WEB UIFrame 开发框架基本约定

# 开发框架结构

## 框架结构图



## 框架结构说明

### 框架层

页面端主要采用了jQuery、SeaJS

JS控制层主要是Arale和Backbone实现MVC模式

请求的转发控制层则采用了SpringMVC框架。

### UI层

**HTML文件**：加载所需要的组件JS,包括框架JS、配置文件、组件的组装，通过标签data-widget进行加载相应的组件和模板文件。

**组件JS**：是组件的控制器，提供加载基础模块JS、加载样式文件、加载模板文件、组件功能继承和扩展、事件绑定、服务交互、回调处理、加载数据等功能。

**组件ＴＰＬ文件**：组件的模板文件，通过特定语法标签加载变量值。然后展现到前台页面

**组件CSS文件**：每个组件的CSS 文件独立，渲染时可根据主题进行渲染。

**SessionFilter(会话拦截器)**：提供是否登录的检查、会话维持等功能，页面请求链接时触

发。

**NcServerRequestFilter(请求拦截器)**：对页面请求进行拦截，处理是否存在Token、是否登录等检查功能。

**Controller:请求处理类**：提供处理从页面传来的请求和数据、回传页面跳转和数据等服务。

**Service(资源服务类)**:负责处理业务逻辑、调用后台服务。

**XXXExceptionResolver：**异常处理程序，将错误返回到页面上。

**XXXExceptionInterceptor：**请求异常拦截器，将异常显示在页面上。

**NCLocatorFactory:**由于NC的特殊安全机制，所以需要进行服务配置，然后调用EJB服务接口。

**XXLoadConfigProperty(配置缓存)：**读取配置文件，建立缓存

**BeanUtils或VO转换器**：NC有专门的类型ＶＯ类，所以要将其转换为普通类型的ＶＯ类

## 公共技术支撑

jQuery：非常轻量简洁的页面元素查找、异步请求框架。

Seajs:模块化编写框架，一个文件即一个模块，可以自由选择暴露的调用方法，可以自由加载所需的模块文件，而不用一次性全部加载。

Arale:基础框架，提供基础的工具类库

Backbone: 为复杂Javascript应用程序提供模型(models)、集合(collections)、视图(views)的结构。其中模型用于绑定键值数据和自定义事件；集合附有可枚举函数的丰富API； 视图可以声明事件处理函数，并通过RESRful JSON接口连接到应用程序。

html5：提高可用性和改进用户的友好体验。

css3:某些特性IE8并不支持，IE9支持

# 命名规则

## JSP

1. jsp文件名全部小写
2. 数据/内容显示页,  名词形式，多个单词用下划线分隔，要求能说明显示内容的信息，为避免冲突，可加上“\_list”或者其他的单词。例如： file\_list.jsp
3. 操作处理页: 名词\_下划线\_动词，例如：file\_delete.jsp
4. 名称最好从范围大到小命名（或者是模块名\_子模块名\_操作的方式），比如service\_order\_delete.jsp，当jsp页面很多的时 候，容易检索和识别。

## JS

1. 变量名称必须为小写字母。
2. 类的命名使用骆驼命名规则，例如：Account, EventHandler
3. 常量 必须在对象（类）或者枚举变量的前部声明。枚举变量的命名必须要有实际的意义，并且其成员必须使用骆驼命名规则或使用大写：

var NodeTypes = {

Element :1,

DOCUMENT:2

}

1. 简写单词不能使用大写名称作为变量名：getInnerHtml(), getXml(), XmlDocument
2. 方法的命令必须为动词或者是动词短语：obj.getSomeValue()
3. 公有类的命名必须使用混合名称（mixedCase）命名。
4. CSS 变量的命名必须使用其对应的相同的公共类变量。
5. 私有类的变量属性成员必须使用混合名称（mixedCase）命名，并前面下下划线（\_）。

var MyClass =function(){

var \_buffer;

this.doSomething=function(){

};

}

1. 变量如果设置为私有，则前面必须添加下划线。例如：

this.\_somePrivateVariable = statement;

1. 通用的变量 须使用与其名字一致的类型名称：

setTopic(topic)// 变量 topic 为 Topic 类型的变量

1. 所有的变量名 必须 使用英文名称。
2. 变量如有较广的作用域（large scope），必须使用全局变量；此时可以设计成一个类的成员。相对的如作用域较小或为私有变量则使用简洁的单词命名。
3. 如果变量有其隐含的返回值，则避免使用其相似的方法：

getHandler();// 避免使用 getEventHandler()

1. 公有变量必须清楚的表达其自身的属性，避免字义含糊不清，例如：

MouseEventHandler，而非 MseEvtHdlr。

1. 类/构造函数 可以使用 扩展其基类的名称命名，这样可以正确、迅速的找到其基类的名称：

## CSS

1. id和class命名采用该版块的英文单词或组合命名，并第一个单词小写，第二个单词开始首字母大写，如:newRelease（最新产品/new+Release)
2. CSS样式表各区块用注释说明
3. 尽量使用英文命名原则
4. 不用加横杠和下划线
5. 尽量不缩写，除非一看就明白的单词

## JAVA

1. 包的命名:全部小写，由域名定义, 例如：net.frontfree.javagroup
2. 类的命名:单词首字母大写, 例如： Graphics
3. 方法的命名:首字母小写，字母开头大写, 例如：drawImage
4. 常量的命名:全部大写，常加下划线, 例如：MAX\_VALUE
5. 参数的命名:参数的命名规范和方法的命名规范相同，而且为了避免阅读程序时造成迷惑，请在尽量保证参数名称为一个单词的情况下使参数的命名尽可能明确。
6. 变量的命名:

主要的的命名规范有以下三种：

* Camel 标记法:首字母是小写的,接下来的单词都以大写字母开头
* Pascal 标记法:首字母是大写的,接下来的单词都以大写字母开头
* 匈牙利标记法:在以Pascal标记法的变量前附加小写序列说明该变量的类型

在Java我们一般使用匈牙利标记法，基本结构为scope\_typeVariableName，它使用1-3字符前缀来表示数据类型，3个字符的前缀必须小写，前缀后面是由表意性强的一个单词或多个单词组成的名字，而且每个单词的首写字母大写，其它字 母小写，这样保证了对变量名能够进行正确的断句。例如，定义一个整形变量，用来记录文档数量：intDocCount，其中int表明数据类型，后面为表意的英文名，每个单词首字母大写。这样，在一个变量名就可以反映出变量类型和变量所存储的值的意义两方面内容，这使得代码语句可读性强、更加容易理解。 对于在多个函数内都要使用的全局变量，在前面再增加“g\_”。例如一个全局的字符串变量：g\_strUserInfo。

## TPL

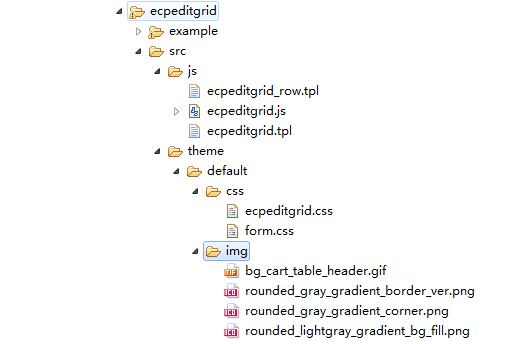
Handlebars是JavaScript一个语义模板库，通过对视图（view）和数据（data）的分离来快速构建Web模板。它采用"Logic-less template"（无逻辑模版）的思路，在加载时被预编译，而不是到了客户端执行到代码时再去编译，这样可以保证模板加载和运行的速度。Handlebars兼容Mustache，你可以在Handlebars中导入Mustache模板。

