## 主要可参考3个代码文件：

核心:

main/index.js, （模块配置处理等)

例子:

content/developing/developing.js, （非define方式写的模块）

content/devTest/devTest.js （define方式写的模块）

## 主题思路

1. 利用requirejs的模块化加载与开发模式，稍微做些配置来兼容我们已有的一些组件；requirejs不重复加载文件，可以避免依赖文件相互覆盖的问题
2. 模块定义可以使用requirejs的define方式，也可以采用普通方式，define方式不需额外工作，普通方式需要做些配置，这些配置会统一处理后交给require.config来注册非requirejs方式的模块
3. 模块文件采用构造函数的写法
4. 在菜单入口调用中加载模块文件并实例化对象
5. 抛弃了我们经常使用的includePage, 菜单入口函数会注入容器div的id, 模块在这里面绘制，于是不再依赖html文件，可以动态绘制页面，也可以自己取得html模板填入（note: 使用i18n的{{}}记法时，需要调用patchHTML处理后再使用），具体可以参考示例文件

## 目录结构

参考广告系统的目录结构，此次单独开出了main目录，分块文件通常放在content下

## 方法实现

1. 全局对象App
2. 非requirejs方式定义的模块配置在App.modules中，目前这里已经写了一些模块，还不完全，可以后续自定义（uiWidget下的组件均是构造函数的写法，满足我们现在的需求，可以添加进来）
3. 这些模块利用requirejs对非define模块的配置 shim，paths来注册
4. 模块配置参数示例：

StyledList: {

isRequireModule: false,

depends: [],

exports: 'StyledList',

path: 'uiWidget/styledList',

}

1. 名称’StyledList’, 向shim,path注册的key值
2. isRequireModule: false, false表明非define方式定义的模块
3. depends: requirejs shim对应的deps，表明依赖
4. exports: requirejs shim的exports, 表示模块定义的全局构造函数，
5. path: requirejs paths 相应key对应的值
6. 这些配置经过处理后给require.config就实现了非define方式模块的注册
7. 菜单入口App.navToxxxxx

这里是进入模块的入口，通过require方法加载模块文件，然后实例化模块，并依据containerId开始执行页面绘制

1. define方式定义模块，因为遵循requirejs，可以直接加载
2. 非define方式定义的模块也是通过require()加载，只是需要到App.modules注册，以经过处理注册到requirejs

3）参考App.navToDeveloping, App.navToDevTest

1. 模块定义

整体思路都是定义构造函数，在菜单入口调用时才实例化，目的是为了应对可能有同一页面的多个实例的情况，是否采用define方式都可以

## 其他

1. sha1.js和endecodebase64.js为了避免污染全部变量做了封装，endecodebase64的使用没有变化，因为原来就是附在my上的，sha1的常用方法，现在也改附到my上了

