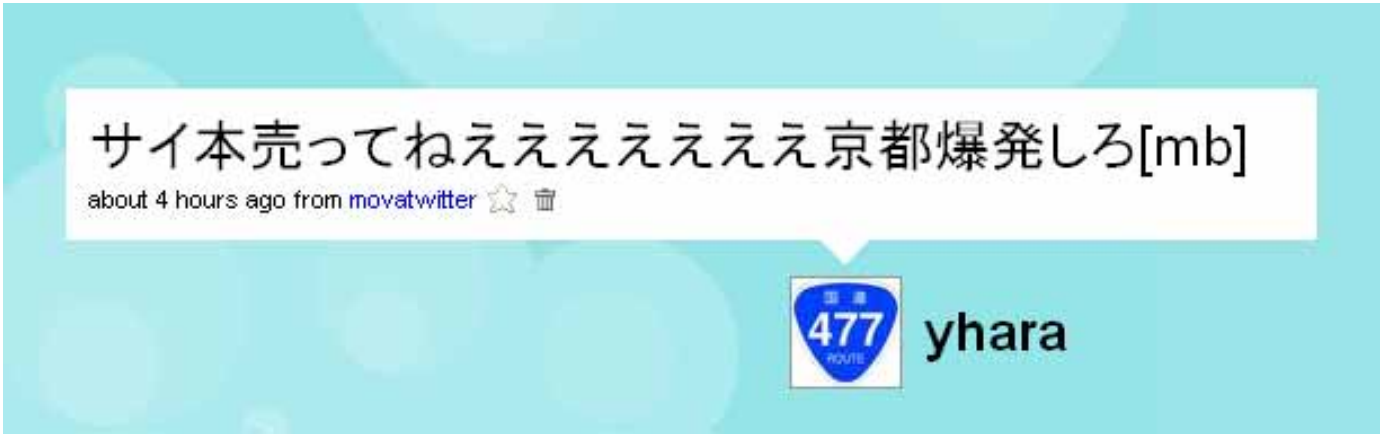
# Javascriptで無限ループを実現する5つの方法

yhara@京大マイコンクラブ

# 自己紹介

- 琵琶湖の方から来ました
- Javascript暦は2年ちょっと
- 普段はRuby、ほかいろいろ(言語マニア)
- 学生をしています

# 京都にはサイ本売ってない説




yhara has joined

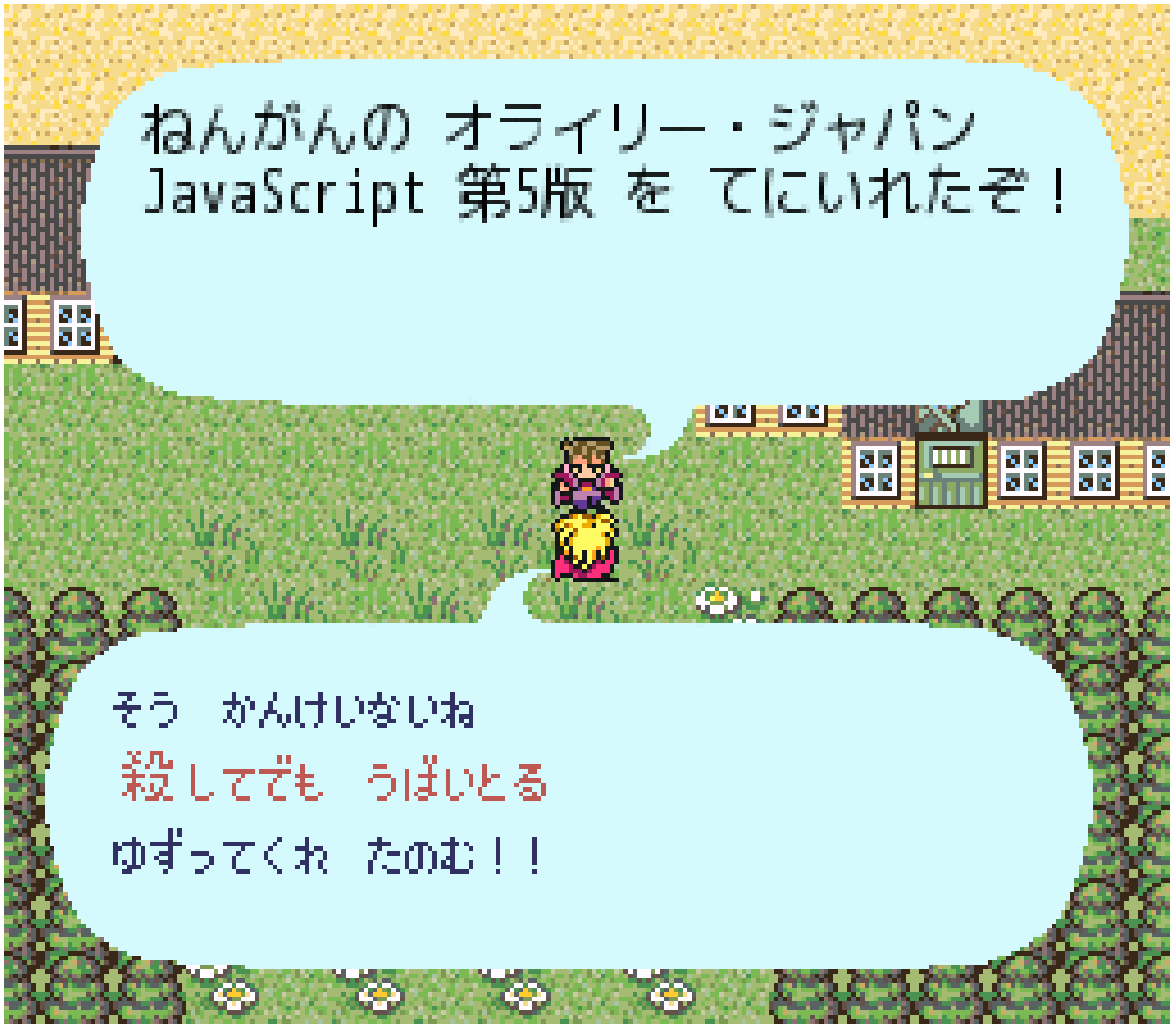
yhara 11:20pm (January 13)

