実践isomorphic (+ Electron)

mizchi / Increments, Inc

Ismorphic Meetup

ismorphicとは

環境を選ばないJavaScript

今日話さないこと

☞ サーバーサイドレンダリング

(自分の)isormorphicの目的

① どこでも動く単機能なモジュールを提供したい

② node.jsのモジュールシステムを活用して開発したい

③ MVC矯正ギプス

1. どこでも動いてほしい

☞ ブラウザ環境

☞ DOMから独立したWorker環境

☞ node環境

☞ ヘッドレステスティング

2. node.jsのモジュールシステムを活用して開発したい

☞ 「nodeモジュールもnode非依存なら動くはず」という考え方

☞ ライブラリ提供時は環境依存かどうかを常に意識する

3. MVC矯正ギプス

☞ pure javascript + 環境の差を吸収できる+α に制限される

☞ MVCでいうビュー層が自然と触り辛くなる

☞ => ドメイン層に注力

一昔前のJSといえば...

jQueryとDOMとJavaScriptの区別がされていないサンプルコードがたくさんあって辛かった

たとえば...

markdownコンパイラのmarkedはどの環境でも動く

☞ nodeでコンパイルできる

☞ ブラウザコンパイルできる

☞ webworkerで1/4ずつ千切って並列ビルドできる

☞ ブラウザを立ち上げずにnodeで単体テストできる

その他、強い動機

とにかく不安定で重いPhantom.jsでテストしたくない

各環境の差

グローバル空間

`window`: DOMが存在する環境

`global`: node環境

`self`: WebWorker / ServiceWorker環境

環境依存API

- ☞ document / navigator
- ☞ setImmediate / requestAnimationFrame
- ☞ ポリフィルがないネイティブモジュールへのrequire(httpやvm)
- ☞ etc...

実際には...

☞ ECMAScriptの範囲内だけで実装するのが(理想的には)正しい

☞ とはいえnodeとブラウザをサポートしてたら十分

☞ worker環境は忘れられがち

ネイティブモジュール

☞ preinstallでgypでネイティブビルドするものは呼べない

☞ ネイティブで選べない

☞ sundown より marked

どうしてもネイティブモジュールを呼びたい

- ☞ emscriptenでビルドする
  - ☞ zlib.jsとか
  - ☞ llvm.jsとか
- ☞ ビルドが高コスト/バイナリサイズが巨大化するので推奨しない

commonjs require

commonjs/require とは

☞ node.jsのモジュール解決システム

☞ require関数とそのファイルを含むpackage.jsonのmainから解決される

最小のモジュール

- package.json
- foo.js

```
module.exports = function(){console.log('foo')};
```

```
{"name":"foo", "main": "foo.js"}
```

commonjsのブラウザ向けプリプロセッサ

- ☞ substack/node-browserify
- ☞ webpack
- ☞ require1k — CommonJS require for the browser in 1k
- ☞ Duo

それぞれ微妙に挙動が異なるが割愛

基本的な仕組み

☞ require関数のポリフィルを挿入する

☞ ASTからrequire('./hoge')を相対パス./hogeへの参照置き換える

☞ ./hoge への参照は同じモジュール内で共有される

browserify/webpackの制限

☞ require先が文字列以外だと参照の追跡を行えない

☞ require関数の参照をコールしたときも追跡しない

ダメな例

```
var x = './foo';
require(x);
```

ダメな例

```
req = require;
req('./foo');
```

ダメな例

```
global.require('foo');
```

※これはあとで使う

実践

altjsやjsxの変形

いろんな方法がある

☞ browserify transform

☞ webpack plugin

☞ gulp/gruntでプリプロセス

☞ require.extensionsでフック(node/electronでのみ有効)

mizchiのたどり着いたベストプラクティス

☞ gulpでsrc以下をlibに吐き出す

☞ libを .gitignore で指定してgitから無視

☞ watchifyでlib以下を監視して差分build

☞ テストはmochaとcoffeeで雑に書く(好みで選ぶ)