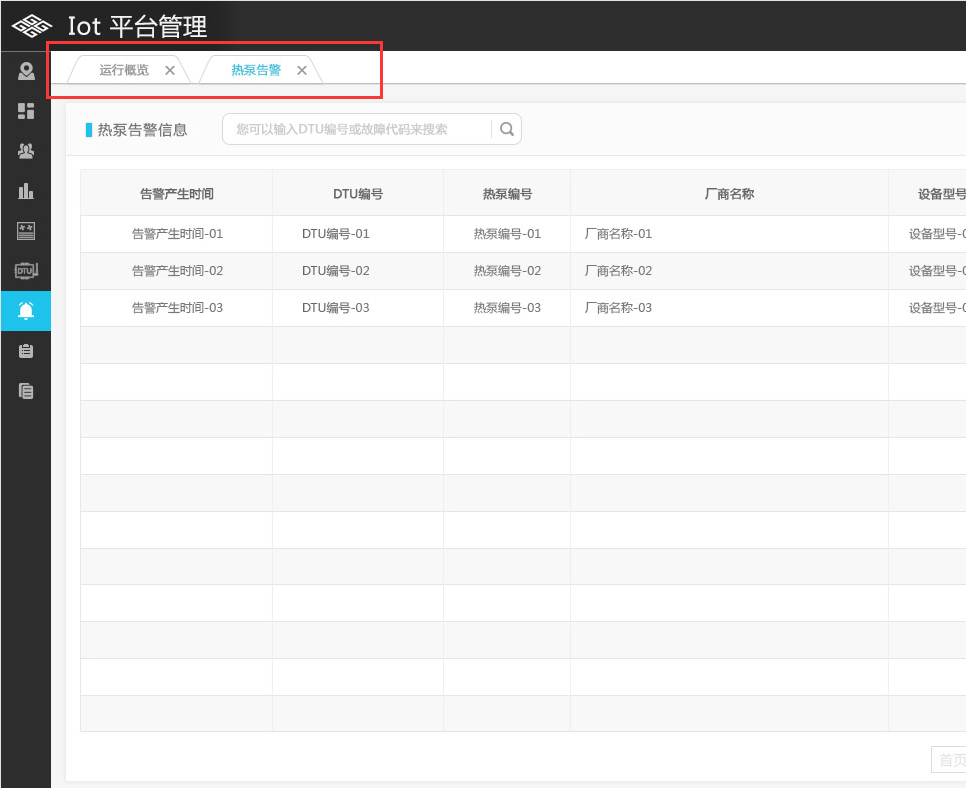
**常见问题F&Q**

**从UI设计上看多个页面能够支持同时打开,但是否会出现同一个页面多开?**

>>> 目前暂时不支持,但是从开发的角度,一定要预留这种可能性.

要注意规避同一个页面被多开时, id冲突问题.

多个不同页面间命名规则遵循: 模块名\_私有名称,用以防止冲突.



**如何避免ID重复**

鉴于我们之前使用的一些组件会要求容器等内容的id,

而现在如果我们写id的话，会不支持页面多开，

所以提供了一个方法 App.getAnotherId()

它可以返回一个在App生命周期内递增的数，

如果确实需要id的话，可以附上它，来避免id重复，

**代码中命名建议**

JS命名规范

标准变量命名使用驼峰式命名

示例：let thisIsMyName;

常量全部大写，并使用下划线连接

示例：const MAX\_COUNT = 10;

构造函数大写第一个字母

示例：function Person(){}

文件名称的命名规则

统一用小写的英文字母，数字和下划线的组合，不得包含汉字空格和特殊字符

示例：

register\_form.aspx，register\_post.aspx，topic\_lock.aspx

banner\_sohu.gif, banner\_sina.gif

menu\_aboutus.gif,menu\_job.gif

title\_news.gif

logo\_police.gif

pic\_people.gif

**Require js 库的使用方式**

**参考**<http://www.requirejs.cn/>

**构造函数推荐写法**

混合的构造函数/原型方式 或者 动态原型方法

参考: <http://www.w3school.com.cn/js/pro_js_object_defining.asp>

**是否能使用ES6?**

原则上可以, 但是必须做过ES6转ES5后的兼容性测试. 既:保证ie10以上浏览器能够运行(因为ie11以下版本2016.1月微软就宣布停止支持了).火狐浏览器和chrome浏览器支持.

**表格组件怎么使用?**

表格组件(刘壬)已做适当改进, 外部文件新的调用此组件格式举例：

let  list\_selfServiceUser = new NewStyledListVersionTwo ({

         rows: 14,

         columns: 9,

         container: jQuery('#modules\_container'),

         titles:[{label:”title”}, {label:”title”}, {label:”title”},

{label:”title”},….], // 这个和以前的表格组件格式是一样的，是一个数组。

         listType: 1,

         data: [],

         styles: { borderColor: 'transparent',

                   borderWidth: 1,

                   titleHeight: 46,

                   titleBg: '#f8f8f8',

                   titleColor: '#666666',

                   cellBg: 'white',

                   evenBg: '#f8f8f8',

                   cellColor: 'rgb(114,114,114)',

                   footColor: 'rgb(121,121,121)',

                   iconColor: 'rgb(93,161,192)',

                   inputBorder: '1px solid rgb(203,203,203)',

                   inputBg: 'white',

                   columnsWidth: [0.12,0.083,0.222,0.076,0.117,0.079,0.07,0.