こんばんわ、サイ本の入手に失敗したyharaです

nanki

京都のサイ本は僕が買い占めた。

# ヨドバシ＋＋


ねんがんの オライリー・ジャパン
JavaScript 第5版 を てにいれたぞ！
そう かんけいないね
殺してでも うばいとる
ゆずってくれ たのむ！！

# (1) 普通に無限ループを書く

```
for(;;){
}
```

```
while(true){
}
```

- ブラウザが固まります (＞＜)

# 予想される反論

- 無限ループなんてつかわねーよwww

- 無限でなくても、時間のかかる処理をループさせるとやっぱりブラウザが固まってしまう

```
while(条件){
  // 時間のかかる処理
}
```

# (2) setTimeoutを使う

- setTimeout(**関数**, **ミリ秒**)
- **指定した時間後に指定した関数を実行してくれる**

```
var f = function(){
  //時間のかかる処理
  if(条件)
    setTimeout(f, 0);
}
```

# でも…、

- setTimeoutはめんどい
- 同期処理がしたいのに、非同期にしないといけない

**理想：(同期)**

```
while(条件){
  //時間のかかる処理
}
```

**現実：(非同期)**

```
var f = function(){
  //時間のかかる処理
  if(条件)
    setTimeout(f, 0);
}
```

# 同様に

- Ajaxもめんどい
- 同期処理がしたい時も、非同期にしないといけない

**理想：(同期)**

```
data = Ajax.get(path);
alert(data);
```

**現実：(非同期)**

```
Ajax.get(path, function(data){
  alert(data);
}
```

- じゃあどうしたらいい・・・？

# (3) jsThread

- **別名**Thread.Concurrent
  - http://jsthread.sourceforge.net/cgi-bin/wiki/wiki.cgi
- JavaScript**のみ**で**スレッドシステムを実装！**

```
// ↓ブラウザなぜか固まらない
Concurrent.Thread.create(function(){
  while(条件){
    //時間のかかる処理
  }
});
```

# 仕組み

- Javascriptで書かれたJavascriptパーザ内蔵
- 指定した関数のソースコードを解析し、自動的にsetTimeoutを挟んでくれる
  - toSource()でソースが取れるjavascriptならでは！
- 15,119行のJavascript (1/8現在)
- 無茶しやがってwww
- その他、サイトに解説あり
  - http://jsthread.sourceforge.net/cgi-bin/wiki/wiki.cgi