☞ power-assertで変形

☞ テスト時はlib側を呼ぶ

```
mocha --require espower-coffee/guess test/*.coffee
```

```
src/
  foo.coffee
  bar.ts
  template.jade
lib/   <-- .gitignore
  foo.js
  bar.js
  template.js
test/
  foo-test.coffee
```

この方式の理由

☞ browseirfyで一括で解決すると、単体テスト時に一箇所書き換えただけで全部のモジュールをビルドする必要がある

☞ 単体テストできることでモジュールの独立性を保証できる

watchify

☞ browserify はエントリポイントから全てゼロからビルドしようとする

☞ watchifyは差分監視して吐き出す

自分のプロジェクトでは 8.3s → 0.02s

Reactいるだけで顕著に変わる

ES6 moduleどうする？

- ☞ import / export
- ☞ 将来的にこっち?(nodeがnode_modulesをどう扱うか決まってない)
- ☞ babelはrequire形式に変換する
- ☞ typescriptの `--target commonjs` はexport default 未対応

Electron (旧 atom-shell)

Electron

☞ Atomのベース

☞ window と global が共存する環境

☞ トップレベルthis は window

Electron環境の特殊なライブラリ

`require('app')`, `require('browser-window')` 等

他、menu, clipboard, crash-reporter,

Electron環境でのbrowserify

☞ node_modules下の不要なファイルを削除してサイズ縮小

☞ 簡易な難読化

自分のプロジェクトでは(250MB → 2.2MB)
使わなくても動くのが強みではある

一部browserifyできないのでどうにかしてかわす

```
var marked = require('marked'); // browserifyで変形
var app = global.require('app'); // そのまま通す
```

※ Oops

Node.js / Electron目線でのブラウザ環境

結局DOMとはなんだったのか

- ☞ もっとも普及した簡易なGUIツールキット
- ☞ nodeで実装したモジュールをさっくりGUIアプリに載せられてサイコー
- ☞ ただしEmbeddedなChromeは含むのでバイナリは大きい(40MB程度)

Electronの果たす役割

- ☞ 変化の早いブラウザ環境を固定するアプローチの一つ
- ☞ Chromeの最新APIを惜しみなく使える現実的なプラットフォーム

他、環境ごとのIsomorphic

View Isormorphic

Reactの場合

☞ `React.renderToString(...)`

☞ `React.renderToStaticMarkup(...)`

jsdom使えば `React.renderToString(...)` までいける

参考: JSDOMとReact.addons.TestUtilsでReactをヘッドレスにテストする - Qiita

Isomorphic for V8

```
ctx = V8::Context.new
ctx.eval """
  var global = {};
"""
ctx.eval $react_source
ctx.eval """
  var React = global.React;
"""
```

V8 bindingあれば他の言語でもいける

Network Isomorphic

ServiceWorkerでIsomorphic

- ☞ ネットワークプロキシ
- ☞ アプリケーションキャッシュ
- ☞ プッシュ通知(これは今回どうでもいい)

Express使いたくない?

☞ ネットワークリクエストがモックできればIn/Out制御できるのでは

☞ => ServiceWorker上でexpress実装したらいいんじゃね？

Sabizan

☞ mizchi/sabizan

☞ ServiceWorker上でexpress風のAPIが書ける

☞ まだまだPoC

mizchi-sandbox/scala-js-in-service-worker

今動いたコード

```
# it will respond to https://localhost:3000/api/user/fuga?foo=bar
proxy.get '/user/:id', ({id}, {foo}, req) ->
  {id, foo}

# Return with promise
proxy.post '/post', ({}, body) ->
  new Promise (done) ->
    setTimeout ->
      done {type: 'this is post:'+params.prop}
    , 300
```

まとめ

(自分の)Isomorphic世界観

☞ View(React)

☞ ネットワーク(ServiceWorker)

☞ モジュール(Browserify)

全部ヘッドレス

おわり