rb"></script>
</html>
```

別ファイルになってた方が普通にRubyでデバッグできたりするんでよさそう

Rubyは動くけどブラウザのコンソールに文字を出力することくらいしかできない

HTML要素の操作などにはJavaScriptの機能を使う

JSライブラリ

RubyからJavaScriptを呼ぶための薄いラッパーが用意されてる

```
require 'js'

JS.eval('alert("hoge")')  # JavaScriptを文字列で渡す
JS.global.alert('hoge')   # メソッドでJavaScript関数を呼ぶ
```

- `JS.eval` で JavaScript を実行できる
- `JS.global` 経由で JavaScript のグローバルオブジェクト取得や関数実行ができる

JavaScript の値を Ruby から見るとすべて `JS::Object`

```
n = JS.eval('return 123')     #=> JS::Object (123)
n.typeof                      #=> "number"
s = JS.eval('return "hoge"')  #=> JS::Object ("hoge")
s.typeof                      #=> "string"
```

Ruby では扱いにくいので `to_i` や `to_s` 等で変換できる

```
JS.eval('return 123').to_i     #=> 123
JS.eval('return "hoge"').to_s  #=> "hoge"
```

```
s = JS.eval('return "hoge"')  #=> JS::Object ("hoge")
s[:length]                    #=> JS::Object (4)
s.call(:charAt, 2)            #=> JS::Object ("g")
s.charAT(2)                   #=> JS::Object ("g")
```

- `JS::Object#[]` でプロパティ取得＆設定
- `JS::Object#call(`*func*`)` で JavaScript の関数呼び出し
- `JS::Object#`*func*`()` でも呼び出せる

JavaScript の `null` や `undefined` も `JS::Object` のインスタンス
Ruby で真偽値として評価すると当然真になるので注意

```
JS.eval('return null')       #=> JS::Object (JS::Null)
JS.eval('return undefined')  #=> JS::Object (JS::Undefined)

!!JS::Null       #=> true
!!JS::Undefined  #=> true
```

Promise / await

JavaScript の Proimse は `await` で待てる
`data-eval="async"` の指定が必要

```
<script type="text/ruby" data-eval="async" src="hoge.rb"></script>
```

```
promise = JS.global.fetch("https://tmtms.net")  #=> JS::Object (Promise)
resp = promise.await  #=> JS::Object (Response)
promise = resp.text   #=> JS::Object (Promise)
promise.await         #=> JS::Object ("<!DOCTYPE html>\n<html>\n ....")
```

HTML要素の操作

ほぼ JavaScript

```
document = JS.global[:document]
hoge = document.getElementById('hoge')
fuga = document.createElement('div')
fuga[:id] = 'fuga'
hoge.appendChild(fuga)
```

HTML要素の`onclick`等のイベント設定では
Rubyのグローバルコンテキストであれば`rubyVM.eval()`を使える

```
<input id="b" type="button" onclick="rubyVM.eval('hoge')">
<script type="text/ruby">
  require 'js'
  def hoge
    JS.global.alert('hoge')
  end
</script>
```

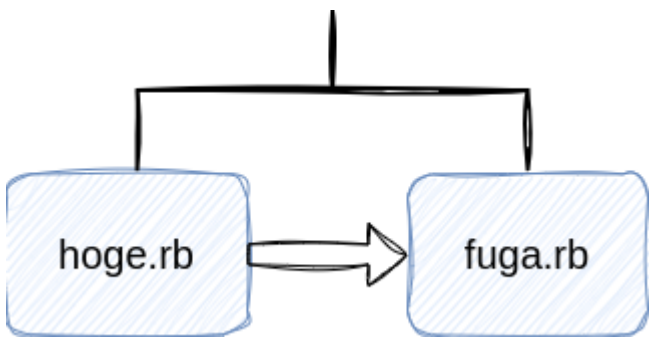
まあでもスクリプトで `addEventListener()` を使うのが良さそう

