

# REGULAR组件设计

# REGULAR组件设计

如何优雅地写Regular组件

# 自我介绍

- ◎ 赵雨森
- ◎ 杭州研究院 - 前端技术部
- ◎ 2014年入职，毕业于西安交通大学
- ◎ [rainfore@github](https://github.com/rainfore), [hzzhaoyusen@corp](mailto:hzzhaoyusen@corp)

# Regular

a concise, flexible framework for creating data-driven component

[Source on Github](#)[Download Latest](#)

BUILT BY

[!\[\]\(23d9fc146e83b5c3013cfa32c784f8d5\_img.jpg\) Star](#)[!\[\]\(c694a3ff3b077d76910920a6a1593ab4\_img.jpg\) Fork](#)[!\[\]\(ec9132f1d27c8919987d92907322654d\_img.jpg\) Tweet](#)

## Creating Component with MVVM Pattern

you will enjoy it just like you enjoy angularjs :)



### Powerful

live templating is string-based, take more control on templating logic.



### Concise

data-binding, directive, filter, event and animation... all of them is supported out of box with concise API



### Flexible

compeletely self-contained, easily integrated and well encapsulation. make it be friendly with large-project

MV\*





RegularJS



# 山寨

Regular = React + Angular

# CSS预处理器





# Writing abstract and modular CSS with MCSS

MCSS是一个CSS Preprocessor, 语法上基于[CSS3 Syntax](#)的超集, 提供 **Nested Ruleset, Variable, first-class function(or mixin), custom atrule**(@extend、@import、@abstract...)等等特性来填补原生CSS的抽象能力弱的缺陷, 帮助我们书写抽象化的CSS

MCSS是有丰富的语言特性的一个DSL, 它甚至允许扩展 `@atrule` 自定义解释策略; 与此同时MCSS是一个易用使用的CSS Parser, 并提供便利化的方式去操作树形结构, 以实现csscomb、prefixr等CSS工具.

MCSS完全使用javascript构建, 你可以分别在browser(ES5 support needed)和nodejs中使用它

目前主页正在建设中, 这是临时性介绍页, 你可以先 [动手试试](#)

有兴趣可以查看下mcss的[实例函数库](#)(类似compass), 你会发现几乎所有在mcss中都可以封装成函数的形势, 如果你愿意

## 安装

### Nodejs

```
npm install -g mcss
```

### Browser

```
<script src="https://github.com/leeluolee/mcss/blob/master/dist/mcss-latest.js"></script>
```

需要支持ES5的浏览器, 绝对只建议在线上环境使用compile后的css文件, 而不是即时compile;

山寨达人



Regular UI

[开始使用](#)

[CSS元件](#)

[CSS模块](#)

[JS元件](#)

[JS模块](#)

## 一款基于RegularJS的前端组件库

轻量、简洁、高效

 [快速开始](#)

 [下载最新](#)

37

★ STARS

0.2.0

📦 LATEST VERSION

17

🍴 FORKS

 [VIEW ON GITHUB](#)

