Getting Started RequireJS

社内勉強会 2012/11/16

@ahomu

http://aho.mu

1. Problems
2. Modular JavaScript
3. Case Study
4. Hands-On

✓ JavaScript 書いてるひと向け

✓ 手を動かすキッカケをつくろう

✓ 実例まじえて、内容やさしめ

Problems

中規模以上のJavaScriptを管理する上での問題

Bad Context

Libs of choice

jQuery

Backbone

Plugins

Your Code

source. Dependency Management with RequireJS

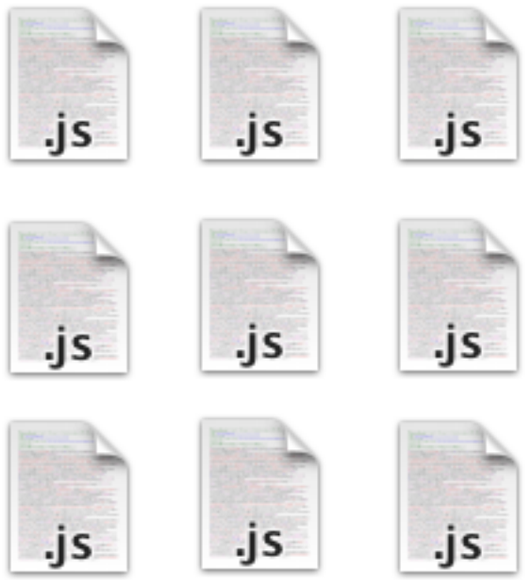
Good Context

Libs of choice

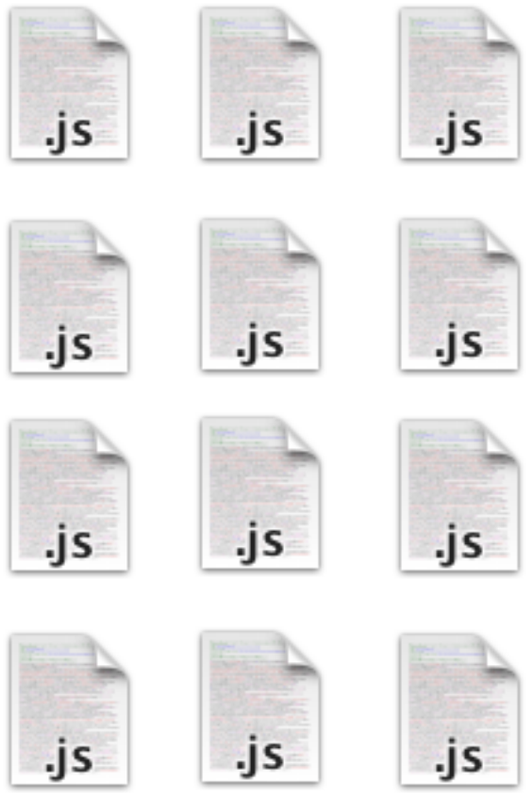
jQuery

Backbone

Plugins



Your Code



source. Dependency Management with RequireJS

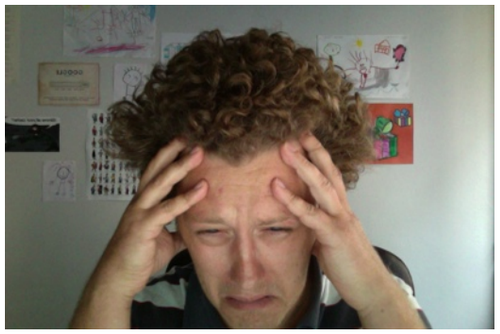
Dependencies

Libs of choice

Plugins

Your Code

source. Dependency Management with RequireJS



Common issues

- われわれの扱うコードは膨大である
- 個々のコードは粒度をそろえたい
- 再利用できるようにしたい
- 依存関係を整理したい
- なんとかしたい

Modular JavaScript

モジュラーなJavaScriptであることのおさらい

Examples

e.g. THE BIG ONE

```
(function() {
  var Application = {
    foo: function() {...},
    bar: function() {...}
  };

  // long historical stacking code...

  new Application();

})();
```