```
<input id="b" type="button">
<script type="text/ruby">
  require 'js'
  document = JS.global[:document]
  document.getElementById('b').addEventListener('click') do |ev|
    JS.global.alert('hoge')
  end
</script>
```

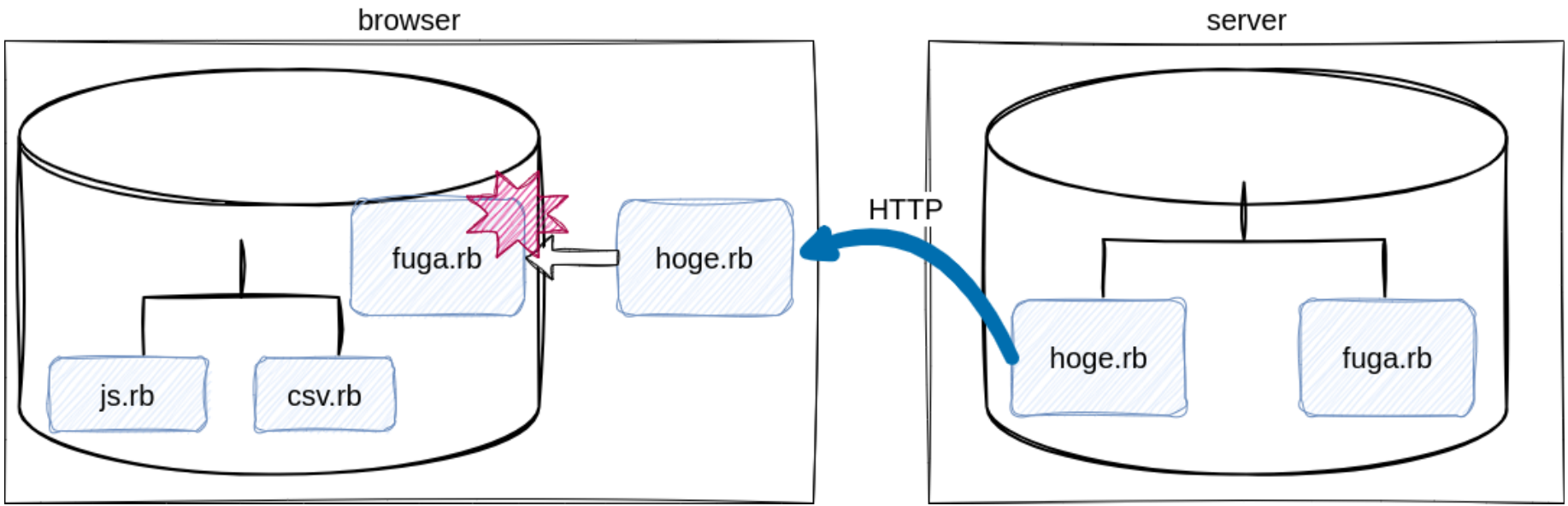
require

require はファイルシステムから .rb ファイルを読む

```ruby
# hoge.rb
require_relative 'fuga'
```



- `<script src=〜>`で読み込んだ.rbからはサーバー上のファイルはrequireはできない
- ファイルシステムが異なる



`JS::RequireRemote.instance.load(path)` で相対パスで指定した .rb を読み込める

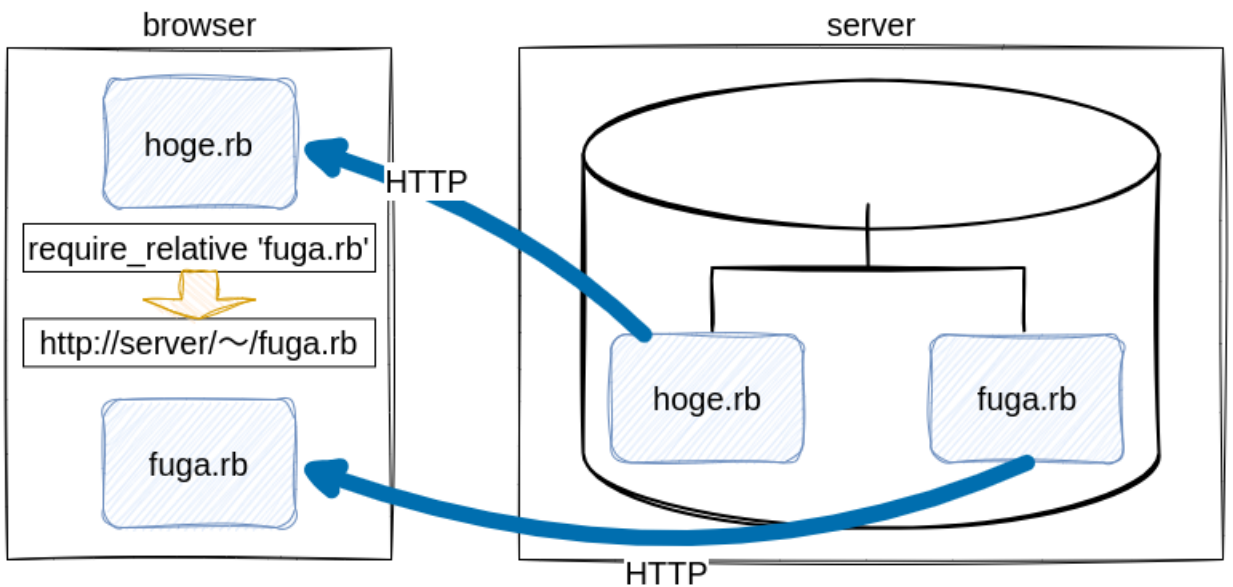
`require_relative` が↑を使うようにパッチ