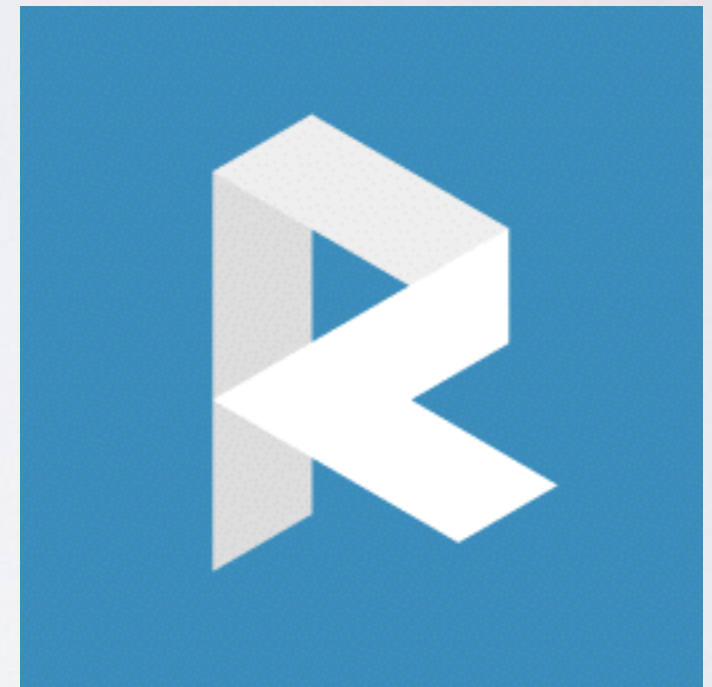
# 参考

- ◉ 公司系： NEC、NEJ
- ◉ jQuery系： Bootstrap、Ulkit、Semantic UI、Amaze UI、Foundation、MaterializeCSS
- ◉ Angular系： Angular UI、Lumx、Angular Material
- ◉ React系： React Bootstrap、Ant Design、Material UI
- ◉ WebComponent系： Polymer
- ◉ 其他： Kendo UI、ExtJS
- ◉ Native： Android Developers、WPF、Adobe Flex

山寨到底

Regular UI v0.1 → Regular UI v0.2

- 包含20多个CSS组件、30多个JS组件
- 支持自定义打成UMD包
- 兼容到IE8
- 目前在三四个项目中实践（踩坑）





组件化

# 组件的概念

在前端开发领域，**组件**是一种对交互元素的封装

“Keep Simple. Everything can be a component.”

——ReactJS

“Keep Simple. Everything can be a component.”

——ReactJS

神马都是组件



网(易)独(家)

有(声)小(说)

# 地火明夷

第四回

带你进入一个奇妙的小说世界



下载客户端

PC 安卓 iPhone WP iPad Mac 六大客户端

## 热门推荐

华语 | 流行 | 摇滚 | 民谣 | 电子

更多



登录网易云音乐，可以享受无限收藏的乐趣，并且无限同步到手机

用户登录

入驻歌手

看大牌明星的私房歌单



夏恋 Otokaze

3.7万

00:00 / 00:00



2

```
<app>
  <g-top>
    <logo /><navbar /><searchBox /><menu />
  </g-top>
  <g-hd><carousel /></g-hd>
  <g-bd>
    <g-mn>
      <column title="热门推荐">...</column>
      <column title="个性化推荐">...</column>
      <column title="新碟上架">...</column>
      <column title="榜单">...</column>
    </g-mn>
    <g-sd>
      <userInfo />
      <sideColumn title="入驻歌手" />
      <sideColumn title="热门DJ" />
    </g-sd>
  </g-bd>
  <g-ft>...</g-ft>
  <player>
    <slider />
    <playList />
    ...
  </player>
  <gotop />
  ...
</app>
```



分类

# 按通用性来分类

- 通用组件： <carousel><pager><tabs><modal>
- 通用业务组件： <column><loginModal>
- 业务组件： <app><logo><userInfo><g-hd><g-bd>

# 按使用场景来分类

- 元件: <button><icon><progress>
- 模块: <carousel><column><player><listView>
- 布局: <g-top><g-hd><g-bd><g-ft>
- 功能: <validation><draggable><resizable>

# 按代码来分类

- HTML: `<button><input><select>`
- HTML+CSS: `<button class="u-btn">`  
`<span class="u-badge">`
- JS: `<validation><draggable><resizable>`
- HTML+CSS+JS: `<carousel><modal><player>`

# 实践 I

## Demo

```
npm install -g rgui-tools  
git clone -b next --depth 1 https://github.com/regular-ui/regular-ui.git demo-music  
rm -rf .git
```

```
npm install  
rgui-tools list --watch
```



# 规范和细节

[戳这里](#)



