SOLUTIONS

陈磊

2018 年 3 月 23 日

目录

1 代码规范 **1**
1.1 编辑器文本规范 1
1.2 命名规范 1
1.2.1 CSS 命名 1
1.3 代码检查 2
1.3.1 CSS 格式化 2
1.3.2 JS 静态代码检查工具 2
1.3.3 JS 语法规范 2

2 兼容性问题解决 **2**
2.1 IE: 盒模型 2
2.2 IE 8: map 2
2.3 IE 8: fontawesome 图标显示为方块 3
2.4 IE 10+ 浏览器定位 3
2.5 IE 6-8 CSS3 媒体查询 (Media Query) 4

3 跨域资源共享/CORS (Cross-origin resource sharing) **4**

4 跨站请求伪造/CSRF (Cross-site request forgery) **4**

5 HTML **4**
5.1 参数 (input) 在 form 之外 4

1 代码规范 2

6 模块化与组件化 **5**
6.1 实现模块化 5
6.2 实现组件化 5

7 权限管理 **5**
7.1 AngularJS 分角色登录 5
7.2 Vue.js 权限管理 6

8 npm 包及私有库 **7**
8.1 npm 包编写 7
8.2 私有库方案 7

9 前后端分离 **8**
9.1 历史 8
9.2 目标/方法 8
9.3 应用 9
9.4 引入 nodejs 层的应用场景 9
9.5 扩展 9

1 代码规范

1.1 编辑器文本规范

规范文件采用的换行符、缩进方式以及编码等等.

1. EditorConfig: `http://editorconfig.org/`
2. VSCode 插件: EditorConfig for VS Code, 可生成配置样本.

`.editorconfig`配置样例

```
root = true

[*]
indent_style = space
indent_size = 2
charset = utf-8
end_of_line = lf
insert_final_newline = true
trim_trailing_whitespace = true
max_line_length = 80
```

1.2 命名规范

1.2.1 CSS 命名

BEM, SMACSS, OOCSS

1.3 代码检查

1.3.1 CSS 格式化

1. CSScomb: `http://csscomb.com`
2. 配置文件`.csscomb.json`示例:`https://github.com/htmlacademy/codeguide/blob/master/csscomb.json`
3. VSCode 插件: CSScomb

1.3.2 JS 静态代码检查工具

1. ESLint: `https://github.com/eslint/eslint`
2. JSLint: `https://github.com/jshint/jshint/`

1.3.3 JS 语法规范

1. airbnb: `https://github.com/airbnb/javascript`
2. standard: `https://github.com/standard/standard`

2 兼容性问题解决

2.1 IE: 盒模型

IE 默认情况下长宽包含 `padding` 和 `border`, 和其他浏览器的长宽存在区别, 建议添加 `border-sizing` 属性. 保持多个浏览器的一致性.

```
box-sizing: border-box;
```

2.2 IE 8: map

IE8 不支持 JavaScript 原生 map 函数, 可在任意地方加入如下的代码片段[1].

[1] Is the javascript .map() function supported in IE8?

```
(function (fn) {
    if (!fn.map) fn.map = function (f) {
        var r = [];
        for (var i = 0; i < this.length; i++)
            if (this[i] !== undefined) r[i] = f(this[i]);
        return r
    }
    if (!fn.filter) fn.filter = function (f) {
        var r = [];
        for (var i = 0; i < this.length; i++)
            if (this[i] !== undefined && f(this[i])) r[i] = this[i];
        return r
    }
})(Array.prototype);
```

或者用 jQuery 的 map 函数.

```
// array.map(function( ) { });
jQuery.map(array, function( ) {
}
```

2.3 IE 8: fontawesome 图标显示为方块

对于修饰性不影响功能的图标, 可以做降级处理, 仅在非 IE 或者 IE9+ (条件注释[2]) 情况下引入 fontawesome 图标库. (谷歌搜索了一堆方案都没用, 最后应用这种方式来解决).

```
<!--[if (gt IE 8) | !IE]><!-->
<link rel="stylesheet" href="font-awesome.min.css">
<!--<![endif]-->
```

2.4 IE 10+ 浏览器定位

IE 10+ 不支持条件注释, 因此需要其他方式定位这些浏览器. 如果只增加 CSS, 可采用以下方式定位[3].

```
/*IE 9+, 以及 Chrome*/
@media screen and (min-width:0\0) {
```

[2] About conditional comments

[3] How do I target only Internet Explorer 10 for certain situations like Internet Explorer-specific CSS or Internet Explorer-specific JavaScript code?

```
}

/*IE 10*/
@media all and (-ms-high-contrast: none), (-ms-high-contrast:
    ↪ active) {
}

/*Edge*/
@supports (-ms-accelerator:true) {
}
```

另一种方式是 JavaScript 检测浏览器版本, 在 `body` 标签为特定浏览器添加 `class` 属性标识.