详细用法参见<http://www.cnblogs.com/iyangyuan/archive/2013/12/12/3471227.html>

# 组件开发过程

## 工程代码结构

一个完整的组件基本上包括:js,tpl模板，css，图片四部分。组织结构如下图所示：



组件之间可以相互依赖，用面向对象的角度来考虑每个组件都可以抽象成一个类，通过seajs语法来定义组件，在组件中可以定义变量，定义方法，定义提供外界访问的接口，还可以在组件中引入其它组件并可以调用引入组件中定义的属性和方法，其中组件包括基础组件和业务组件，基础组件是所有业务组件构建的基础基础组件包括：widget，handlerbar，class，event，base

## 组件开发过程

下面以最基本的表格组件为例详细介绍一下组件的具体开发工程

首先按照3.1介绍定义好一个组件的基本结构，定义完成之后对seajs进行配置，主要是配置一下基准路径和js文件对应的别名。见下代码。关于seajs开发组件的基本约定详细介绍参照seajs开发规范，这里不做详细介绍。

**var** path="/ruleValidator/component";

seajs.config({

alias : {

'class' : path+'/webpub/core/class/src/class',

'events' : path+'/webpub/core/event/src/events',

'editable' : path+'/webpub/core/editable/src/editable',

'base' : path+'/webpub/core/base/src/base',

'widget' : path+'/webpub/core/widget/src/widget',// 组件核心

在完成前面的操作之后现在开始模块部分的开发代码如下。原代码参见Grid.js

/\*\*

\* Grid 组件

\*

\*/

define(**function**(require, exports, module) {

**var** $ = require("$");

require("../theme/default/css/grid.css")

**var** Widget = require("widget");

**var** grid\_tpl = require("./grid.tpl");

**var** handlebars = require("handlebars");

**var** GridFormat = require("./GridFormat");

如述代码引入了一个基本的表格组件需要依赖基础组件，模板和css样式：

Jquery：开源的js库，封装了大量的api简化用原生的js开发带来的难度，主要用来操纵dom，封装了ajax便于与后台进行数据交互。

grid.css：组件用到的css文件。

widget：Widget 是 UI 组件的基础类，约定了组件的基本生命周期，实现了一些通用功能。基于 Widget 可以构建出任何你想要的 Web 界面组件。

grid.tpl：定义的模板可以理解为html，可用通过handlebars往模板中插入相应的值。模板的定义规范参照handlebar.js

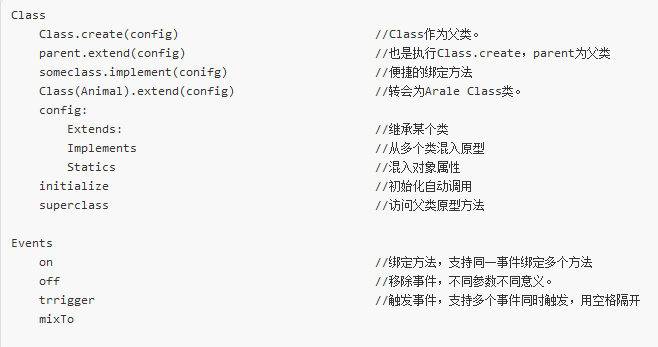
handlebars：handlebars 使用了模版，只要你定义一个模版，提供一个json对象，handlebars 就能把json对象放到你定的模版中，也就是往grid.tpl模板中插入相应的值。

gridFormat：grid列扩展组件主要用来格式化日期。

其中handlebars，widget，jquery属于基础组件。

下面详细介绍一下widget.js:widget.js是组件开发的核心组件任何组件都直接或者间接的调用此组件进行绘制页面，该组件引用了handlebars组件实现对组件模板的加载数据的装配，该组件主要包括初始化公共参数，初始化特有参数，加载数据，把组装好的元素渲染到页面中，绑定事件监听。其中部分方法已经实现，部分方法只是声明定义在子类中具体实现。Widget组件还引用了base组件，下面具体介绍一下base组件：

Base组件：相当于object提供了get和set方法该组件引用了class组件和event组件。



如上图所示：

Class组件：提供create 方法创建的类，并提供 extend 方法，可以继续创建子类。

Event组件：提供了添加事件，移除事件，触发事件的方法。

接下来我们利用基础组件来构建我们的组件，如下代码所示：

Gride.tpl:

<table class="infolist ecp\_grid\_table" cellspacing="0" cellpadding="0" >

<thead>

<tr class="row\_bg">

{{#if attrs.ismulti}}

<td style="width:25px;"><input type="checkbox" class="J\_grid\_selected\_all"/>选择</td>

{{/if}}

{{#if attrs.showRowNum}}

<td>序号</td>

{{/if}}

{{#if attrs.issingle}}

<td>选择</td>

{{/if}}

{{#each attrs.columns}}

<td style="text-align:{{this.align}};">{{this.label}}</td>

{{/each}}

</tr>

</thead>

<tbody>

{{#grid this}}

{{this}}

{{/grid}}

</tbody>

</table>

这个是tpl模板可以理解成html，通过handlerbar.js组件可以往其中的{{}}赋值，tpl文件和js文件息息相关，详细语法请参照handlebars.js

/\*\*

\* 扩展handlebars注册一个名字叫grid的方法

\*/

handlebars.registerHelper("grid", **function**(context, options) {

**var** ret = "";

**if** (context.model) {

**var** records = context.model.records;

// 数据选择方式

**var** ismulti = context.attrs.ismulti;

**var** issingle = context.attrs.issingle;

**var** showRowNum = context.attrs.showRowNum;

**var** columns = context.attrs.columns;

**if** (records) {

**for** ( **var** i = 0; i < records.length; i++) {

// 设置隔行换色

**if** (i % 2 == 1) {

ret += "<tr class='row\_even'>";

} **else** {

ret += "<tr>"

}

**if** (ismulti || issingle) {

**var** input;

**if** (ismulti) {

input = '<input type="checkbox" name="grid"/>';

} **else** {

input = '<input type="radio" name="grid"/>';

}

ret += "<td style='width:25px;'>" + input + "</td>";

}

**if** (showRowNum) {

ret += "<td class='gridIndex'>" + (i + 1) + "</td>";

} **else** {

ret += "<td class='gridIndex hidden'>" + (i + 1) + "</td>";

}

**for** ( **var** j = 0; j < columns.length; j++) {

ret += "<td style='width:" + columns[j].width + "px;text-align:" + columns[j].align + "'>";

**var** data = records[i];

**var** colId = columns[j].key;

// 格式化列接口

// 直接在format配置中写function

**var** commonformat = columns[j].format;

// GridFormat配置中配置的方法

**var** gridformat = GridFormat[columns[j].format];

// 其他子类format中配置的方法

**var** subformat;

**if** (context.attrs.format) {

subformat = context.attrs.format[columns[j].format];

}

**if** (**typeof** (commonformat) == "function") {

ret += commonformat(data[colId], data);

} **else** **if** (**typeof** (gridformat) == "function") {

ret += gridformat(data[colId], data);

} **else** **if** (**typeof** (subformat) == "function") {

ret += subformat(data[colId], data);

} **else** **if** (colId != **null**) {

**var** value = data[colId];

**if** (value != **null**) {

ret += data[colId];

} **else** {

ret += "";

}

}

ret += "</td>";

}

ret += "</tr>";

}

}

}

**return** ret;

});

这段代码运用引入的handlebars组件，并调用了其中的Handlebars.registerHelper = **function**(name, fn, inverse)方法，拓展handlebars为其注册一个名字叫grid的方法主要是为了实现表格隔行换色的效果。以及单选框复选框的选择，是否显示行号，和格式化操作。

**var** Grid = Widget.extend({

template : grid\_tpl,

//实现了widget定义的custAttr方法初始化一下特有的参数

initCustAttr : **function**() {

**var** attrs = **this**.get("attrs");

**var** ismulti = attrs.ismulti;

**switch** (ismulti) {

**case** **true**:

attrs.ismulti = **true**;

attrs.issingle = **false**;

**break**;

**case** **false**:

attrs.issingle = **true**;

attrs.ismulti = **false**;

