# 超远前端框架API

### 框架说明

#### 前端页面系统架构

全路径中，除了index.html页面以外，其他的页面都是html片段，系统前端属于单页面系统架构。

**弹框页面2**

**弹框页面3**

**弹框页面1**

**模块页面4**

**模块页面3**

**模块页面2**

**模块页面1**

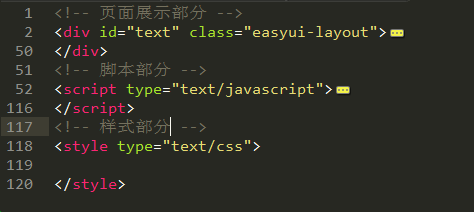
**main.html(首页)**

**login.html(登录)页**)

**index.html**

所以在开发过程中，我们主要的开发任务是完成模块页面和弹框页面的设计和代码编辑。

#### HTML页面(模板)架构



这个是各个页面的设计架构，页面展示部分要以easyui-layout组件开始，并给与一个唯一ID，一般用文件名即可。

页面工作原理：

1. 通过easyui的panel组件的href加载机制，加载这个代码片段后，开始解析DOM部分，并执行javascript脚本，加载样式。
2. 所有的加载的DOM，javascript和css都是应用在全局。
3. 多个TAB页签打开的页面可能会有以下冲突：
   * 1. DOM的ID冲突

如果同时打开的两个页面有相同的ID，通过$(“#id”)或者document.getElementById(“id”)这样的方式获取DOM元素，可能获取到的就是其他页面的元素。

* + 1. CSS冲突  
       页面中定义的CSS会影响其他页面相同的元素。
    2. javascript冲突

两个同时打开的页面，如果都定义的相同的全局变量，则后打开的页面中的变量会覆盖先打开的页面的变量。再返回先前打开的页面进行操作就会有问题。

#### 框架解决页面冲突的原理

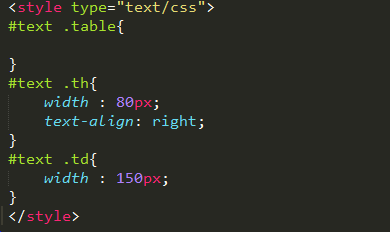
1. 样式冲突

首先，要给页面一个唯一的ID，一般定义在最外层的DOM元素中，也就是easyui的layout组件：



这个ID一定要唯一，一般设置成文件名即可，这个ID的用处就是用来定义样式的。

在各个页面中的style标签里，通过这个ID来给页面添加个性化样式，如下：



通过这样的编辑样式的方式就可以解决样式冲突问题。

1. DOM冲突和javascript冲突

首先，在脚本区域，禁止定义全局变量，禁止使用$([jQuery选择器])或者document.getElementById(id)的方式来获取页面中的DOM元素。

在脚本区域中，统一调用InitPage方法来初始化页面和定义页面的作用域：



这个方法有两个参数，第一个参数是页面需要注入的参数，后面会详细说明，第二个参数是一个函数，这个函数的作用域就是本页面脚本作用域，所有的代码都只能写在这个函数中，这样就避免的Javascirpt的冲突，因为每个页面都有一个作用域，各个页面在自己页面定义的变量和方法都互相不干扰。

这个作用域函数还有一个参数：$scope，这个是该作用域与DOM交互的核心对象，只有通过$scope中的特定操作才可以与DOM进行数据绑定，事件交互，DOM元素处理等。后面会具体说明。

$scope中有一个$()方法，可以通过$scope.$([jQuery选择器])方法来获得本页面的DOM元素，完全可以替代原来的$([jQuery选择器])方法，而且只会获取到本页面那种的元素，这样就解决了DOM冲突问题。