# 命名规范

- jQuery系: autocomplete, datettimepicker, progressbar, productdatagrid
- React系: <Navbar><ListGroup><ProgressBar>  
<ProductDataGrid>
- Angular系和WebComponent系: <md-menu-bar>  
<paper-radio-group><uid-progressbar>

# 命名的一致性

- 组件名和类名：Modal, DetailModal, ListView, DatePicker
- 对象名：modal, detailModal, listView, datePicker
- 标签名：<modal><detailModal><listView>
- CSS：m-modal, m-detailModal, m-listView

# 实践2

创建一个App和Player组件

# CSS规范

遵循NEC规范，做一些补充



# class的私有化

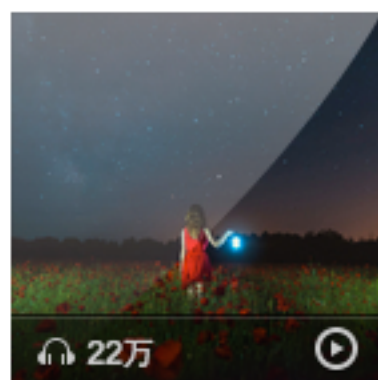
## 热门推荐

华语 | 流行 | 摇滚 | 民谣 | 电子

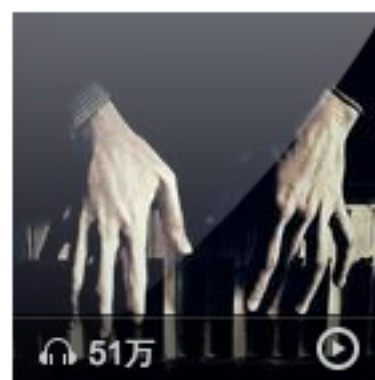
更多 →



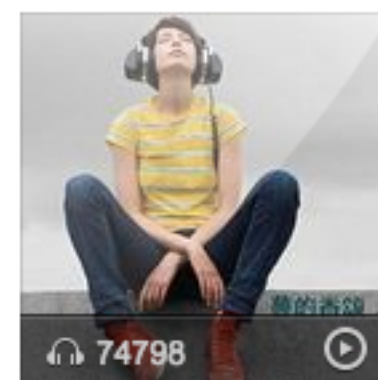
56万  
【校园歌曲大赛】不唱情歌，唱什么腻？



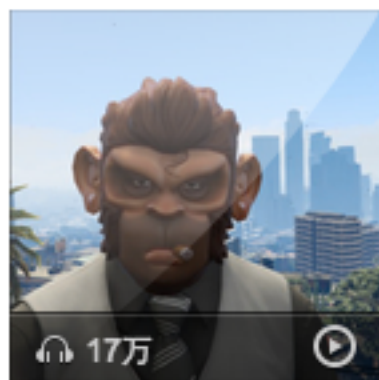
22万  
【冷门/女声】夜深了，花睡了



51万  
古典钢琴曲



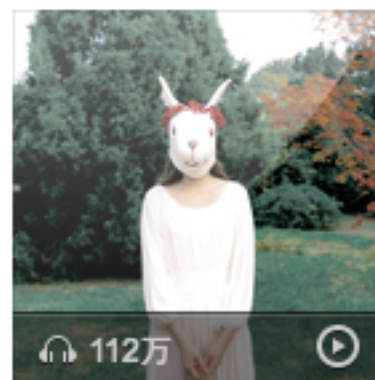
74798  
电台节目 一人一首成名曲——yeye浪潮篇



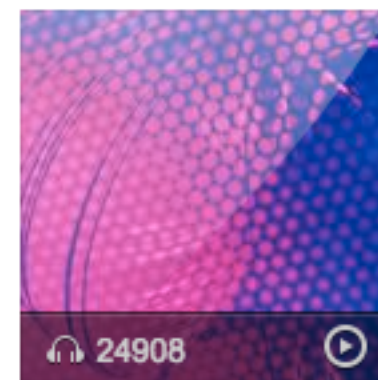
17万  
HoHo游戏视频 BGM(持续更新ing)