**break**;

**default**:

attrs.ismulti = **false**;

attrs.issingle = **false**;

}

},

//加载数据并调用widget的.setModel方法把数据交给widget进行组装

loadData : **function**(params) {

**var** grid = **this**;

//是否自动加载

**var** autoRender = **this**.get("autoRender");

**var** model = **this**.get("model");

**if** (model)

**return**;

**if** (autoRender) {

**var** url = **this**.get("url");

$.ajax({

type : 'GET',

url : url,

success : **function**(data) {

grid.setModel(data);

**if** (parent) {

parent.seajs.emit("event\_grid\_loaded");

} **else** {

seajs.emit("event\_grid\_loaded");

}

},

error : **function**() {

},

dataType : 'json',

data : params

});

}

},

handlebars : handlebars,

format : {},

//查询设置自动加载

query : **function**(params) {

**this**.set("autoRender", **true**);

**this**.loadData(params);

},

// checkbox绑定事件，全选，反选

bindEvent : **function**() {

**var** grid = **this**;

$(".J\_grid\_selected\_all").live("click", **function**() {

**var** checked = $(**this**).attr("checked");

**if** (!checked) {

checked = **false**;

}

// 为了防止多页签和grid合用的时候带来的相互干扰的问题，详情见渠道

$(**this**).parents(".ecp\_grid\_table").find("[name=grid]").each(**function**() {

$(**this**).attr("checked", checked);

});

});

**var** event = **this**.get("event");

**if** (event) {

**for** ( **var** i = 0; i < event.length; i++) {

$("." + event[i].key).live(event[i].trigger, event[i].handler);

}

}

},

getSelectedIndex : **function**() {

$(".ecp\_grid\_table").find("[name=grid]").each(**function**() {

$(**this**).attr("checked", checked);

})

},

/\*\*

\* 获取被选中的数据

\*/

getSelectedData : **function**() {

**var** data = [];

**var** that = **this**;

$("tbody input:checked").each(**function**() {

**var** index = $(**this**).parent().parent().find(".gridIndex").text();

data.push(that.get("model").records[index - 1]);

});

**return** data;

},

/\*\*

\* 获取选择的行数据

\*/

getSelectedRow : **function**() {

**var** data = [];

**var** that = **this**;

$("tbody input:checked").each(**function**() {

**var** row = $(**this**).parent().parent();

data.push(row);

});

**return** data;

}

});

这段代码主要是拓展widget组件来绘制表格，初始化特有参数，加载数据，绑定事件，并且添加了一些新的方法。

最后用module.exports = Grid;：提供外界访问的接口。

在定义完一个组件之后，接下来介绍一下组件的调用过程具体代码如下：

<div id=*"grid"* style="width: *500px*;"></div>

seajs.use( '/web/component/webpub/common/grid/src/js/Grid',

**function**(Grid) {

**var** grid = **new** Grid(

{

attrs : {

url : 'data.json',

columns : [ {

label : "商品名称",

key : "name"

}, {

label : "商品编码",

key : "code"

}, {

label : "商品价格",

key : "price"

} ]

},

renderTo : "grid"

});

});

我们通过seajs.use指定调用哪个组件，通过new的方式创建一个组件并且传入特定的参数，其中url是获取数据的一个地址，label对应的是tpl模板中的{{this.label}}，key对应的是获取到的json中的name， renderTo是把装载好的组件渲染到哪个元素下即id为grid的div。当加载这个html页面浏览器会显示我们需要的表格如图：



上述主要是描述了grid组件的开发过程。总结一下就是，定义好模块的代码结构，编写对应的css,js,tpl文件。其中重点是编写js文件具体步骤是引入基本组件，然后调用基本组件定义的方法构建我们的组件。

## UI组件构造

一个完整的组件基本上包括:js,tpl模板，css，图片四部分，一个组件可能会引入多个基本组件和业务组件，基本组件是组件构建的根本是必须直接或者间接引入的，业务组件需要根据实际情况引入。然后我们运用引入的组件来组建新的组件，包括定义属性，定义新的方法，拓展父类的方法等。并且提供对外访问的接口供其它组件调用。

## 组件组装

一个页面一般情况下都是若干个组件组成的，要引入这些组件首先需要引入sea.js以及对其配置的js文件，如下图所示：

<!DOCTYPE html >

<html>

<head>

<title>NC规则校验系统－系统管理</title>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<script type=*"text/javascript"*

src=*"/ruleValidator/component/webpub/core/seajs2/dist/sea.js"*

data-config=*"/ruleValidator/html/global.js"*

data-main=*"/ruleValidator/html/main.js"*></script>

</head>

下述代码介绍一下组件的组装过程

<body class=*"bg\_white"*>

<div data-widget=*"subnav"*></div>

<div data-page=*"addressManager"*>

<!-- 查询条件表单 -->

<div id=*"container\_form"* style="margin-top: *10px*;"></div>

<!-- 展现网格 -->

<div id=*"container\_grid"* style="float: *left*; width: *100%*"></div>

</div>

<script type=*"text/javascript"*>

seajs.use(['$', 'form', 'ecppagegrid','user'],

**function**($, Form, EcpPageGrid, Address) {

**var** form = **new** Form({

attrs : {

id : "form",

items : [{

label : "用户编码",

name : 'userCode'

},{

label : "用户名称",

name : "userName"

}],

buttons : [{

text : "查询",

id : "submit",

handler : **function**() {

grid.query($("#form"));

**return** **false**;

}

}, {

text : "添加",

type : "button",

id: "address",

handler: **function**(){

Address.addAddress(grid);

}

} ]

},

renderTo : 'container\_form'

});

**var** grid = **new** EcpPageGrid({

attrs : {

url : G.API.USER\_QUERY,

columns : [{

label : "用户编码",

key : "userCode",

width : "20%",

align : "left"

}, {

label : "用户名称",

key : "userName",

width : "20%",

align : "left"

},{

label : "手机",

key : "cellPhone",

width : "15%",

align : "left"

},{

label : "email",

key : "email",

width : "25%",

align : "left"

} ,{

label : "操作",

key : "caozuo",

width : "20%",

format:**function**(value, data) {

**return** '<a id="button\_address\_update" class="J-address-update ecpgrid\_btn">编辑</a> '

+'<a id="button\_address\_delete" class="J-address-delete ecpgrid\_btn">删除</a>';

},

align : "center"

} ]

},

renderTo : "container\_grid",

autoRender : **true**,

});

//修改用户

$('.J-address-update').live('click', **function**() {

**var** index=$(**this**).parent().parent().find(".gridIndex").text();

**var** record = grid.get("model").records[index - 1]

Address.updateAddress(record,grid);

});

//删除用户

$('.J-address-delete').live('click', **function**() {

**var** index=$(**this**).parent().parent().find(".gridIndex").text();

**var** record = grid.get("model").records[index - 1]

Address.deleteAddress(record,grid);

});

});

</script>

</body>

</html>

上述代码主要运用了三个组件包括面包屑导航组件，表单组件，表格组件，对于不需要配置就能自动渲染的组件比如面包屑导航我们通过<div data-widget=*""*></div>

方式来渲染，其中 data-widget中的值对应的是模块对应的别名。

对于需要配置的组件比如表格组件和表单组件，首先我们定义一个div作为组装后的组件的父元素并指定一下id然后我们通过seajs.user([,,],function(){

})方式来选择需要调用的组件并且在回调函数中运用我们选择的组件通过我们在3.2组件开发过程里介绍的方式来绘制页面。

## 服务调用

前端和后端的之间数据的交互多数都是通过ajax方式来实现的，代码如下

Js代码：

$.ajax({

type : "POST",

url : G.API.USER\_UPDATE,

dataType : "json",

data : $($(top.document).find("#addressEdit").get(0).contentWindow.document).find('#form-address-update').serialize(),

success : **function**(data) {

**if** (data!=**null**&&data.message!="excep") {

Notice.show($('#resource-grid'), '修改成功');