### 核心方法InitPage

#### InitPage方法说明



在script标签中直接调用InitPage方法：

参数1：需要注入的参数数组

参数2：初始化调用的方法

再次说明：禁止在InitPage()方法以外定义全局变量，所有的初始化方法和函数定义都必须卸载第二个参数的函数中。

#### 多页面间的参数传递

我们在开发过程中会进行各种页面跳转，打开页面等操作，页面跳转一般采用的是easyui的panel组件的href方式（tab页签，弹窗，panel的refresh操作都属于），属于异步加载，所以要实现各个页面的作用域数据交互，我们就必须使用$scope.$setParam()的方式来设置参数，在InitPage方法的第一个参数来自动完成参数的入住（也可手动调用$scope.$getParam()方法注入）

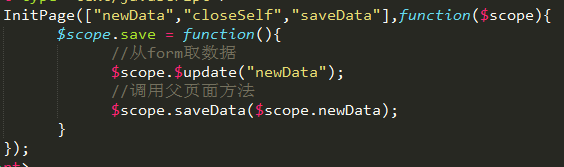
使用最多的场景如下：

页面1中我们需要打开一个编辑页面2，并给页面2传递需要编辑的数据，还有一些工具方法，比如关闭窗口，保存操作等。

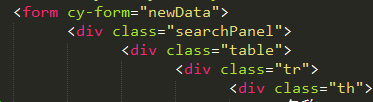


在打开页面之前，我会将这些需要的参数设置好，参数可以是数据，可以是字符串，可以是一个方法，甚至可以将本页面的 $scope对象传递进来。

在页面2中，通过InitPage的第一个参数数组来接受这些参数：



这样就完成了在页面2中自动注入父页面传递来的参数，可以直接使用了，甚至可以直接与DOM进行绑定：



具体的数据绑定原理后面会有说明。

还有一中方式是使用url中的参数来实现简单参数的传递：



打开这个页面的时候，将参数加到url后面，在页面中获取参数方式如下：



这种方式的传递参数只可以是打开一个新的tab页签或者使用$scope.$openDialog方法来传递URL中的参数，其他方式不支持。

#### 数据和DOM加载流程

1. 注射参数和工具变量方法。

2. easyui框架解析组件

3. 自定义事件初始化

4. 解析自定义组件

5. 全部组件加载完毕后，调用$scope.load()方法，如果没有定义load方法则不执行。

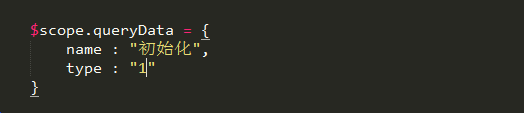
#### 组件使用说明

##### cy-form

用于$scope中的变量与from标签进行数据绑定，使用方法如下：



图中，cy-from属性值为queryData，表示该from下的所有输入框会自动绑定$scope. queryData的属性值，输入框数据有变化也同样会修改$scope. queryData的值。如果首先初始化了$scope.queryData变量，会初始化注入变量中的值到form中，如下：

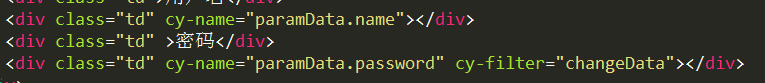


在页面打开时，效果如下：



##### cy-name和cy-filter

使用方法如下：

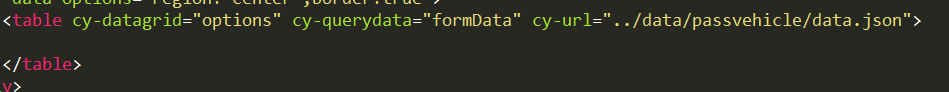


有上图可以看出，这两个组件的使用场景是展示页面，页面中没有可以用来绑定数据的from，只有<div><span>等元素，如果需要初始化时注入一些数据，就需要对每一个展示的容器元素定义一个cy-name属性，属性值对应的是$scope中的对象属性。