2.5 IE 6-8 CSS3 媒体查询 (Media Query)

引入 Respond.js.

```
<!--[if lt IE 9]>
<script src="respond.min.js"></script>
<![endif]-->
```

3 跨域资源共享/CORS (Cross-origin resource sharing)

- TAT.Johnny, iframe 跨域通信的通用解决方案, alloyteam
- JasonKidd,「JavaScript」四种跨域方式详解, segmentfault

4 跨站请求伪造/CSRF (Cross-site request forgery)

5 HTML

5.1 参数 (input) 在 form 之外

input 在 form 之外时, 在 input 元素内添加 form 属性值为 form 的 ID[4]. 这样 input 仍然可以看做隶属于此表单, jQuery `$('#formid').serialize();` 能够获取 form 之外的输入框值. 或者在提交 (`submit`) 表单时会同样提交 `outside` 这个值.

```
<form id="formid" method="get">

   <label>Name:</label>
   <input type="text" id="name" name="name">

   <label>Email:</label>
   <input type="email" id="email" name="email">
   <input type="submit" form="contact_form" value="send form" />
</form>
<input type="text" name="outside" form="formid">
```

注意: IE8 `$('#formid').serialize();` 无法获取 `outside` 值.

6 模块化与组件化

内嵌框架图 (embedded)

模块化, 强调内聚, 包含完整的业务逻辑, 可以方便业务的复用. 组件化, 强调复用, 重点在于接口的暴露, 和构件的概念类似.

6.1 实现模块化

业务相关的特殊性都应包含在同一个模块内, 具体到前端, 这些特性包括与业务相关的接口、状态、路由等.

[4]PLACING FORM FIELDS OUTSIDE THE FORM TAG

6.2 实现组件化

关键是如何暴露接口, 方便外部复用.

7 权限管理

7.1 AngularJS 分角色登录

不同角色/权限登录后所见菜单不一样. 方案如下:

1. 给不同的路由配置其角色/权限属性.
2. 登录进入时, 记录角色/权限.
3. 进入主页, 根据角色/权限构建菜单 (view 中包含全部菜单, 非此角色菜单移除 Dom).
4. 点击菜单进入到对应路由时, 根据判断路由的角色/权限属性是否和登录进入时记录的一样.

此方案包括两部分的权限限制, 其一是将不必要的菜单移除 Dom, 但菜单对应的路由依然可用, 只是在页面上没有对应可操作的视图, 其二是路由和登录的角色/权限匹配. 以 AngularJS 为例, 对应每个步骤的代码如下.

1. 路由属性.

```
$stateProvider.state('Registration.Instructors', {
    url: "/Instructors",
    templateUrl: '/Scripts/App/Instructors/Templates/instructors.
        ↪ html',
    controller: 'InstructorController',
    data: { auth: "Admin"}
})
```

2. 登录用户权限/角色信息可记录到 `rootScope` 中, 比如 `rootScope.adminType = "Admin"`.
3. 菜单保留与移除. `ng-if="adminType==='Admin'"`[5].
4. 路由和登录角色/权限匹配[6].

```
app.run(function($rootScope){
  $rootScope.$on('$stateChangeStart', function(event, toState,
      ↪ toParams, fromState, fromParams){
    if ( toState.data.auth !== $rootScope.adminType ) {
```

[5] what is the difference between ng-if and ng-show/ng-hide

[6] angularjs: conditional routing in app.config

```
        event.preventDefault();
        return false;
    }
  })
});
```

7.2 Vue.js 权限管理

- 基于 Vue 实现后台系统权限控制
- 用 addRoutes 实现动态路由
- 手摸手，带你用 vue 撸后台系列二 (登录权限篇)
- Vue 后台管理控制用户权限的解决方案？
- 自定义指令
- https://codepen.io/diemah77/pen/GZGxPK