10000行超えたあたりで狂気

際限なく重くなるエディタ

e.g. Namespace (1/3)

```
// main.js
window.NS = {};

// page.js
window.NS.Page = {
    initialize: function() {},
    getContent: function() {},
    render    : function() {}
};

// util.js
window.NS.Util = {};
```

e.g. Namespace (2/3)

```
// first
<script src=”js/main.js”></script>
<script src=”js/util.js”></script>
<script src=”js/page.js”></script>
```

よく分割されてるように見える

e.g. Namespace (3/3)

```
// finally
<script src="js/main.js"></script>
<script src="js/util.js"></script>
<script src="js/module_1.js"></script>
<script src="js/module_6.js"></script>
<script src="js/module_5.js"></script>
<script src="js/module_2.js"></script>
<script src="js/module_3.js"></script>
<script src="js/module_4.js"></script>
<script src="js/module_7.js"></script>
<script src="js/page.js"></script>
```

読み込み順という名の依存性地獄
ていうか手書きムリ

RequireJS

殺伐とした依存解決地獄に参上!

ヽ|・∀・|ノ

`define`

```
/**
 * @method define
 *
 * @param {String}   [id]
 * @param {Array}    dependencies
 * @param {Function} factory
 */
define(id, dependencies, factory);
```

JavaScript (page.js)

```
// page.js (depend moduleA & Util)
define(['moduleA', 'util'], function(moduleA, Util) {
  function Page($elm) {
    this.$elm = $elm
    Util.greatFunction();
  }

  Page.prototype.render = function() {
    return this.$elm.html(moduleA.getContent());
  }

  return Page;
});
```

JavaScript (main.js)

```
// main.js
define(['jquery', 'page'], function($, Page) {

  // initialize
  new Page($('#container'));

  // some business
  Page.render();

});
```

HTML

```
<!-- Simple! -->
<script data-main="main" src="require.js">
</script>
```

問題は解決された！

```
<!-- Same code -->
<script src="require.js"></script>
<script>
require(['main'], function(Main) { });
</script>
```

A JAVASCRIPT MODULE LOADER

WHY AMD?

This page talks about the design forces and use of the Asynchronous Module Definition (AMD) API for JavaScript modules, the module API supported by RequireJS. There is a different page that talks about general approach to modules on the web.

MODULE PURPOSES

§ 1

What are JavaScript modules? What is their purpose?

- ***Definition***: *how to encapsulate a piece of code into a useful unit, and how to register its*

AMD

- Asynchronous Module Definition
- 依存定義の解決 with 非同期ローダー
- HTMLはシンプルに定義して済ます
- PC環境向けなら、AMDのままでも良さそう

Given benefits

- RequireJS は非常に賢い
- 依存解決がシンプルになる
- define を利用したパターンが強制される
- コーディングパターンの統制が取りやすい
- モジュラーであることを意識しやすくなる

Case Study

開発中プロジェクト的な事例紹介

Using libraries

- RequireJS
- Backbone.js
- lodash
- Zepto
- Jasmine + Sinon + Coffee + Testem

Tour

- Backbone 地獄（90ファイルくらい）
- Jasmine のあたり
- Grunt でかためてるところ
  - Devel
  - Build

THE BIG ONE AGAIN

- Productionでは1ファイルにしたいよね
- HTTPリクエストが少ないのは良いことだ
- `grunt-requirejs`で、ひとまとめに出力
- ついでに UglifyJS で optimize

Difference from concat

- 小〜中規模であればconcatで良いかな、と
- シンプルだし
- が、concatは依存性を示せない
- が、concatは再利用性をサポートしない
- なぜなら、何のパターンも提示しないから