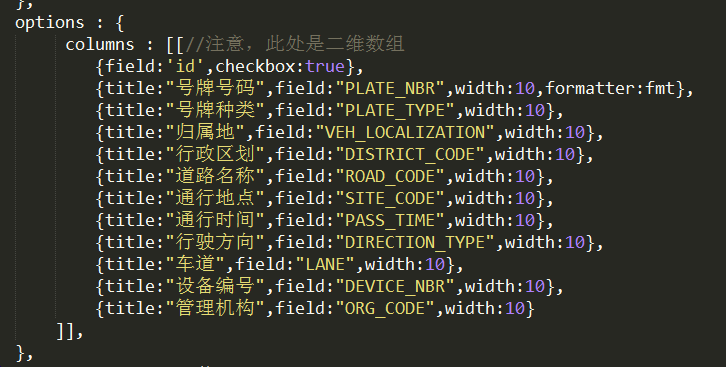
如果$scope中对应的值是code，或是其他需要转化的数据，还需要增加一个cy-filter属性，属性中是对用$scope中对应的转化方法。

##### cy-datagrid，cy-querydata和cy-url

使用方法入下：



上图中，会按照cy-datagrid属性值找到对应的options对象，创建dataguid，例如：



定义好列属性后，就可以在页面中看到列表的效果。

cy-querydata中的属性值与$scope中对应的属性值绑定，用作查询数据时的参数，cy-url是查询调用的http链接。

##### 事件绑定

针对页面上的按钮，如果使用原来的onclick，onmouseover等是无法调用到页面脚本作用域中的对象的，这时候就需要我们使用自定义的事件代理，cy-click,cy-mouseover等，使用方式如下：



如上图，cy-click的属性值对用的就是$scope对象中相对应的方法名



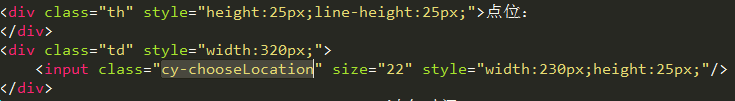
在定义的方法中，方法会默认有个参数：event，这个是事件参数，可以通过这个参数来获得与事件相关的属性，具体参考Jquery文档事件对象部分。

##### 组件扩展

目前可以使用的组件有：

###### chooseLocaion选择点位

使用方式：



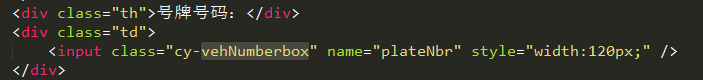
展示效果：



点击选择按钮，自动调用选择按钮页面。

###### vehNumberbox车牌选择框

使用方式：



展示效果：



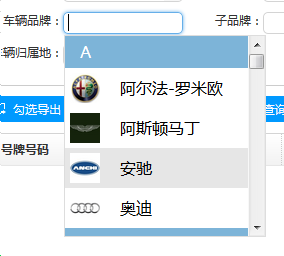
点击输入框后面的按钮会自动弹出编辑车牌窗口。

###### brandbox车辆品牌选择框

使用方式：



展示效果：



同时增加一个$scope方法$getBrandNameByCode()可以根据品牌编码获得品牌名称

###### 图片放大镜组件（公共页面）

图片放大镜组件是由一个公共页面实现的，用相应的容器加载公共页面，并传递图片参数就可以实现。

公共页面相对路径：tpls/public/image-view.html

必要参数：images 图片路径数组

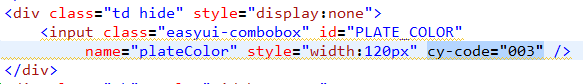
如下，panel加载方式：



使用弹框方式也可以，使用$scope.$openDialog()打开这个页面就可以。

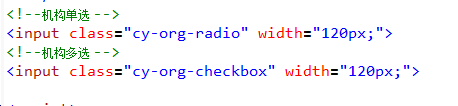
###### 系统代码绑定

绑定数据使用的是cy-code属性，给combobox组件增加一个cy-code属性，属性值为codeType，框架会自动从数据库中读取相应的系统代码数据并添加到combobox组件中，例：