```ruby
require 'js/require_remote'
module Kernel
  prepend Module.new {
    def require_relative(path)
      caller_path = caller_locations(1,1).first.absolute_path || ''
      dir = File.dirname(caller_path)
      file = File.absolute_path(path, dir)
      super file
    rescue LoadError
      JS::RequireRemote.instance.load(path)
    end
  }
end
```

https://ledsun.hatenablog.com/entry/2023/11/14/183047 を改変

require_relative が動く！

```
# hoge.rb
require_relative 'fuga'
```



JSrbライブラリ

https://github.com/tmtm/jsrb

- JSをもう少しRubyっぽく書けるようにする
- 戻り値を`JS::Object`ではなくRubyのオブジェクトに変換したり
- Enumerable 化して each が使えたり
- null や undefined を nil にしたり

```
# JS
elements = JS.global[:document].querySelectorAll('div')
elements[:length].to_i.times do |i|
  elements[i][:style][:color] = 'red'
end
```

```
# JSrb
elements = JSrb.document.query_selector_all('div')
elements.each do |element|
  element.style.color = 'red'
end
```

ruby.wasmを使った簡単な例

デモ

```
<style>.hoge-color{padding:10px}</style>
<div class="hoge-color"><input class="name" value="red"><input class="code"></div>
<div class="hoge-color"><input class="name" value="blue"><input class="code"></div>
<div class="hoge-color"><input class="name" value="yellow"><input class="code"></div>
```

```
def color_code(color)
  JSrb.global.get_computed_style(color).background_color.scan(/\d+/).map{format('%02x',_1.to_i)}.join
end
colors = JSrb.document.query_selector_all('.hoge-color')
colors.each do |color|
  name = color.query_selector('input.name')
  color.style.background_color = name.value
  code = color.query_selector('input.code')
  code.value = color_code(color)
  name.add_event_listener('change') do
    color.style.background_color = name.value
    code.value = color_code(color)
  end
end
```

- HTMLに書かれてる要素をquerySelectorで取得したり
- createElementで要素を新しく作ったり
- addEventListenerでイベント追加したり
- styleでスタイルを変更したり

まあJavaScriptとだいたい同じ

ところでカスタム要素って知ってます？

私は最近知りました（フロントエンド疎いので）

カスタム要素(Custom Elements)

```
<style>hoge-color{display:block;padding:10px}</style>
<hoge-color color="red"></hoge-color>
<hoge-color color="blue"></hoge-color>
<hoge-color color="yellow"></hoge-color>
```

```
// JavaScript注意
class HogeColor extends HTMLElement {
  static observedAttributes = ['color']
  connectedCallback() {
    var name = this.appendChild(document.createElement('input'))
    name.value = this.getAttribute('color')
    name.addEventListener('change', ()=>{this.setAttribute('color', name.value)})
    this.style.backgroundColor = name.value
    this.code = this.appendChild(document.createElement('input'))
    this.code.value = colorCode(this)
  }
  attributeChangedCallback(name, oldValue, newValue) {
    this.style.backgroundColor = newValue
    if (this.code) this.code.value = colorCode(this)
  }
}
customElements.define('hoge-color', HogeColor)
```

カスタム要素

- 独自のタグを作ることができる
- 要素の生成時や属性が変わったときの振る舞いをJavaScriptで書ける
- スタイルシートも指定できる
- 動的に生成された要素でも動くので便利そう

Rubyでも書きたい！！

CustomElement

https://github.com/tmtm/custom_element

やってみたらできた 

