Backbone.js X RequireJS

Quick Guide

RequireJS

RequireJS 是 AMD 規範的實作版本

require(options);

```
require({
  baseUrl: './',
  paths: {
    alias_name: full_path,
  }
});
```

require API 可以接受一個物件做為設定檔
其中 baseUrl 指向 HTML 頁面所在路徑

require(dependencies, callback);

```
require([
  'js_file_path1',
  'js_file_path2',
  'plugin_path!js_file_path3',
  'plugin_path!js_file_path4',
], function (obj1, obj2, obj3, NS4) {
  obj1.doSomething();
  obj2.doSomething();
  obj3.doSomething();
  obj4 = new NS4.Model();
});
```

載入的相依模組與 callback 的參數為一對一關係

define(id?, dependencies?, factory);

```
define([
  'js_file_path1',
  'js_file_path2',
  'plugin_path!js_file_path3',
], function(obj1, obj2, constructor3) {
  return {
    method1: function () {
      obj1.doSomething();
      obj2.doSomething();
      obj3 = new constructor3();
    }
  };
});
```

define 與 require 原理類似
但主要是用來定義模組

```
<project>
├── css
│   └── screen.css
├── index.html
├── js
│   ├── model
│   ├── view
│   ├── router
│   ├── app.js
│   ├── build.js
│   └── main.js
└── lib
    ├── backbone
    │   └── backbone-min.js
    ├── jquery
    │   └── jquery-min.js
    ├── requirejs
    │   ├── order.js
    │   ├── text.js
    │   └── require.js
    └── underscore
        └── underscore-min.js
```

目錄結構是可以自訂的

```
<project>
└── js
    ├── model
    │   └── Config.js
    ├── router
    │   └── Router.js
    └── view
        ├── Status.js
        └── Switch.js
```

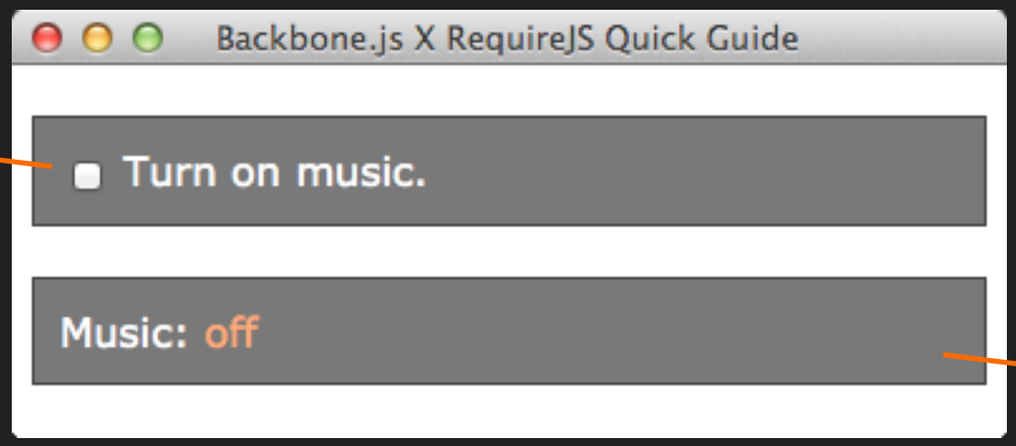
必要時可以再多一個 collection 資料夾

Example

Config
(Model)

music: false

Switch
(View)



Status
(View)

之前在讀書會介紹過的範例

```
<script src="lib/jquery/jquery-min.js"></script>
<script src="lib/underscore/underscore-min.js"></script>
<script src="lib/backbone/backbone-min.js"></script>
<script src="js/app.js"></script>
</body>
```

```
<script data-main="js/main" src="lib/requirejs/require.js">
</script>
</head>
```

進入點為 js/main.js

js/main.js

```
require({
  baseUrl: './',
  paths: {
    order: 'lib/requirejs/order',
    text: 'lib/requirejs/text'
  }
});

require([
  'order!lib/jquery/jquery-min',
  'order!lib/underscore/underscore-min',
  'order!lib/backbone/backbone-min',
  'order!js/app',
], function () {
  App = _.last(arguments);
  App.initialize();
});
```

underscore 和 Backbone.js 的官方版本不支援 AMD

js/app.js

```
define([
  'js/router/Router'
], function (Router) {
  return {
    initialize: function () {
      new Router;
      Backbone.history.start();
    }
  };
});
```