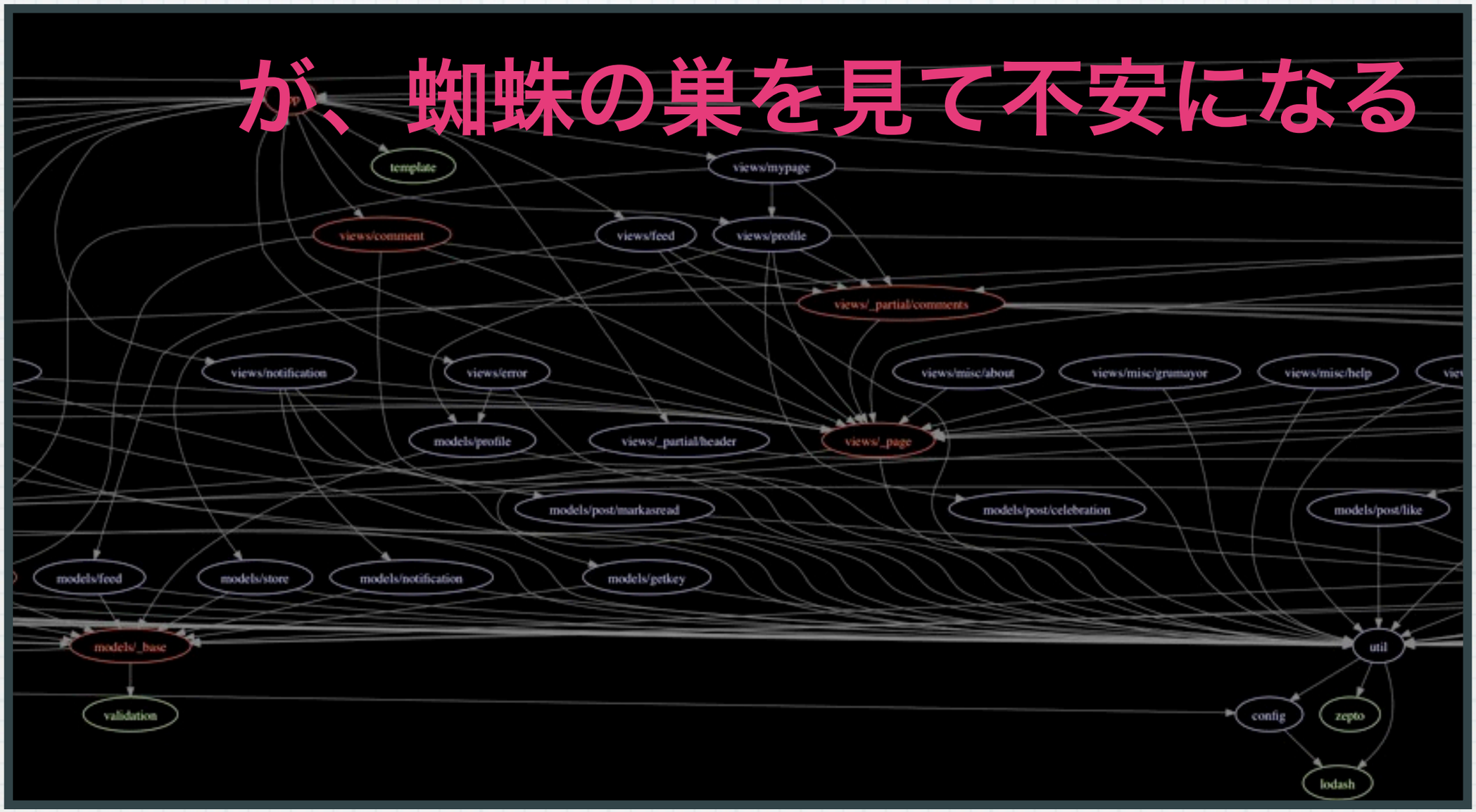
Seems like good!

```
define([

  'util',

  // main
  'views/_page',
  'models/store',

  // comment action
  'views/_action/post-comment',
  'views/_action/receipt',
  'models/post/comment',

  // share action
  'views/_action/post-share',
  'models/post/share-ranking',

  // comments
  'views/_partial/comments',

  // post model (welcome / subscription)
  'models/post/welcome',
  'models/post/subscription'

], function(Util, PageView, Store,
            CommentPostView, ReceiptView, PostComment,
            SharePostView, PostShareRanking,
            CommentsPartialView,
            PostWelcome, PostSubscription
  ) {
  /**
   * @class StoreView
   * @extends BaseView
   */
  return PageView.extend({
```

依存を意識する良い習慣

見通しも良くなる

Deep dependencies...



Call stack Hell...

Umm...