07,0.146]

         }

});

原因是发现表格存在每行中多列与其他列样式不同（比如表格一共有7列，其中5列文字

居中，2列左对齐）

目前外部新的引用方式如下：

（1）表格body中多列列样式和其他列不同 时，

var list\_table = new NewStyledListVersionTwo({

  rows: 7,

  columns: 7,

  container: jQuery("table\_container"),

  titles: titles\_label\_array,

  listType: 1,

  data: [],

  styles: { borderColor: 'transparent',

                 borderWidth: 1,

                 titleHeight: 46,

                 titleBg: '#f8f8f8',

                 titleColor: '#666666',

                 cellBg: 'white',

                 evenBg: '#f8f8f8',

                 cellColor: 'rgb(114,114,114)',

                 footColor: 'rgb(121,121,121)',

                 iconColor: 'rgb(93,161,192)',

                 inputBorder: '1px solid rgb(203,203,203)',

                 inputBg: 'white',

                 columnsWidth:

[0.1444,0.1285,0.09531,0.2397,0.08,0.1596,0.1497]

        },

tableBodyTdDiffStyles:{"column":[4,6],"className":["table\_td\_basic\_two","tab

le\_td\_basic\_three"]}

//  这句意思是第4列，第6列需要修改样式，第4列样式为table\_td\_basic\_two，第6列

样式为 table\_td\_basic\_three

（2）表格body中多列列样式和其他列不同  这种情况 不存在时，原本表格控件使用方

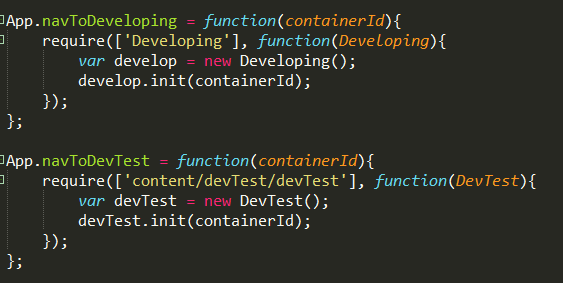
法不变。外部调用时不必增加代码。（不用增加  tableBodyTdDiffStyles：… 这行）