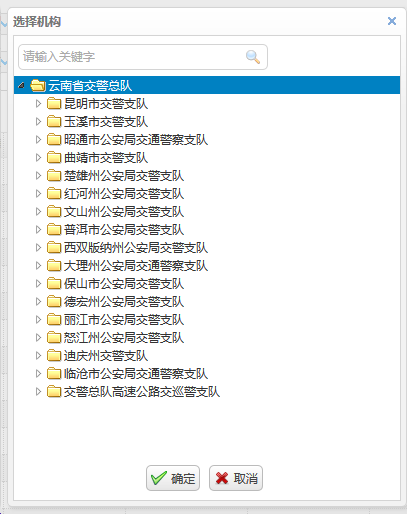


###### 机构选择组件

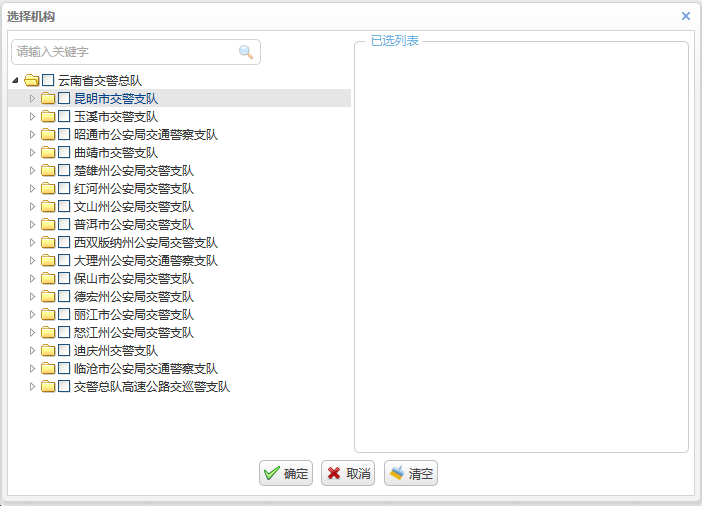
通过class为cy-org-raido加载单选机构组件，class为cy-org-checkbox加载多选机构组件，例：