# (4) JSDeffered

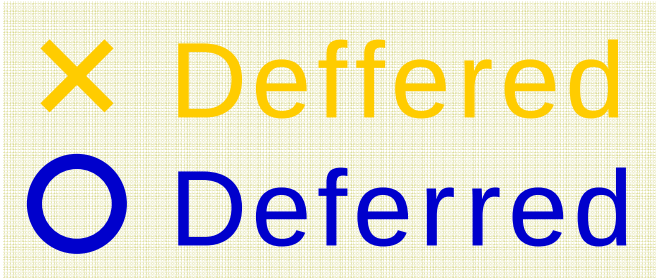
- **関数合成を駆使して、同期処理をそれっぽく書ける**
  - http://coderepos.org/share/wiki/JSDeferred

```
// loopの例：

loop(ary, function (n) {
  //時間のかかる処理
  return wait(0.1);
});
```

```
// nextの例：

loop(5, function (i, o) {
  //ループ処理
  return o.last? i : wait(1);
}).
next(function (e) {
  //ループが終わったらする処理
  print("end ["+e+"]");
})
```

# 注意点

- つづりが　むずかしいです(^o^)




- Defer (延期する) + r + ed とおぼえよう！

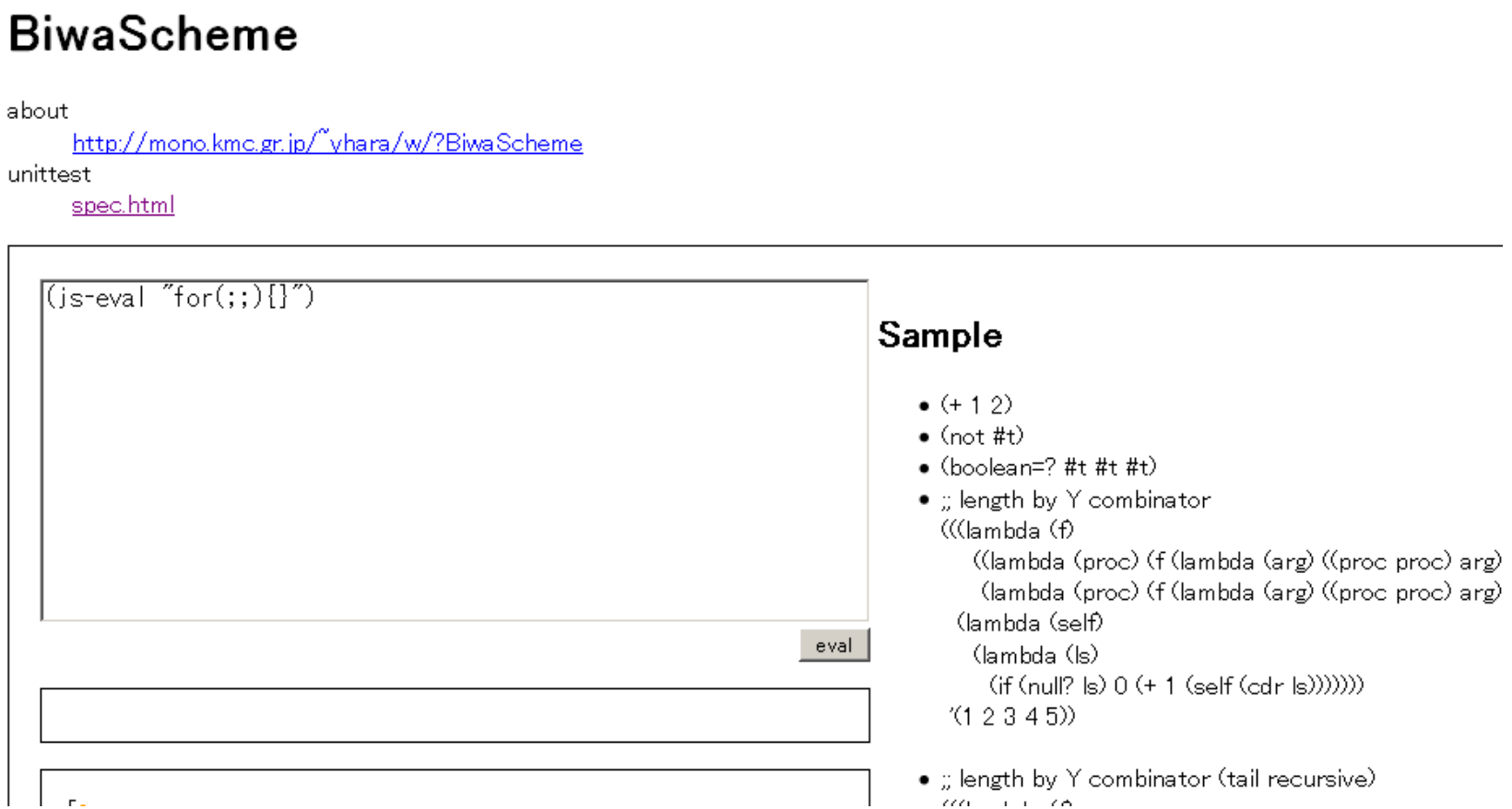
# JSDeffered＝DSL

- JavaScriptを拡張した新しい言語みたいな感じ
  - parallel()：**並列実行**
  - wait()：**待つ、中断**
  - next()：**順次実行**
  - loop()：**繰り返し**
  - error()：**エラー時**

- **言語内言語？**

# (5) Javascriptで別の言語を実装

- **例えば**Scheme
  - http://mono.kmc.gr.jp/~yhara/w/?BiwaScheme



# BiwaScheme

- Javascript**で書かれた**Scheme**処理系**
  - http://mono.kmc.gr.jp/~yhara/w/?BiwaScheme
- R6RS**準拠を目標に開発中です**(>_<)