注意 js/appjs 模組直接回傳的是 object

js/router/Router.js

```
define([
], function () {
  return Backbone.Router.extend({
    routes: {
      '': 'index'
    },
    index: function () {
    }
  });
});
```

Router 模組回傳的是 constructor

js/model/Config.js

```
define(function () {
  return Backbone.Model.extend({
    defaults: {
      music: false
    }
  });
});
```

沒有相依模組時，第一個參數也可以省略

js/router/Router.js

```
define([
  'js/model/Config'
], function (Config) {
  return Backbone.Router.extend({
    routes: {
      '': 'index'
    },
    index: function () {
      var config = new Config();
    }
  });
});
```

callback 的 Config 參數

即為 js/model/Config.js 回傳的 constructor

js/view/Switch.js

```
define([
  'text!template/switch.html'
], function (viewTemplate) {
  return Backbone.View.extend({
    events: {
      'click #switch': 'toggleMusic'
    },
    initialize: function () {
      this.$el.html(viewTemplate);
    },
    toggleMusic: function (e) {
      this.model.set('music', $(e.target).prop('checked'));
    }
  });
});
```

利用 text plugin 把樣版載入為字串

js/router/Router.js

```
define([
  'js/model/Config',
  'js/model/Switch'
], function (Config, Switch) {
  return Backbone.Router.extend({
    routes: {
      '': 'index'
    },
    index: function () {
      // ...
      var switchView = new Switch({
        el: '.input',
        model: config
      });
    }
  });
});
```

Model 與 View 沒有相依性
所以不需指定載入順序

js/view/Status.js

```
define([
  'text!template/status.html'
], function (viewTemplate) {
  return Backbone.View.extend({
    initialize: function () {
      this.$el.html(viewTemplate);
      this.model.on('change', this.render, this);
    },
    render: function () {
      var music = this.model.get('music');
      var status = music ? 'on' : 'off';
      $('#status', this.el)
        .removeClass('on off')
        .addClass(status)
        .text(status);
      return this;
    }
  });
});
```

因為是沒有變化的靜態樣版

所以直接在 initialize 方法中塞到 this.$el.html 中

js/router/Router.js

```
define([
  'js/model/Config',
  'js/model/Switch',
  'js/model/Status'
], function (Config, Switch, Status) {
  return Backbone.Router.extend({
    routes: {
      '': 'index'
    },
    index: function () {
      // ...
      var statusView = new Status({
        el: '.output',
        model: config
      });
    }
  });
});
```

所有的 Model 和 View 都載入後就會執行

Optimize

r.js

```
# npm -g install requirejs
```

r.js 要把 require(config) 裡的 config 一一加入

```
# r.js -o name=js/main \
  out=js/main-built.js \
  baseUrl="./" \
  paths.order="lib/requirejs/order" \
  paths.text="lib/requirejs/text"
```

out 的部份可以自訂

不一定要使用 js/main-built.js

js/build.js

```
({
  baseUrl: '../',
  name: 'js/main',
  out: 'main-built.js',
  paths: {
    order: 'lib/requirejs/order',
    text: 'lib/requirejs/text'
  }
})
```

注意 build.js 的所在位置
及 baseUrl 所對應的路徑

改用 js/build.js 最佳化

```
# r.js -o js/build.js
```

注意事項

- 一切在 App.initialize 開始。
- 因為是非同步載入的關係，所以要注意 define 的 factory 作用時機。有順序關係時，用 order plugin 。
- text plugin 載入進來的樣版，會以字串的形式一併被 r.js 編譯進去。
- 注意 factory 回傳的是純 object 還是 constructor 。
- baseUrl 很重要，代表目前 index.html 所在的網址。

其他心得

- RequireJS 支援目前常見瀏覽器，也包含行動裝置瀏覽器。 基本上看到 IE6+ 就可以放心了
- require 會利用 script 標籤的 async 屬性來載入 js 檔案。並在完成載入後，透過 readyStateChange 來決定載入的檔案是否要被執行。
- define 會使用到 queue 的機制來存放模組，然後再轉給 require 來載入。
- AMD 可以用在類似 Facebook 分段載入的情況，以提供使用者好的 UI 經驗。

the end