- まあ、気にしないことにした
- Zepto や Underscore (lodash) のように global に export する奴らはサボった
- テスト周りはもうちょい、どうにかしたい

Hands-On

ブラウザAMD ＆ r.js Optimize

Agenda

- ブラウザ上でいわゆるAMD
- requirejs.config する
- r.js で optimize
- almond に差し替えて
- grunt-requirejs してみる（いけるか？）

Documents

- http://requirejs.org/docs/start.html
- http://requirejs.org/docs/errors.html
- https://github.com/jrburke/r.js/blob/master/build/example.build.js
- https://github.com/jrburke/almond

Try AMD Steps...

- js/main.js の中で、define してみる
- data-main で読んでみる
- js/sub.js も define してみる
- js/main.js を js/sub.js に依存させる

Configuration Steps...

- requirejs.config() で baseUrl する
- js/lib/*.js を paths する
- shim する

Install r.js

```
# グローバルにインストール
% npm install -g requirejs
```

```
# パスが通っていれば、r.jsが利用可能になる
% r.js
See https://github.com/jrburke/r.js for usage
```

Using r.js

```
% r.js -o out=js/all.js\
         baseUrl=js\
         name=bootstrap\
         mainConfigFile=js/bootstrap.js

# mainConfigFileに含まれるbaseUrlは、
# 指定ファイルからの相対になってしまうので、
# オプションで指定するほうがよい
```

Almond

```
% r.js -o out=js/all.js\
         baseUrl=js\
         name=lib/almond\
         include=bootstrap\
         mainConfigFile=js/bootstrap.js

# 非同期ロード処理など、不要な機能を除いたshim
# 1ファイルにまとめること
# 依存モジュール名を動的に指定しないこと
```

grunt-requirejs

```
% npm install -g grunt

% npm link

% grunt requirejs
```

Appendix1: with Backbone

```
define(['backbone'], function(Backbone) {
  return Backbone.Model.extend({
    url: 'path/to',
    defaults: {
      foo: 'bar'
    },
    validation: {
      // ...
    }
  });
  // 蛇足 newしたのをreturnするとsingletonぽい
});
```

Appendix2: with Jasmine

```
// SpecRunner.html
require(['spec/util', 'spec/main'], function() {
  var env = jasmine.getEnv();
  env.addReporter(new jasmine.HtmlReporter);
  env.execute();
});
```

```
// spec.coffee
define ['util'], (Util)->
  describe 'Commmon Utilities', ->
    util = null
    beforeEach ->
      util = new Util

    it 'Should `plzTruthy` returns true', ->
      expect(Util.plzTruthy()).toBe true
```

Appendix3: Existing Module

```
// たとえば Zepto ($)
// `define`が定義されていて、AMDぽかったら...
if (typeof define === 'function' && define.amd) {
  define([], function() {
    return Zepto;
  });
}
```

Appendix4: CommonJS Sugar

```
define(function(require) {
  var hoge = require('hoge'),
      fuga = require('fuga'),
      piyo = require('piyo');

  return {
    hogehoge: hoge.toString(),
    fugafuga: fuga.toString(),
    piyopiyo: piyo.toString()
  }
});
```

Appendix5: madge

```
# https://github.com/pahen/node-madge
% npm install -g madge
% brew install graphviz

# モジュールの依存関係を可視化
% madge --image out.png --format amd ./app
```

Conclusion

- RequireJSはライブラリでありつつ
  依存解決するためのモダンパターンそのもの
- べた書きの JS から、疎結合な JS に
- 集団開発や長期リファクタリングを見越して
  導入すると、さらに価値があるはず
- ぜひ現場で展開していただけると幸い

おわり

ありがとうございました

1. ParisJS #10 : RequireJS (slide)
2. Modular JavaScript with AMD and RequireJS
3. RequireJS + PhoneGap
4. Getting Started Managing Client Side Scripts with Require.js « I Am Not Myself
5. Namespaces are Old School ¦ Blog ¦ Miller Medeiros
6. AMD is Not the Answer : Tom Dale
7. JavaScript module loaders: necessary evil? ¦ Ruben Verborgh
8. japh(r) by Chris Strom: Pretty Backbone.js Views with Require.js