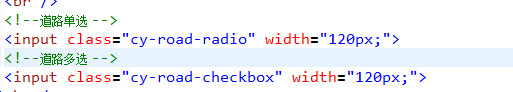
单选界面：



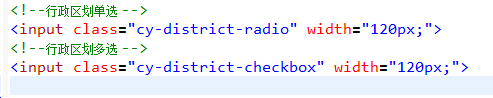
多选界面：



###### 道路选择组件



###### 行政区划选择组件



界面风格与机构相同

### $scope对象API

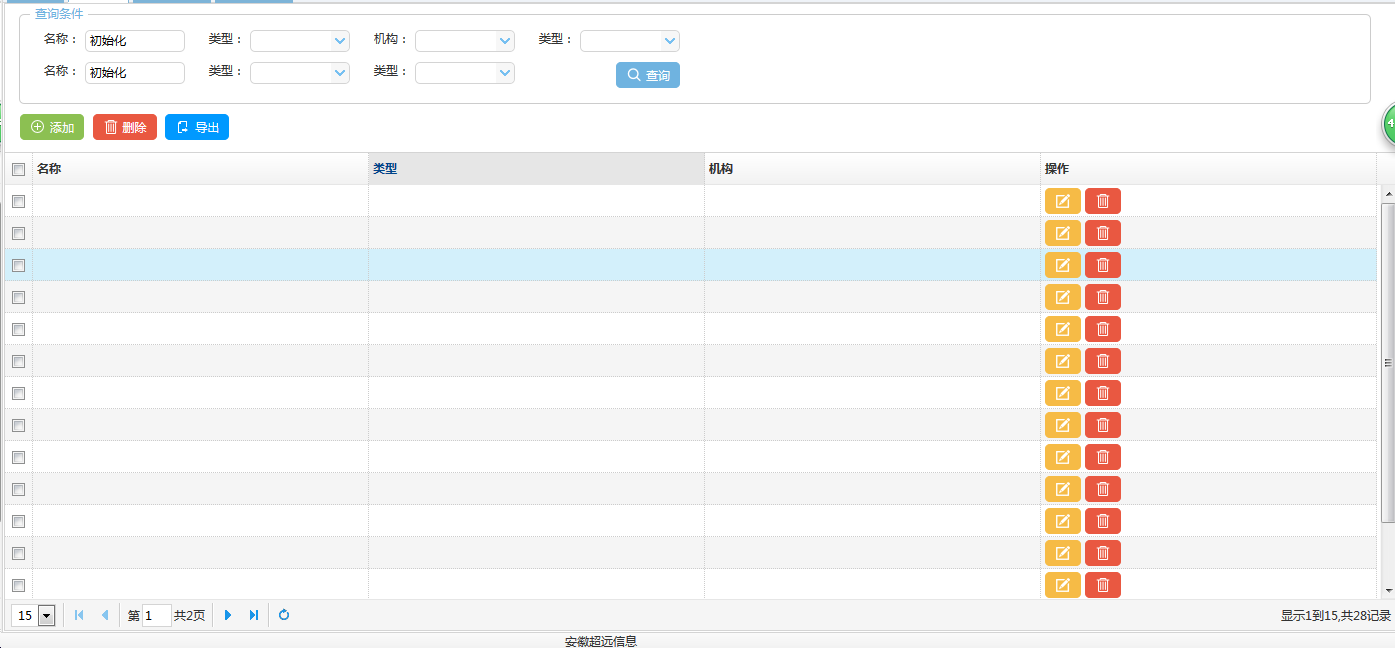
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 参数 | 描述 | 返回值 | |
| ContextId |  | 登录时保存的ContextId |  |
| UserInfo |  | 用户对象 |  |
| $ | selecter:jQuery选择器 | 在本作用域中获取DOM元素 | Jquery对象 |
| $search | queryData:绑定的数据名  params：额外的参数  callback：回调函数 | 根据配置的数据查询数据，并自动显示在datagrid中，只有给datagrid配置了cy-querydata属性和cy-url属性才可以使用这个方法 |  |
| $setData | key:键  value：值 | 缓存全局数据，使用key：value的形式 |  | |
| $getData | key : 键 | 获得数据 | 数据 | |
| $excute | fn:方法  param1：参数1  param2：参数2  param3：参数3 | 执行一个方法，并传递参数，最多3个参数，过多的参数请合并传递，3个参数以上请封装传递 | 方法的返回值 | |
| $setParam | key:键  value：值 | 设置参数 |  | |
| $getParam | key : 键 | 手动获得参数，获得参数后，参数删除 | 参数值 | |
| $openDialog | name：名称  options：窗口参数  isNew：是否始终打开新窗口 | 打开窗口并按照名称缓存起来，可以配置是打开新窗口还是打开先前关闭同名的窗口 |  | |
| $getDialog | name：名称 | 根据名称获得窗口对象 | 窗口对象 | |
| $updateData | name:数据绑定对象名称，参考cy-form属性 | 更新from数据到cy-form绑定的对象中 |  | |
| $updateDom | name:数据绑定对象名称，参考cy-form属性或者cy-name属性 | 更新数据到cy-from和cy-name绑定的DOM对象中，与初始化操作相同，可看做手动更新DOM方法 |  | |
| $ajaxRequest | obj：ajax参数 | ajax提交 |  | |
| $formSubmit | obj：form提交参数 | form提交 |  | |
| $eventProxy | 无 | 重置本页面所有的cy事件，一般用于非初始化状态新增一个按钮，按钮增加cy-click事件，调用$scope. $eventProxy()方法之后，cy-click事件生效 |  | |
| $closeSelf | 无 | 如果本页面是弹框页面，会给$scope对象自动注入这个方法，可以通过调用$scope.$closeSelf关闭本窗口，与窗口的右上角”x”按钮功能一样 |  | |
| $destroySelf | 无 | 如果本页面是弹框页面，会给$scope自动注入这个方法，可以用来销毁窗口对象，与$closeSelf的区别就是$closeSelf只会关闭页面，不会将页面中的DOM元素销毁，而本方法会直接销毁有关窗口的所有DOM |  | |
| $params.get | 参数名称 | 获得url中的参数 |  | |
| $getCodeName | type：系统代码类型编码，  id：系统代码id | 根据代码类型和代码获得名称 |  | |
| $getDefaultCode | codeTypes:系统代码数组 | 提前加载好页面需要用到的数据字典 |  | |
| $getOrgName | code：机构代码 | 根据机构代码获得机构名称，代码可以是多个以逗号分隔的形式 |  | |
| $getRoadName | code ：道路代码 | 根据道路代码获得道路名称，代码可以是多个以逗号分隔的形式 |  | |
| $getDistrictName | code: 行政区划代码 | 根据行政区划代码获得行政区划名称，代码可以是多个以逗号分隔的形式 |  | |