// [2] 调用grid的查询

grid.query('#form');

}**else** **if**(data.message=="excep"){

alert("相同的用户编码已存在！！！");

}**else** {

Notice.show($('#resource-grid'), '修改失败');

}

}

});

url映射关系：

USER\_UPDATE:G.ctx+"/account/updateUser.json",//修改用户

后台代码：

@RequestMapping(value = "updateUser.json", method = RequestMethod.*POST*)

**public** @ResponseBody

UserUIView updateUser(@ModelAttribute UserUIView userView,

HttpServletRequest request) **throws** PubBaseException {

**try** {

userView = accountService.updateUserView(userView);

} **catch** (PubBusinessException e) {

userView.setMessage("excep");

*logger*.error(e);

**throw** e;

}

**return** userView;

}

这个是一个典型的提交表单的方法，项目中采用springmvc框架(不做介绍)，url定义的是一个路径用来映射到后台代码中的一个方法，本项目中在api.js中配置了G.API.USER\_UPDATE具体指向的路径。详细介绍请参照前面的开发框架结构。

## 异常处理

本框架异常处理一种方式是通过

**if** (console) {

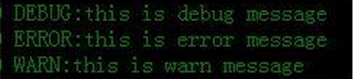
console.error("没有找到id为" + config.renderTo + "的元素，请检查组件renderTo对应的id是否存在！");

}

方式把出错信息输出到控制台，通过火狐的firebug就能看到错误信息

另一种方式是调用log组件以日志的方式输出错误信息

通过log.error()方式通过快捷键alt+F11显示错误信息。如图：



# 组件列表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | | 组件名称 | 组件说明 |
| 核心组件（WebContent\component\webpub\core） | | | |
| 1 | | base | 基组件，相当于Object，对外提供set, get，change 能力 |
| 2 | | class | 类组件，用于组件的扩展、构造与执行，使组件像类一样拥有create, implement, extend, Mutators能力 |
| 3 | | aspect | AOP组件，使组件拥有事件执行时的before，after处理能力 |
| 3 | | editable | 可编辑组件，使组件拥有editInit,doEdit,saveEdit,reset能力 |
| 4 | | Event | event组件，使组件拥有事件on,off,trigger,mixTo处理能力 |
| 5 | | handlebars | Jquery提供的模版引擎插件，通过此插件使组件拥有对组件模板（\*.tpl）的load,compile能力，具体参见  <http://www.cnblogs.com/enein/archive/2012/11/28/2793120.html> |
| 6 | | jquery | 用seajs封装过的jquery组件，相当于jquery |
| 7 | | seajs | 遵循[CommonJS](http://baike.baidu.com/view/9194056.htm)规范的JavaScript模块加载框架，提供config,use,define,require,exports等方法来使用组件，具体参见  <http://blog.codinglabs.org/articles/modularized-javascript-with-seajs.html> |
| 8 | | widget | 基UI组件，相当于swing jcomponent，所有UI组件继承此组件，拥有model,prop,implemnets,template,setTemplate,classPrefix,setModel,initialize,loadData,parseElement,render,beforeRender,afterRender,parseElementFromTemplate,setup,initAttr,\_parseDataAttrsConfig,initCustAttr,bindEvent,bindListener,change,compile,handlebars等能力 |
| 9 | | autoRender | 使UI组件具有自动渲染功能，渲染时查找所有具有data-widget标记的Div，然后获取data-widget标记值（已注册的组件名），再加载对应的组件。 |
| 公共UI组件（WebContent\component\webpub\common） | | | |
| 1 | adaptdialog | | 自适应弹出框组件 |
| 2 | bar | | 以列表方式展现的导航组件 |
| 3 | dialog | | 带确认取消按钮的弹出框组件 |
| 4 | Drop\_menu | | 下拉菜单组件 |
| 5 | echarts | | Google图表组件 |
| 6 | ecpgrid | | 电子销售封装的表格组件 |
| 7 | ecppagegrid | | 电子销售封装的带分页功能的表格组件 |
| 8 | editgrid | | 可编辑表格组件，含编辑删除，按钮 |
| 9 | extendgrid | | 可展开表格组件，支持明细行展开 |
| 10 | form | | 窗体组件，窗体元素包含下拉列表框，隐藏框，复选框，单选框，日期控件，时间控件，范围控件等 |
| 11 | grid | | 基本的表格组件 |
| 12 | imgresize | | 图片自适应组件 |
| 13 | list | | 无序列表组件 |
| 14 | loading | | 进度条组件 |
| 15 | log | | 日志组件，支持不同日志级别，并将日志输出到firebug控制台 |
| 16 | pricturedisplay | | 图片切换组件 |
| 17 | posdialog | | 可定位对话框组件 |
| 18 | quotegrid | | 报价表格组件 |
| 19 | slide | | 图片幻灯片放映组件 |
| 20 | Slide2 | | 滑动组件 |
| 21 | tabbar | | 页签条组件 |
| 22 | tabpanel | | 页签面板组件 |
| 工具类组件（WebContent\component\webpub\tools） | | | |
| 1 | backbone | | 由jquery提供的前台mvc框架，具体参见  <http://www.csser.com/tools/backbone/backbone.js.html> |
| 2 | bigdecimal | | 大数据运算工具 |
| 3 | ca | | ca 安全认证工具 |
| 4 | codemirror | | 代码高亮插件，具体用法详见  <http://www.open-open.com/lib/view/open1356778271870.html> |
| 5 | jsl | | json格式化工具类 |
| 6 | format | | 数字日期格式化工具类 |
| 7 | formatJson | | 格式化json工具类 |
| 8 | jasmine | | 单元测试框架，详见  <http://www.orzpoint.com/learning-jasmine/> |
| 9 | json | | Json组件，提供对json串的parse, stringify支持 |
| 10 | jsuri | | Javascript uri工具类 |
| 11 | math | | 计算函数组件，支持大数据增加（accAdd），删除（accMul）操作 |
| 12 | Math.uuid | | 唯一码生成器组件，支持生成uuid，uuidFast，uuidCompact操作 |
| 13 | moment | | Javascript日期处理工具类，具体详见  <http://momentjs.com/> |
| 14 | operationFlow | | 流操作组件，Generate a operation flow plugin |
| 15 | placeholder | | html5 placeholder 属性 在不支持placeholder的浏览器中通过js来模拟 |
| 16 | request | | 请求获取当前信息 |
| 17 | sessionStorage | | HTML5 sessionStorage cross-browser implementatio，用法详见  <http://www.open-open.com/lib/view/open1329285678983.html> |
| 18 | store | | Javascript实现浏览器本地存储api工具，用法详见  <http://www.open-open.com/lib/view/open1331779537249.html> |
| 19 | ztree | | 订单详细页面相关js |
| Jquery组件 | | | |
| 1 | Jquery.autocomplete | | jquery自动提示插件 |
| 2 | Jquery.cookie | | Jquery cookie插件 |
| 3 | Jquery.customparam | | Jquery ajax |
| 4 | Jquery.formvalidator | | Jquery表单验证插件 |
| 5 | Jquery.json | | Jquery json插件 |
| 6 | Jquery.templates | | Jquery模板处理插件 |
| 7 | Jquery.navigator | | Jquery 导航插件 |
| 8 | Jquery.pagination | | Jquery分页插件 |
| 9 | Jquery.plupload | | Jquey [Web文件上传](http://www.open-open.com/lib/view/open1337932090824.html)插件 |
| 10 | Jquery.poshytip | | Jquery 文本提示控件 |
| 11 | Jquery.tipInside | | Jquery 提示控件 |
| 12 | Jquery.ui.core | | Jquery ui核心 |
| 13 | Jquery.ui.datepicker | | Jquery日期框控件 |
| 14 | Jquery.validation | | Jquery验证框架 |