**关于菜单的部分主框架**

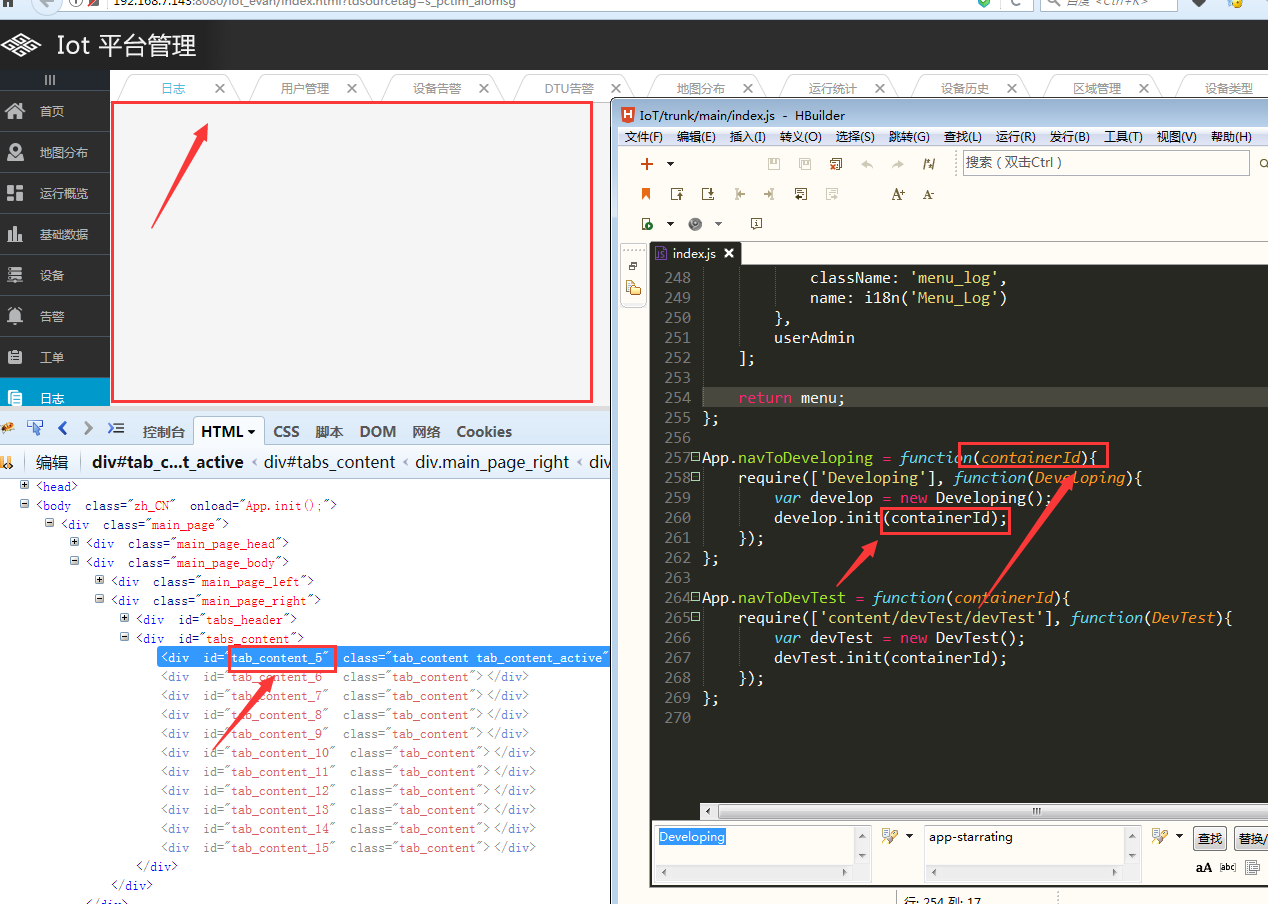
马强已经给出大部分内容如下



但是具体的初始化内容,还是需要每个模块自己写



**TAB容器ID的使用**



**弹框组件**

关于iot项目中的弹框组件，

弹框组件的js还是用之前项目使用的popupDialog组件。外部调用方法和以前一样。

只是这次各页面弹框中css内容，我这里统一了一下，希望大家用统一的样式。

弹框中可能要用到的css:  uiWidget\styles\popupDialog\_v2.css

弹框可能要用到的图片目录：uiWidget\images\

Html 例子如下：

图片:



Html 代码：

新的弹框例子（代码）如下：

<html>

<head></head>

<body>

  <div class="popup\_dialog\_v2">

   <div class="popup\_dialog\_v2\_header">

    <div class="popup\_dialog\_v2\_header\_left">

     <div class="popup\_dialog\_v2\_header\_left\_bar"></div>

     <div class="popup\_dialog\_v2\_header\_left\_title">

      安装信息2

     </div>

    </div>

    <div class="popup\_dialog\_v2\_close popup\_dialog\_clear"></div>

   </div>

   <div class="popup\_dialog\_v2\_body">

    <div class="popup\_dialog\_v2\_input\_container" style="float:left">

     <div class="popup\_dialog\_v2\_input\_lable">

      热泵编号

     </div>

     <input class="popup\_dialog\_v2\_input\_box" disabled="disabled" value="热

泵编号-01" />

    </div>

    <div class="popup\_dialog\_v2\_input\_container"