22114  
电台节目 【吻安】你可以习惯一个人生活吗？



112万  
五月民谣 青葱岁月里的失眠时光



24908  
电台节目 极致电音



```
<column title="热门推荐">
  <playlist title="古典钢琴曲" />
  <playlist title="极致电音" />
  <playlist title="夜深了" />
  <playlist title="五月民谣 青葱岁月" />
</column>
```

```
<div class="m-column">
  <div class="head">
    <h3 class="title">热门推荐</h3>
  </div>
  <div class="body">
    <div class="u-playList">
      <div class="cover">
        
        ...
      </div>
      <div class="title">古典钢琴曲</div>
    </div>
    ...
  </div>
</div>
```

```
.m-column .head {...}
.m-column .head .title {...}
.m-column .body {...}

.u-playList .cover {...}
.u-playList .cover img {...}
.u-playList .title {...}
```

# 方法一

```
.m-column>.head {...}  
.m-column>.head>.title {...}  
.m-column>.body {...}  
  
.u-playList>.cover {...}  
.u-playList>.cover>img {...}  
.u-playList>.title {...}
```

## Semantic UI

```
.ui.list>.item>.icon+.content, .ui.list>.item>.image+.content {  
  display: table-cell;  
  padding: 0 0 0 .5em;  
  vertical-align: top;  
}
```

缺点： 层级如果太深，影响性能

# 方法二

```
.m-column .column_head {...}  
.m-column .column_title {...}  
.m-column .column_body {...}  
  
.u-playList .playList_cover {...}  
.u-playList .playList_cover>img {...}  
.u-playList .playList_title {...}
```

## Bootstrap

```
.panel .panel-heading {...}  
.panel .panel-title {...}  
.panel .panel-body {...}  
.panel-primary .panel-heading {...}
```

缺点：有时名字可能比较长

# 实践3

完善Player组件样式

# MCSS规范

## 项目中组件的样式不是孤立的

Link

按钮

选项1

上一页

1

2

...

4

5

6

...

10

11

下一页

Primary

Content



# 定义变量

```
.u-btn {  
  padding: 0 12px;  
  $line-height: $height-base;  
  
  background: $brand-primary;  
  color: white;  
  $border: $border-base;  
  $border-radius: $border-radius-base;  
}
```

```
.m-panel {  
  background: white;  
  border-top: 3px solid $brand-primary;  
  $border-radius: $border-radius-base;  
  $box-shadow: 0 1px 1px rgba(0, 0, 0, 0.15);  
  
  .panel_hd {  
    padding: 10px;  
    border-bottom: 1px solid $brand-default;  
  }  
}
```

# 定义变量

- 变量的命名必须有意义
  - 正确示例: `$brand-primary`, `$font-size-base`, `$u-radio_size`
  - 错误示例: `$color1`, `$fs0`, `$bg3`

# 定义变量

- 也不一定要把所有的值都定义成变量

```
.u-btn-special {  
  padding: 0 40px;  
  $line-height: $height-base;  
  
  background: #f5f9fb;  
  color: #999;  
  $border: #e1e8ed;  
  $border-radius: $border-radius-base;  
}
```

# 主要色系

- 一个项目中的主要色系以`$brand-`开头

```
/* Brand Colors */  
$brand-primary = #3c8dbc;  
$brand-info    = #00c0ef;  
$brand-success = #00a65a;  
$brand-warning = #f39c12;  
$brand-error   = #dd4b39;
```

# 扩展样式

- 常见状态: default, primary, info, success, warning, error, disabled, muted
- 大小分级: xxs, xs, sm, base, lg, xl, xxl
- 颜色分级: darkest, darker, dark, base, light, lighter, lightest



- 尽量使用[MASS库](#)中的函数简化代码，如各种CSS3的前缀等。

```
.u-input,  
.u-select,  
.u-textarea {  
    $border-radius: 2px;  
    $transition: all linear 0.2s;  
    $placeholder({  
        color: $u-field_placeholder_color;  
        $opacity: 1;  
    });  
}
```