|  |  |  |
| --- | --- | --- |
| 序号 | 组件名称 | 组件职能 |
| 1 | 日志组件 |  |
| 2 | 模态弹出框组件 |  |
| 3 | 图片自适应 |  |
| 4 | 表格 |  |
| 5 | 可定位弹出框 |  |
| 6 | 进度条 |  |
| 7 | 可展开表格 |  |
| 8 | 滑动组件 |  |
| 9 | 可编辑表格 |  |
| 10 | 图片展示组件 |  |
| 11 | 唯一码生成器 |  |
| 12 | 数字日期格式化 |  |
| 13 | 计算函数（增） |  |
| 14 | 页面的输入提示组件 |  |
| 15 | 请求获取当前信息 |  |
| 16 | 本地缓存 |  |
| 工程组件 | | |
| 17 | 页头 |  |
| 18 | 页脚 |  |
| 19 | 顶部栏 |  |
| 20 | 登录 |  |
| 21 | 商品分类 |  |
| 22 | 最近浏览 |  |
| 23 | 二级导航 |  |
| 24 | 订单数量 |  |
| 25 | 公告 |  |
| 26 | 购物车 |  |
| 27 | 查询表单组件 |  |
| 28 | Tab表单组件 |  |
| 工程工具组件 | | |
| 29 | 上下文 |  |
| 30 | 行计算 |  |
| 31 | 自定义校验器 |  |

# 组件使用

## 对话框(dialog)



### 使用说明

*本组件的使用说明，包含关键属性、关键方法。*

<h2>界面</h2>

<div id=*"graybar"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use([ G.base+'common/bar/src/js/bar' ],

**function**(Bar) {

**var** option = {

attrs : {

classPrefix : "default",

title : '友情连接'

},

renderTo : 'graybar'//放到哪个div

};

**var** graybar = **new** Bar(option);

});

</script>

## 通知框( notice)



### 使用说明

var Notice = require('./notice.js');

Notice.show($('#resource-grid'), '添加成功');

## 基本表格（grid）



### 使用说明

<h2>界面</h2>

<div id=*"grid"* style="width: *650px*; display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('ecpgrid',

**function**(EcpGrid) {

**var** option = {

attrs : {

url : G.API.USER\_QUERY,

columns : [{

label : "用户编码",

key : "userCode",

width : "30%",

align : "left"

}, {

label : "用户名称",

key : "userName",

width : "20%",

align : "left"

},{

label : "手机",

key : "cellPhone",

width : "15%",

align : "left"

},{

label : "email",

key : "email",

width : "25%",

align : "left"

}]

},

renderTo : 'grid',

autoRender:**true**

};

**var** ecpgrid = **new** EcpGrid(option);

});

</script>

## 分页表格(pagegrid)



### 使用说明

<h2>界面</h2>

<div id=*"orderGrid"* style="float: *left*; width: *100%*"></div><script

type=*"text/javascript"*>

seajs.use('ecppagegrid',**function**(EcpPageGrid) {

**var** ecpGrid = **new** EcpPageGrid(

{

attrs : {

isPagination : **true**,

url : "/ecp/order/ajaxQryOrderListByCond",

columns : [

{

label : "订单商品",

key : "ccustomerpoid",

format : "ORDER\_PRODUCT",

width : 190,

align : "left"

},

{

label : "订单数量",

key : "ntotalnum",

width : 90,

align : "left"

},

{

label : "订单金额",

key : "nheadsummny",

format : "ORDER\_SUMMY",

width : 90,

align : "left"

},

{

label : "订单状态",

key : "fstatusflag",

format : "ORDER\_STATUS",

width : 90,

align : "left"

},

{

label : "订单日期",

key : "dbilldate",

format : "DATE",

width : 90,

align : "left"

} ]

},

renderTo : "orderGrid",

autoRender : **false**

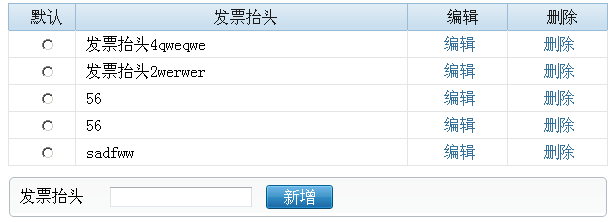
});

ecpGrid.query();

});

</script>

## 可编辑表格(editgrid)



### 使用说明

<h2>界面</h2>

<div id=*"grid"* style="width: *650px*; display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('ecpgrid',

**function**(EcpGrid) {

**var** option = {

attrs : {

url : G.API.USER\_QUERY,

columns : [{

label : "用户编码",

key : "userCode",

width : "30%",

align : "left"

}, {

label : "用户名称",

key : "userName",

width : "20%",

align : "left"

},{

label : "手机",

key : "cellPhone",

width : "15%",

align : "left"

},{

label : "email",

key : "email",

width : "25%",

align : "left"

}]

},

renderTo : 'grid',

autoRender:**true**

};

**var** ecpgrid = **new** EcpGrid(option);

});

</script>

## 列表（list）



### 使用说明

<h2>界面</h2>

<div id=*'list'* style="display: *inline-table*;">

<script type=*"text/javascript"*>

seajs.use('list',

**function**(List) {

**var** option = {

attrs:{

url:'/web/component/webpub/common/list/test/data.json'

},

renderTo:"list",

autoRender:**true**

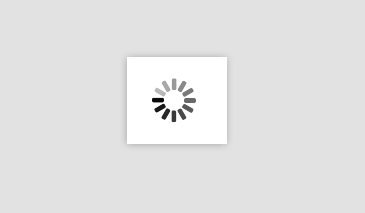
};

**var** list= **new** List(option);

});

</script>

## 加载中(loading)



### 使用说明

<h1>loading组件</h1>

<ol>

<li>

<h2>使用说明</h2>

<div id=*"preview"*></div>

<script type=*"text/javascript"*>

seajs.use("loading", **function**(Loading) {

Loading.load("preview");

});

</script>

</li>

## 日志(log)



### 使用说明

<h2>使用说明</h2>

<p>按alt+F11 显示和隐藏 日志框</p>

<script type=*"text/javascript"*>

seajs.use([ "$", 'log' ], **function**(

$, Log) {

Log.debug('this is debug message');

Log.error('this is error message');

Log.warn('this is warn message');

});

</script>

## 图片展示



### 使用说明

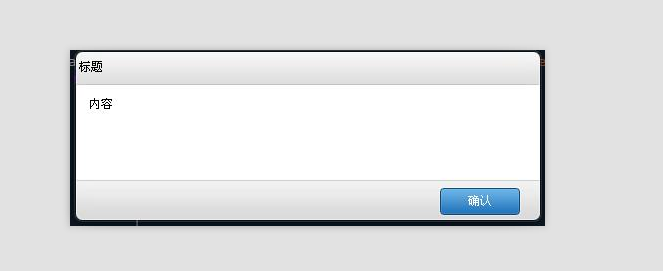
<li>

<h2>界面</h2>

<div data-widget=*"picturedisplay"*></div>

<li>

## 模态对话框



### 使用说明

<h2>界面</h2>

<div id=*"dialog"* style="width: *520px*;"></div>

<script type=*"text/javascript"*>

seajs.use('dialog', **function**(Dialog) {

**var** contentHtml = "<iframe id='addressEdit' data-resize='true' src='"+G.PAGE.USER\_EDIT+"' frameborder='0' scrolling='no' width='450px' height='200px'></iframe>";

Dialog.dialog({

title : '新增用户',

hasCloseTip : **true**,

content : contentHtml,

isHtmlContent : **true**,

pandTop:7,

type : 'dialog',

buttons : [ {

name : '保存',

isDefault : **true**,

href : '#',

method : **function**() {

alert("保存");

}

}

, {

name : '取消',

isDefault : **false**,

href : '#',

method : **function**() {

Dialog.close();

**return** **false**;

}

} ]

});

});