style="float:left;margin-left:55px">

     <div class="popup\_dialog\_v2\_input\_lable">

      热泵编号

     </div>

     <div class="popup\_dialog\_v2\_selector">

      <div class="popup\_dialog\_v2\_input\_box\_select">

       热泵编号-01

      </div>

      <div class="popup\_dialog\_v2\_select\_icon"></div>

     </div>

    </div>

   </div>

   <div class="popup\_dialog\_v2\_foot">

    <div class="popup\_dialog\_v2\_foot\_button\_light"

style="margin-right:24px">

     <div class="popup\_dialog\_v2\_foot\_content">

      添&nbsp;加

     </div>

    </div>

    <div class="popup\_dialog\_v2\_foot\_button\_dark">

     <div class="popup\_dialog\_v2\_foot\_content">

      取&nbsp;消

     </div>

    </div>

   </div>

  </div>

</body>

</html>

</html>

**日历组件**

请大家注意，在页面多开时，日历组件的containerid必须不同。

新的日历组件调用方法，可参考下面里例子，基本和以前是一样的（绿色部分请注意）：

 var cal  = new DatePickerVersionTwo({

      containerId: "develop\_calendar",

      editable: true,

      dateInputStyles: {

        borderColor: "#c6c6c6",

        width: "150px"

      },

      calendarStyles: {

        width: 428,

        navTitleBgColor: '#ffffff',

        datesViewHeight: 160,

        datesViewGridColor: '#ffffff',

        datesViewCellColor: '#ffffff',

        weekdaysHeight: 22,

        weekdaysColor: '#a1a1a1',

        currMonthColor: 'rgb(121,121,121)',

        nonCurrMonthColor: 'rgb(200,200,200)',

        align: 'left'

      }

    });

                   this.cal=cal;

                   this.cal.onChange = function () {

};

**styleableList.js**

添加了一个新组件 styleableList.js, 用法基本和 styledList 相同，一些区别和特点：

1. js中的样式多数都去掉了，方便在css中样式覆写

2. label 不再只是字符串，支持function和其他对象，方便一些新需求

3. styles中仅剩的columnsWidth会转变成百分比赋值，方便自适应宽度

4. 表格下翻页等用iot的新样式

鉴于“导出”功能可能涉及的aqua内容，我们在基础架构中新增了一些aqua相关的

内容，不过还没实装，因为还没确认iot里使用aqua,

还加了paas的公共属性，详细的内容参见svn 12728的修改。

也希望这里的东西能给其他东西的引入提供一些参考。

1. my

    a. 添加了两个方法，涉及aqua contentType的

    b. 支持requirejs define等模块引用

2. tifAqua

    a. 这个是把teacherpanel tif\_aqua的内容搬进来的

    b. 它使用define定义的

    c. 这里搬的是最初引进到teacherpanel的版本，svn 4047

    d. 去掉了dojo相关的引用

    e. 使用前要先调用setUp将my.aqua的host, restRoot, accessKeyId, secretAccessKey赋值进去，

        备注一下，不管哪里require的tifAqua, 其实指向同一个对象，虽然多个使用setUp的会相互覆盖，

        不过从目前来看，主要是将my.aqua的值赋值进去，暂时不会出现交叉污染问题

3. fileUpload

    a. define定义的

    b. 使用tifAqua了,

    c. 具体使用基本不变

4. aquaUtil

    a. 它是访问aqua的一个简单封装，提供一些简单访问，功能上应该没有tifAqua原始

    b. define定义的

5. 在develTest中试了一下，看起来各个模块之间的调用还走的通，仅供参考

**Tab 切换方法**

svn提交了tab切换相关的方法，

第一个入参都是menu的id, 支持第二个入参的可由接收方自定义

App.newTab,

支持第二个入参，如果原来菜单已经打开了，会关掉开新的，这个应该是现在多数需要传参的情况下使用的方法

App.showTab,

没有第二个入参，如果原来菜单已经打开了，就切换过去，没有才开新的

App.addTab

支持第二个入参，不过这个是为了支持多开加的，现在暂时应该还用不到

第二个参数会传到App.navToxxxx去作为containerId后的参数，

模块里可以自定义处理方式，也可以参考DevTest里的传递方式