# 实践4

完善整体页面样式

# JS规范

- JS代码遵循Google的JavaScript规范（中文版）；
- ES6代码遵循阮一峰《ECMAScript 6入门》中的编程风格；
- 代码的注释遵循jsDoc的规范。

# Regular规范

- Regular UI的所有组件都继承自Component基类

```
const Component = Regular.extend({...});  
  
const Modal = Component.extend({...});  
const Navbar = Component.extend({...});  
const Carousel = Component.extend({...});  
...
```

# 初始化data是个好习惯

```
const A = Component.extend({
  data: {
    value: null,
    size: 'normal',
    list: [],
  }
});
```

```
const A = Component.extend({
  config() {
    this.data = {
      value: null,
      size: 'normal',
      list: [],
    }
  }
});
```


```
const A = Component.extend({
  config() {
    data.value = data.value || null;
    data.size = data.size || 'normal';
    data.list = data.list || [];
  }
});
```

```
const A = Component.extend({
  config() {
    this.data = Object.assign({
      value: null,
      size: 'normal',
      list: [],
    }, this.data);
  }
});
```




# 初始化data是个好习惯


```
const A = Component.extend({
  data: {
    value: null,
    size: 'normal',
    list: [],
  }
});
```




```
const A = Component.extend({
  config() {
    this.data = {
      value: null,
      size: 'normal',
      list: [],
    }
  }
});
```



```
const A = Component.extend({
  config() {
    data.value = data.value || null;
    data.size = data.size || 'normal';
    data.list = data.list || [];
  }
});
```



```
const A = Component.extend({
  config() {
    this.data = Object.assign({
      value: null,
      size: 'normal',
      list: [],
    }, this.data);
  }
});
```



# 最佳实践

```
const Component = Regular.extend({  
  /**  
   * @protected  
   */  
  config() {  
    this.data = Object.assign({  
      readonly: false,  
      disabled: false,  
      visible: true,  
      'class': '',  
    }, this.data);  
    this.supr();  
  },  
})  
.filter(filter)  
.directive(directive);
```

```
var Component = Regular.extend({  
  /**  
   * @protected  
   */  
  config: function () {  
    Regular.util.extend(this.data, {  
      readonly: false,  
      disabled: false,  
      visible: true,  
      'class': '',  
    });  
    this.supr();  
  },  
})  
.filter(filter)  
.directive(directive);
```

# config & init

- 数据处理放在config里面
- DOM操作放在init里面

```
const A = Regular.extend({  
  config(data) {  
    data.size = 'normal';  
    ...  
  },  
  init(data) {  
    let $element = this.$refs.element;  
    ...  
  },  
});
```

Regular中尽量的不要操作DOM

# 模板风格

- 模板尽量采用HTML5风格
  - 字符串加双引号
  - bool属性不加{true}或{false}
  - 表达式用{}且不加双引号
- 每个组件都实现class扩展

# 最佳实践

```
<progress percent=72 state="error" striped active />  
<pager class="m-pager-custom" current=6 total=11 on-nav={this._onNav($event)} />
```





```
  
<input type="text" on-change={this._onChange($event)}>  
<pager current=6 total=11 on-nav={this._onNav($event)} />
```

# \$update

- 组件内部异步回调函数中使用

```
const A = Regular.extend({
  foo() {
    ajax.get(url, (result) => {
      this.data.value = result.value;
      this.$update();
    });
  }
});
```

- 外部修改组件数据时使用

```
let a = new A();
a.data.value = 18;
a.$update();
```

# 实践5

完成Navbar和Carousel

- ⦿ filter
- ⦿ computed
- ⦿ directive
- ⦿ event

# 实践6

完成MusicListView和Slider



# 总结

# 总结

- 组件的概念
- 组件的分类
- 规范和细节
- 云音乐案例

# THANKS

Regular UI POPO交流群：1319383