</script>

## 报价表



### 使用说明

<h2>界面</h2>

<div id=*"grid"* style="width: *712px*;display: *inline-block*;"></div>

<script type=*"text/javascript"*>

seajs.use('quoteGrid',

**function**(QuoteGrid) {

**var** quotegrid = **new** QuoteGrid({

attrs : {

url : 'data.json'

},

renderTo : "grid"

});

});

</script>

## 选项卡



### 使用说明

<li>

<h2>界面</h2>

<div id=*"tab"* style="width: *409px*; height:*200px*; display: *inline-block*;"></div> <script

type=*'text/javascript'*>

seajs.use(['tabbar','list'],

**function**(TabBar, List) {

**var** option = {

attrs : {

items : [ {

title : '采购动向',

content : '内容采购动向'

}, {

title : '政策法规',

content : '内容政策法规'

}, {

title : '公告',

content : **new** List({

attrs : {

url : 'data.json'

}

})

}, {

title : '其他',

content : '内容D'

} ],

triggerType : 'click',

activeIndex : 1,

moreurl : "http://www.baidu.com"

},

renderTo : 'tab',

theme : 'default',

model : '{}'

};

**var** tabbar = **new** TabBar(option);

});

</script></li>

## 选项面板



### 使用说明

<h2>界面</h2>

<div id=*"tab"* style="width: *409px*;"></div>

<script type=*'text/javascript'*>

seajs.use(['tabpanel','list'],

**function**(TabPanel, List) {

**var** option = {

attrs : {

items : [ {

title : '页签A',

content : **new** List({

attrs:{

url:'/web/component/webpub/common/list/test/data.json'

},

renderTo:"list",

autoRender:**true**

})

}, {

title : '页签B',

content : '内容A'

}, {

title : '页签C',

content : **new** List({

attrs:{

url:'/web/component/webpub/common/list/test/data.json'

},

renderTo:"list",

autoRender:**true**

})

}, {

title : '页签D',

content : '内容A'

} ],

triggerType : 'click',

activeIndex : 0

},

renderTo : 'tab',

theme : 'default',

model : '{}'

};

**var** tabPanel = **new** TabPanel(option);

});

</script>

## 横幅



### 使用说明

<div data-widget="banner"></div>

## 按钮



### 使用说明

<li>

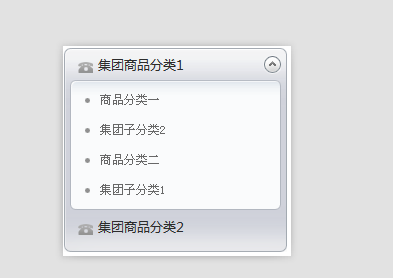
<h2>界面</h2>

<div style="display: *inline-block*;" data-widget=*"button"* data-attrs-id=*"button1"* data-attrs-text=*"普通按钮"* data-attrs-type=*"normal"*></div>

<div style="display: *inline-block*;" data-widget=*"button"* data-attrs-text=*"蓝色按钮"* data-attrs-type=*"blue"*></div>

<li>

## 类别



### 使用说明

<h2>界面</h2>

<div id=*"category"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('category',**function**(Category){

**var** options= {

attrs:{

url:'data.json'

},

renderTo:'category'

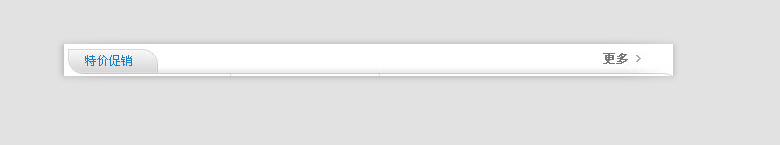
};

**var** c=**new** Category(options);

});

</script>

## 更多导航



### 使用说明

<h2>界面</h2>

<div id=*"bar"* style="width: *200px*; display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use(['/ecp/component/ecp/common/ecpbar/src/js/ecpbar' ],

**function**(EcpBar) {

**var** option2 = {

attrs : {

classPrefix : "default",

title : '查询结果',//头部标题

icon : 'query',//"查询"

top\_tip : **true**,//头部是否数字提示

top\_link : **true**,//头部是否显示"更多"链接

top\_url : 'http://www.baidu.com',//顶部链接的url,

top\_tip : **false**

},

renderTo : 'bar'//放到哪个div

};

**var** bar = **new** EcpBar(option2);

});

</script>

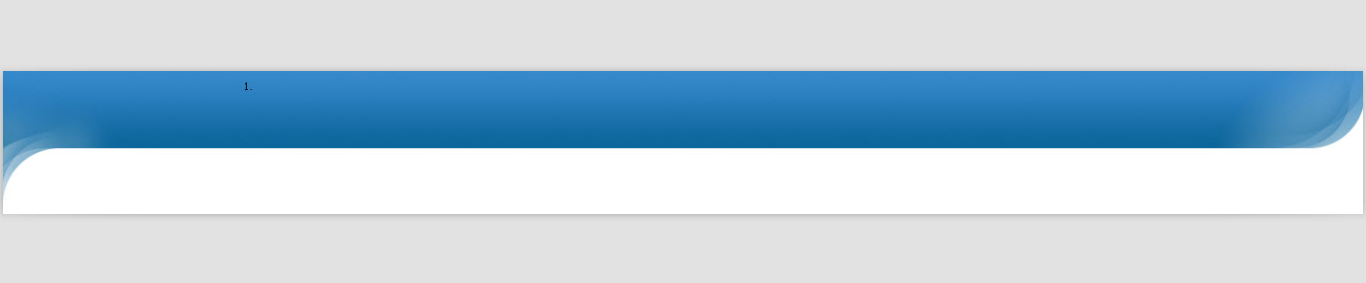
## 底部(footer)



### 使用说明

<div data-widget="footer"></div>

## 顶部(top)

**

### 使用说明

<div data-widget="top"></div>

## 表格



### 使用说明

<h2>界面</h2>

<div id=*"form"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('form',**function**(Form){

**var** form=**new** Form({

attrs:{

id:"productQuery",

url:"/ecp/order/qryOrderByCondition",

items:[

{

label:"商品分类",

name:"pkChannelNode",

type:"select",

emptyText:"全部",

options:"CODENAME\_PRODUCT\_CATEGORY"//关键字

},

{

label:"商品品牌",

name:"brandID",

type:"select",

emptyText:"全部",

options:"CODENAME\_PRODUCT\_BRAND"

}

],

buttons:[

{

text:"提交",

handler:**function**(){

productGrid.query($("#productQuery"));

**return** **false**;

}

},

{

text:"重置",

handler:**function**(){

$(':input','#productForm')

.not(':button, :submit, :reset, :hidden')

.val('')

.removeAttr('checked')

.removeAttr('selected');

**return** **false**;

}

}

]

},

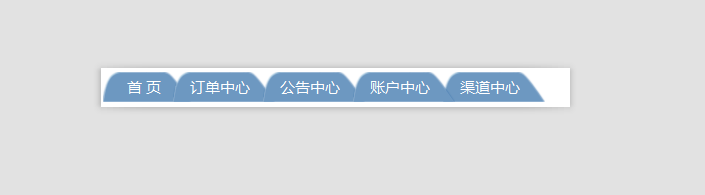
renderTo:'form'

});

});

</script>

## 导航条



### 使用说明

<h1>导航栏组件</h1>

<ol>

<li>

<h2>界面</h2>

<div data-widget=*"navigation"*></div>

</li>

## 模糊匹配检索



### 使用说明

<h2>界面</h2>

<input class=*"J\_autoComplete"* />

<script type=*"text/javascript"*>

seajs.use('productAutoComplete',**function**(ProductAutoComplete){

ProductAutoComplete.init({

callback : **function**(json) {

console.log(json);

}

});

});

</script>

## 搜索栏



### 使用说明

<h2>界面</h2>

<div id=*"search"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('search',**function**(Search){

**var** options= {

attrs:{

//对应搜索框的url

indexurl:'data.json',

//图标对应的url

iconUrl:"http://www,baidu.com",

//高级搜索对应的url

advanceQueryUrl:"http://www,baidu.com"