页面事件，当定义以下方法是，页面有相应的动作时会自动调用，仅限于使用tab页签打开的页面，弹窗或者其他方式打开的页面只有load()方法有效

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法名 | 参数 | 描述 | 返回值 | |
| load() | 无 | 页面所有内容加载并解析完毕时调用 | 无 |
| close() | 无 | 页面关闭时调用 | 无 |
| leave() | 无 | 离开页面切换到别的页签时调用 | 无 |
| enter() | 无 | 重新进入页面所在页签时 | 无 |

### 页面统一布局

#### 整体页面风格



页面分为查询条件区，按钮区，和数据展示区。

1. 查询条件区在外部增加一个框，左上角有个查询条件标题。

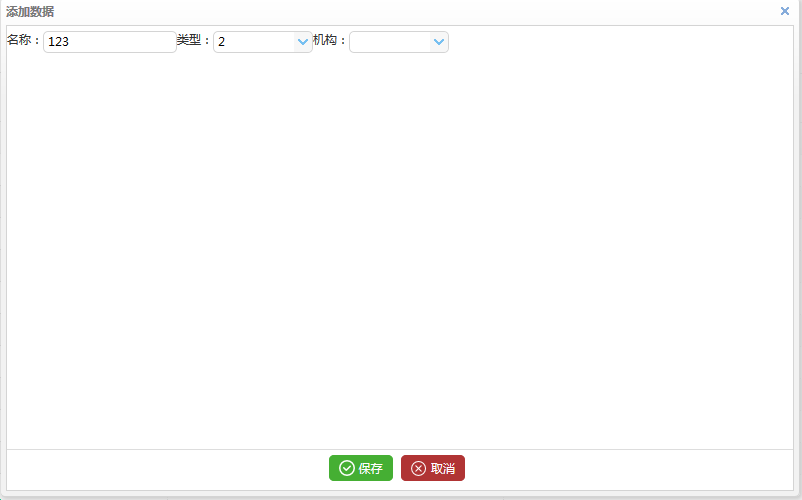
2. 查询按钮在查询条件的最后一行末尾。

3. 按钮区的按钮按照“新增”，“删除”和其他按的顺序排列。

4. 列表中，操作栏中按照“编辑”，“删除”和其他按钮的顺序排列

5. 双击列表中的一行是查看详细信息。

弹框中的页面风格



1. 按钮统一放在弹框下边的中央。

2. 按钮按照“保存”“取消”和其他按钮的顺序配列。

3. 内容展示区如果要出现滚动条，不能影响按钮的位置。

4. 弹框的宽度不能超过1000px，高度不能超过550px。

#### 按钮

一般的按钮采用边框+图标+文字来实现，列表后面的按钮不需要文字，需要给按钮加一个title即可。特殊的按钮或者未定义按钮可以不加图标，只使用边框+文字的方式来实现。

按钮采用easyui的按钮样式，并结合自己的按钮样式。定义一个按钮方式如下



其中 easyui-linkbutton和data-options是easyui的按钮定义方式，我们系统在此基础上增加了一个button-\*样式。无法找到对应的单独样式，使用button也可以。

如果用js创建一个按钮，则通过下面的方式，以添加按钮为例：

var button = $(“<a class=’button-add’>添加</a>”)；

button.linkbutton({

iconCls:”icon-add”

})