```
class HogeColor < CustomElement
  self.observed_attributes = ['color']
  def connected_callback
    name = append_child(JSrb.document.create_element('input'))
    name.value = get_attribute('color')
    name.add_event_listener('change'){set_attribute('color', name.value)}
    style.background_color = name.value
    @code = append_child(JSrb.document.create_element('input'))
    @code.value = color_code(self)
  end
  def attribute_changed_callback(name, old, new)
    style.background_color = new
    @code.value = color_code(self) if @code
  end
end
CustomElement.define('hoge-color', HogeColor)
```

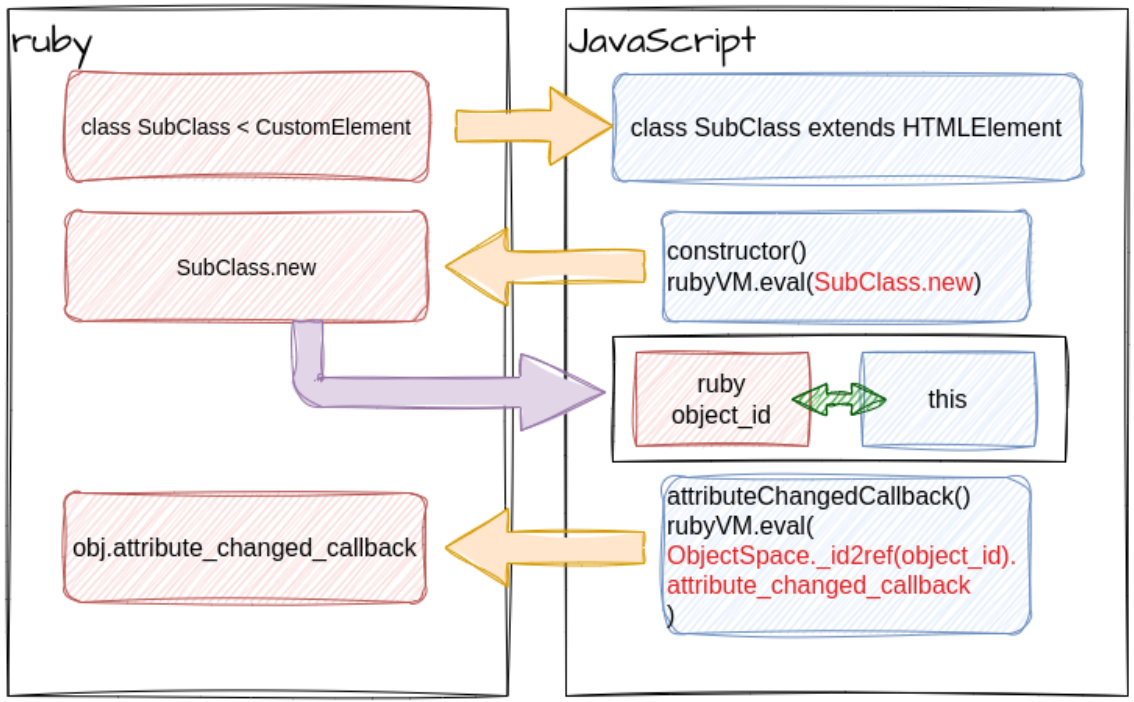
デモ(動きは同じなのでやらない)

だいたい同じ

```
// JavaScript
class HogeColor extends HTMLElement {
  static observedAttributes = ['color']
  connectedCallback() {
    var name = this.appendChild(document.createElement('input'))
    name.value = this.getAttribute('color')
    name.addEventListener('change', ()=>{this.setAttribute('color', name.value)})
    this.style.backgroundColor = name.value
    this.code = this.appendChild(document.createElement('input'))
    this.code.value = colorCode(this)
  }
  attributeChangedCallback(name, oldValue, newValue) {
    this.style.backgroundColor = newValue
    if (this.code) this.code.value = colorCode(this)
  }
}
customElements.define("hoge-color", HogeColor)
```

```
# Ruby
class HogeColor < CustomElement
  self.observed_attributes = ['color']
  def connected_callback
    name = append_child(JSrb.document.create_element('input'))
    name.value = get_attribute('color')
    name.add_event_listener('change'){set_attribute('color', name.value)}
    style.background_color = name.value
    @code = append_child(JSrb.document.create_element('input'))
    @code.value = color_code(self)
  end
  def attribute_changed_callback(name, old, new)
    style.background_color = new
    @code.value = color_code(self) if @code
  end
end
CustomElement.define('hoge-color', HogeColor)
```

仕組み


ruby
class SubClass < CustomElement
SubClass.new
obj.attribute_changed_callback
JavaScript
class SubClass extends HTMLElement
constructor()
rubyVM.eval(SubClass.new)
ruby
object_id
this
attributeChangedCallback()
rubyVM.eval(
ObjectSpace._id2ref(object_id).
attribute_changed_callback
)

まとめ

趣味のRubyプログラミング楽しい

誰の役に立たなくても車輪の再発明でも楽しい

ブラウザでも Ruby プログラムが動く！

JavaScript でできないようなこともできる

JavaScript より遅いけど気にしたら負け

Ruby でフロントエンドが書けるの最高！

ネコチャン絵文字 ©しかまつ https://note.com/shikamatsu/n/nd217dc0617db