},

renderTo:'search'

};

**var** s=**new** Search(options);

});

</script>

## 购物车



### 使用说明

<h2>界面</h2>

<div id=*"shopcart"* ></div>

<script type=*"text/javascript"*>

seajs.use('shopcart',

**function**(ShopCart) {

**new** ShopCart({

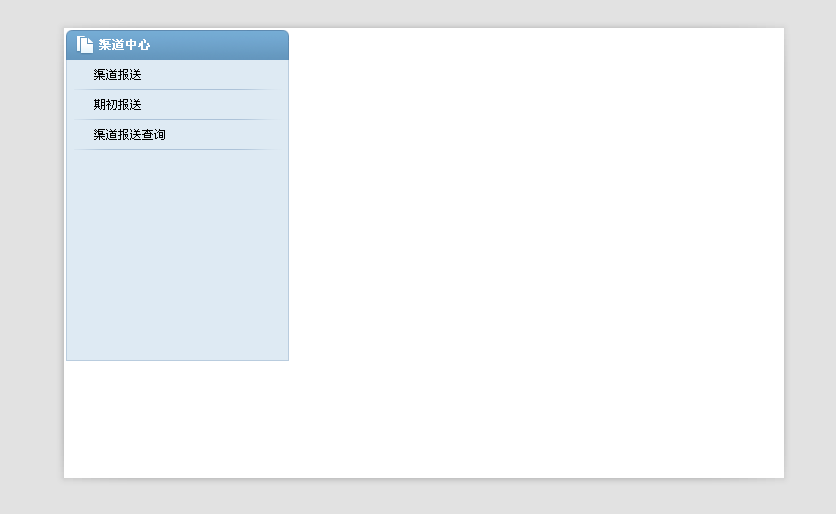
renderTo : "shopcart"

});

});

</script>

## 左侧导航



### 使用说明

<h2>界面</h2>

<div id=*"sidemenu"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('sidemenu',**function**(SideMenu){

**var** options= {

attrs:{

url:'data.json',

title:'渠道中心'

},

renderTo:'sidemenu'

};

**var** s=**new** SideMenu(options);

});

</script>

## 单点登陆



### 使用说明

<h2>界面</h2>

<div id=*"signon"*></div>

<script type=*"text/javascript"*>

seajs.use('signon',

**function**(Signon) {

**var** signon = **new** Signon({

renderTo : "signon"

});

});

</script>

## 单点登陆表单



### 使用说明

<h2>界面</h2>

<div id=*"signonform"*></div>

<script type=*"text/javascript"*>

seajs.use('signonform',**function**(SignOnForm) {

**var** signonform = **new** SignOnForm({

renderTo : "signonform"

});

});

</script>

## billball



### 使用说明

<h2>界面</h2>

<div id=*"billboard"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('billboard',**function**(Billboard){

**var** options= {

attrs:{

url:'data.json'

},

renderTo:'billboard'

};

**var** c=**new** Billboard(options);

});

</script>

## 公告（bulletin）



### 使用说明

<h2>界面</h2>

<div id=*"bulletin"* ></div>

<script type=*"text/javascript"*>

seajs.use('bulletin',

**function**(Bulletin) {

**new** Bulletin({

renderTo : "bulletin"

});

});

</script>

## floor



### 使用说明

<h2>界面</h2>

<div id=*"channel"* style="display: *inline-table*;"></div>

<script type=*"text/javascript"*>

seajs.use('/ecp/component/ecp/common/channel/src/js/channel',**function**(Channel){

**var** options= {

attrs:{

url:"/ecp/side/onsale?type=specialProd",

title:"特价促销",

moreLink:"javascript:viewOnSaleSpecial()"

},

renderTo:'channel'

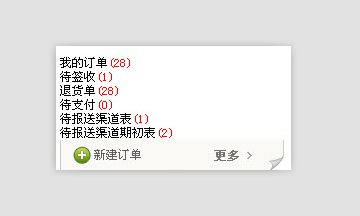
};

**var** c=**new** Channel(options);

});

</script>

## 订单（ orderbox）



### 使用说明

<h2>界面</h2>

<div id=*"orderbox"* ></div>

<script type=*"text/javascript"*>

seajs.use('orderbox',

**function**(OrderBox) {

**new** OrderBox({

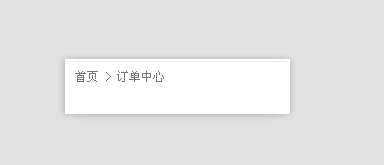
renderTo : "orderbox"

});

});

</script>

## 产品描述



### 使用说明

<li>

<h2>界面</h2>

<div data-widget=*"productdesc"*></div>

</li>

# WEB UIFrame开发规范

*制定WEB UIFrame的开发规范。目的：可参考公共开发规范*

## 层叠样式规范

### 文件规范

1. 文件均归档至约定的目录中

1、所有的CSS分为两大类：通用类和业务类。通用的CSS文件，放在如下目录中：

基本样式库 /css/core

通用UI元素样式库 /css/lib

JS组件相关样式库 /css/ui

业务类的CSS是指和具体产品相关的文件，放在具体的模块下

模块名/css/default（或主题名）/

2、文件引入可通过外联或内联方式引入

link和style标签都应该放入head中，原则上，不允许在html上直接写样式。

避免在CSS中使用@import，嵌套不要超过一层。

外联方式：<link rel=”stylesheet” href=”…” />（类型声明type=”text/css”可以省略）

内联方式：<style>…</style> （类型声明type=”text/css”可以省略）

1. 文件名、文件编码及文件大小

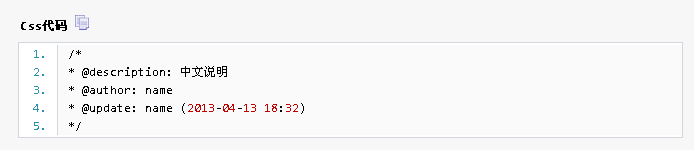
1、文件名必须由小写字母、数字、中划线组成

2、文件必须用UTF-8编码，使用UTF-8（非BOM），在HTML中指定UTF-8编码，在CSS中则不需要特别指定因为默认就是UTF-8。

3、单个CSS文件避免过大（建议少于300行）

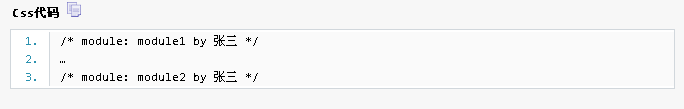
### 注释规范

**一、文件顶部注释（推荐使用）**



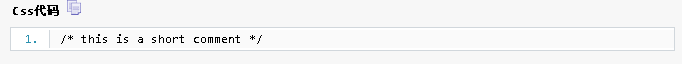
二、模块注释

模块注释必须单独写在一行

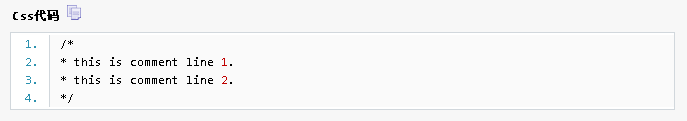


三、单行注释与多行注释

单行注释可以写在单独一行，也可以写在行尾，注释中的每一行长度不超过40个汉字，或者80个英文字符。

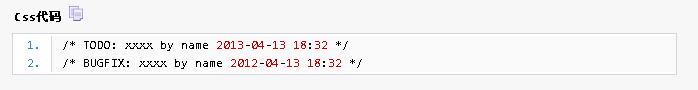


多行注释必须写在单独行内



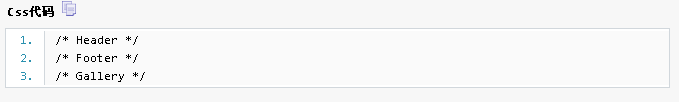
四、特殊注释

用于标注修改、待办等信息



五、区块注释

对一个代码区块注释（可选），将样式语句分区块并在新行中对其注释



### 命名规范

使用有意义的或通用的ID和class命名：ID和class的命名应反映该元素的功能或使用通用名称，而不要用抽象的晦涩的命名。反映元素的使用目的是首选；使用通用名称代表该元素不表特定意义，与其同级元素无异，通常是用于辅助命名；使用功能性或通用的名称可以更适用于文档或模版变化的情况。