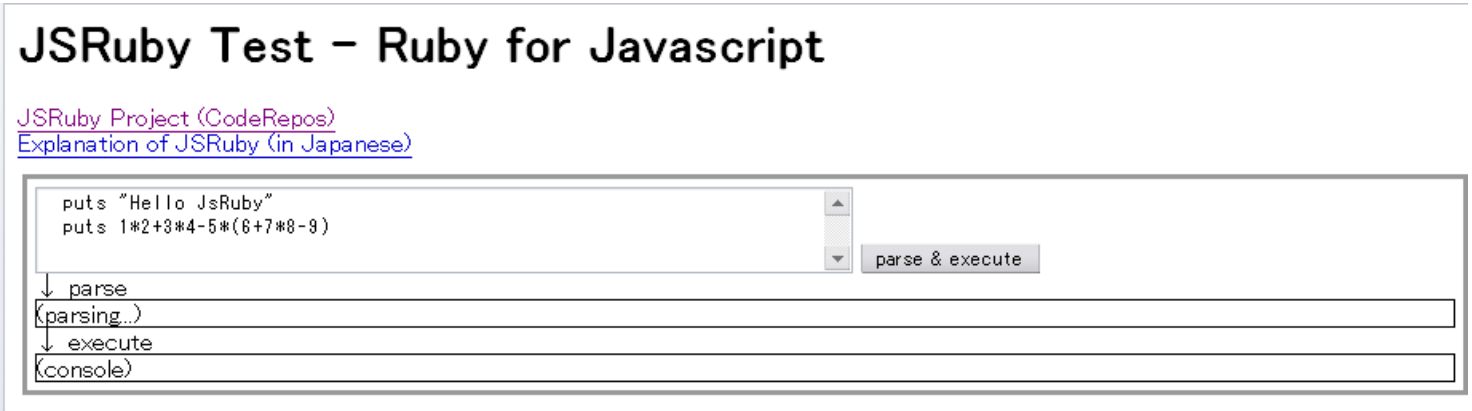
```
(lambda (f)
  ;; 時間のかかる処理
  (sleep 0)
  (if 条件 (f) 返り値))
```

```
(let1 s (http-request (path))
  (print s))
```

  - (sleep 0)**のところでインタプリタの状態を保存**
  - setTimeout**で指定した時間後にインタプリタを再開**

# JSRuby

- Javascriptで書かれたRuby処理系(！)
  - http://coderepos.org/share/wiki/JSRuby
- FizzBuzzを動かす程度の能力




- sleepはまだなさそうだけど、原理的にはできるはず

- cf. HotRuby (YARVの中間コードをjsで実行)
  - http://d.hatena.ne.jp/yukoba/20071231/p1

# まとめ

- さよならsetTimeout

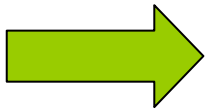
- とにかくわかりやすく書きたい→jsThread
- とにかく軽い方がいい→JSDeffered
- (どちらでもない)→BiwaScheme

# おまけ (Reposh)

- Reposh – Repository Shell
  - http://mono.kmc.gr.jp/~yhara/w/?Reposh

**Reposh**なし:

```
$ svn st
$ svn di foo.rb
$ svn ci –m “foo”
```



**Reposh**あり:

```
> [Enter]
> di foo.rb
> ci –m “foo”
```

- svn, svk, hg, darcsに対応(ディレクトリ構成から自動判定)