8 npm 包及私有库

8.1 npm 包编写

npm 包通常会兼容不同的应用场景: nodejs、require.js 和浏览器. 所以会包含一段用于判断运行环境的代码, 如下.

```
// from Vue.js
(function (global, factory) {
  typeof exports === 'object' && typeof module !== 'undefined' ?
      ↪ module.exports = factory() :
  typeof define === 'function' && define.amd ? define(factory) : (
      ↪ global.NPM_THING = factory());
}(this, (function () {
  'use strict';
  // code
  // return NPM_THING;
})));
```

1. 判断是否是 nodejs 环境: `typeof exports === 'object' && typeof module !== 'undefined'`.
2. 判断是否是 require.js: `typeof define === 'function' && define.amd`

其中的 `this` 指向全局变量, nodejs `this` 为 `global`, 浏览器中 `this` 为 `window`.

8.2 私有库方案

为了避免重复造轮子, 提供编码效率, 同时又可以避免企业内部的业务逻辑暴露, 于是对私有库有需求. 期望, 如果私有库中有, 则从私有库中下载, 否则从公开的库中下载.

npm 的包都是公开的, 提供的企业私有化方案是收费的. 开源方案有:

1. cnpm: `https://github.com/cnpm/cnpmjs.org`
2. sinopia: `https://github.com/rlidwka/sinopia`

两者的对比 (企业私有 npm 服务器):

–	cnpm	sinopia
系统支持	非 windows	全系统
安装	复杂	简单
配置	较多，适合个性化需求较多的	较少
配置——修改默认镜像	不支持	支持
存储	mysql	文件格式，直观
服务托管	默认后台运行	pm2, doker, forever
文档资料	较多	较少

9 前后端分离

内容简述: 简单描述前后端分离的历史状况, 明确前后端划分的原则: 后端面向数据, 前端面向用户, 在服务端引入 nodejs 可以解决的问题 (应用的场景).

9.1 历史

1. 前后端耦合. 例如, ASP.NET Webform 和 jsp 的标记语言的写法, 每次请求由后端返回, 且后端的语言变量混在 HTML 标签中.
2. 前后端半分离. 例如, ASP.NET MVC 和 Spring MVC 视图由后端控制, V (视图) 由前端人员开发. 开发新的页面需要后端新建接口, 编程语言通常在一个工程中, and so on.
3. 前后端完全分离. 前后端通过接口联系. 前后端会有部分逻辑重合, 比如用户输入的校验, 通常后端接口也会处理一次. 前端获取数据后渲染视图, SEO 困难.

9.2 目标/方法

1. 后端: 数据处理; 前端: 用户交互[7].
2. 前端向后扩展 (服务端 nodejs): 解决 SEO、首屏优化、部分业务逻辑复用等问题; 前端向前扩展: 实现跨终端 (iOS 和 Android, H5, PC) 代码复用.

解决的问题:

1. core: 优化交互体验, 提高编码效率.
2. SEO.
3. 性能优化.
4. 首屏优化.
5. 代码复用 (业务逻辑, 路由, 模板).

9.3 应用

1. 服务端框架: Koa, Express.
2. 同构框架支持: nuxt.js(可用 Koa 替换 Express).
3. 路由用 history mode (Vue.js), 如果后端不配置, 直接进入页面无法访问. 可复用模板, 直接访问时后端渲染, 路由访问时前端渲染[8].
4. 服务端, 浏览器端及 Native 端都可应用的第三方库: axios, moment.js.

9.4 引入 nodejs 层的应用场景

除了上面所述的性能和 SEO 等问题, 还可以作为中间件, 抹平同类型系统的差异, 构建统一的平台.

比如, 对于多个定制化的产品, 每个产品都对应有运营平台, 用于观测用户使用情况. 由于历史上造成的差异, 每个运营平台都需要重新构建一套运行于浏览器端的前端工程. 对于这种业务相似度较高的情况, 就可以在服务端引入 nodejs, 构建统一的平台, 抹平已有系统之前的差异 (比如接口有不同的风格), 只需要实现一套 Web APP. 同时也方便了后期其他定制产品的扩展.

9.5 扩展

1. 美团点评点餐, 美团点评点餐 Nuxt.js 实战, 2017-08-09

[7] Balint Sera, On the separation of front-end and backend, 2016-06-15

[8] 赫门, 淘宝前后端分离实践, 2014

2. Jason Strimpel, Maxime Najim, 同构 JavaScript 应用开发, 2017
3. Nicholas C. Zakas, Node.js and the new web front-end, 2013-10-07