$(按钮容器).append(button)；

通过跟换这三个属性来更改按钮的样式，目前定义好的样式有(持续更新中)：

##### 添加

class :button-add

icon : icon-add

效果图



##### 编辑

class :button-edit

icon : icon-edit

效果图



##### 删除

class :button-remove

icon : icon-remove

效果图



##### 查询

class :button-search

icon : icon-search

效果图



##### 保存

class :button-save

icon : icon-save

效果图



##### 关闭/取消

class :button-close

icon : icon-close

效果图



##### 清空

class :button

icon : icon-clear

效果图



##### 导入/导出

class :button

icon : icon-export,icon-inport

效果图



##### 启用，禁用，报废

class :button-enable，button-disable，button-drop

icon : icon-enable,icon-disable,icon-drop

效果图



##### 上一个，下一个

class :button-prev，button-next

icon : icon- prev,icon- next

效果图



### 前端编码规范

#### CSS，DOM，JavaScript编码规范

前段编码规范要按照一般通用编码的方式，确保页面整体美观，整洁。

<http://www.css88.com/archives/5361>

这个网址中的前端代码规范比较全面。

#### 其他框架规范

可以参考tpls/passVehicle/vehicleManager文件夹下的3个页面文件，这三个页面文件分别是列表页面，添加/修改页面和查看页面

##### 核心规范

1. 不允许定义全局的javascript变量，方法，保证整个javascript编码环境不受污染。

2. 同样不允许定义全局的CSS样式，只能根据页面最外层的唯一ID去修改本页面的元素样式。

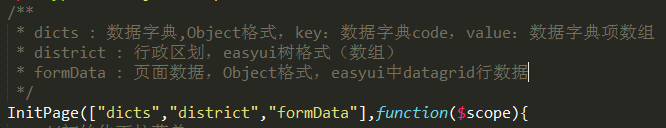
3. 在Javascript代码中，不允许使用$([selecter])或者document.getElementById([ID])等全局查找方式来查找DOM元素，否则可能会出现意想不到的问题，必须使用$scope.$([selected])的方式。

##### 注释

我们编写的所有代码，要考虑可读性，读者可能是新员工，其他同事，或者未来的自己，所以可读性高的代码可以降低大量的沟通成本。提高可读性的最有效方法之一，就是增加有效的注释。

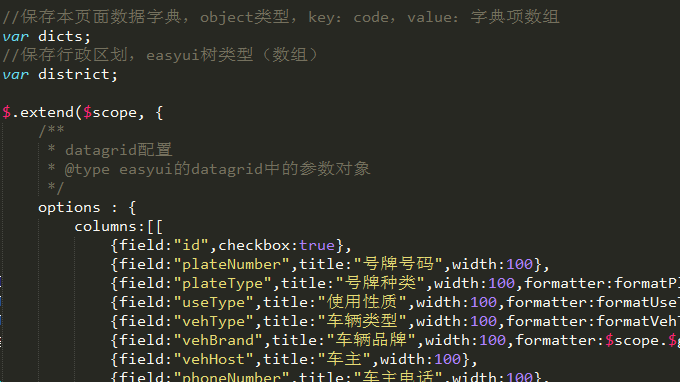
###### InitPage注释

InitPage是页面的核心方法，所以在InitPage之上要详细说明其他页面传来的参数的用途，格式，例如：



###### 变量注释

在页面中定义的变量，必须增加注释，包括说明和类型：



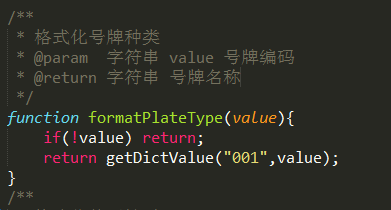
###### 方法注释

$scope中注入的方法和自定义方法都需要添加注释，注释中要说明：

1. 参数说明
2. 参数类型
3. 返回值类型（没有返回值不写）

参考格式如下，其他格式只要符合要求即可：





###### 关键位置注释

在代码业务逻辑处理的关机位置，如if，for，switch，ajax等，需要增加相应的注释说明，例如：