/\* 不推荐: 无意义 \*/ #yee-1901 {}

/\* 不推荐: 与样式相关 \*/ .button-green {}.clear {}

/\* 推荐: 特殊性 \*/ #gallery {}#login {}.video {}

/\* 推荐: 通用性 \*/ .aux {}.alt {}

常用命名（多记多查英文单词）：page、wrap、layout、header(head)、footer(foot、ft)、content(cont)、menu、nav、main、submain、sidebar(side)、logo、banner、title(tit)、popo(pop)、icon、note、btn、txt、iblock、window(win)、tips等。

ID和class命名越简短越好，只要足够表达涵义。这样既有助于理解，也能提高代码效率。

/\* 不推荐 \*/ #navigation {}.atr {}

/\* 推荐 \*/ #nav {}.author {}

类型选择器避免同时使用标签、ID和class作为定位一个元素选择器；从性能上考虑也应尽量减少选择器的层级。

/\* 不推荐 \*/ul#example {}div.error {}

/\* 推荐 \*/#example {}.error {}

命名时需要注意的点：

规则命名中，一律采用小写加中划线的方式，不允许使用大写字母或 \_

命名避免使用中文拼音，应该采用更简明有语义的英文单词进行组合

命名注意缩写，但是不能盲目缩写，具体请参见常用的CSS命名规则

不允许通过1、2、3等序号进行命名

避免class与id重名

id用于标识模块或页面的某一个父容器区域，名称必须唯一，不要随意新建id

class用于标识某一个类型的对象，命名必须言简意赅。

尽可能提高代码模块的复用，样式尽量用组合的方式

规则名称中不应该包含颜色（red/blue）、定位（left/right）等与具体显示效果相关的信息。应该用意义命名，而不是样式显示结果命名。

一、常用id的命名：

(1) 页面结构

容器: container

页头：header

内容：content/container

页面主体：main

页尾：footer

导航：nav

侧栏：sidebar

栏目：column

页面外围控制整体布局宽度：wrapper

左右中：left right center

(2) 导航

导航：nav

主导航：mainbav

子导航：subnav

顶导航：topnav

边导航：sidebar

左导航：leftsidebar

右导航：rightsidebar

菜单：menu

子菜单：submenu

标题: title

摘要: summary

(3) 功能

标志：logo

广告：banner

登陆：login

登录条：loginbar

注册：regsiter

搜索：search

功能区：shop

标题：title

加入：joinus

状态：status

按钮：btn

滚动：scroll

标签页：tab

文章列表：list

提示信息：msg

当前的: current

小技巧：tips

图标: icon

注释：note

指南：guild

服务：service

热点：hot

新闻：news

下载：download

投票：vote

合作伙伴：partner

友情链接：link

版权：copyright

二、常用class的命名

(1) 颜色:使用颜色的名称或者16进制代码，如

.red { color: red; }

.f60 { color: #f60; }

.ff8600 { color: #ff8600; }

(2) 字体大小,直接使用”font+字体大小”作为名称，如

.font12px { font-size: 12px; }

.font9pt {font-size: 9pt; }

(3) 对齐样式,使用对齐目标的英文名称，如

.left { float:left; }

.bottom { float:bottom; }

(4) 标题栏样式,使用”类别+功能”的方式命名，如

.barnews { }

.barproduct { }

### 书写规范

1. 排版规范

(1) 使用4个空格，而不使用tab或者混用空格+tab作为缩进；

(2) 规则可以写成单行，或者多行，但是整个文件内的规则排版必须统一；

单行形式书写风格的排版约束：

* + - 如果是在html中写内联的css，则必须写成单行；
    - 每一条规则的大括号 { 前后加空格 ；
    - 每一条规则结束的大括号 } 前加空格；
    - 属性名冒号之前不加空格，冒号之后加空格；
    - 每一个属性值后必须添加分号; 并且分号后空格；
    - 多个selector共用一个样式集，则多个selector必须写成多行形式 ；
    - 多行形式书写风格的排版约束：
    - 每一条规则的大括号 { 前添加空格；
    - 多个selector共用一个样式集，则多个selector必须写成多行形式；
    - 每一条规则结束的大括号 } 必须与规则选择器的第一个字符对齐；
    - 属性名冒号之前不加空格，冒号之后加空格；
    - 属性值之后添加分号；

1. 属性编写顺序

* 显示属性：display/list-style/position/float/clear …
* 自身属性（盒模型）：width/height/margin/padding/border
* 背景：background
* 行高：line-height
* 文本属性：color/font/text-decoration/text-align/text-indent/vertical-align/white-space/content…
* 其他：cursor/z-index/zoom/overflow
* CSS3属性：transform/transition/animation/box-shadow/border-radius
* 如果使用CSS3的属性，如果有必要加入浏览器前缀，则按照 -webkit- / -moz- / -ms- / -o- / std的顺序进行添加，标准属性写在最后。
* 链接的样式请严格按照如下顺序添加： a:link -> a:visited -> a:hover -> a:active

1. 规则书写规范

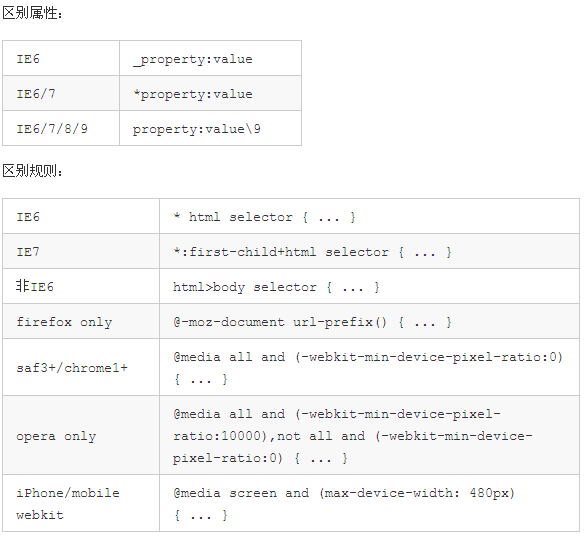
* 使用单引号，不允许使用双引号;
* 每个声明结束都应该带一个分号，不管是不是最后一个声明;
* 除16进制颜色和字体设置外，CSS文件中的所有的代码都应该小写;
* 除了重置浏览器默认样式外，禁止直接为html tag添加css样式设置;
* 每一条规则应该确保选择器唯一，禁止直接为全局.nav/.header/.body等类设置属性;

1. 代码性能优化

* 合并margin、padding、border的-left/-top/-right/-bottom的设置，尽量使用短名称。
* 选择器应该在满足功能的基础上尽量简短，减少选择器嵌套，查询消耗。但是一定要避免覆盖全局样式设置。
* 注意选择器的性能，不要使用低性能的选择器。
* 禁止在css中使用\*选择符。
* 除非必须，否则，一般有class或id的，不需要再写上元素对应的tag。
* 0后面不需要单位，比如0px可以省略成0，0.8px可以省略成.8px。
* 如果是16进制表示颜色，则颜色取值应该大写。
* 如果可以，颜色尽量用三位字符表示，例如#AABBCC写成#ABC 。
* 如果没有边框时，不要写成border:0，应该写成border:none 。
* 尽量避免使用AlphaImageLoader 。
* 在保持代码解耦的前提下，尽量合并重复的样式。
* background、font等可以缩写的属性，尽量使用缩写形式 。

1. CSS Hack的使用

请不用动不动就使用浏览器检测和CSS Hacks，先试试别的解决方法吧！考虑到代码高效率和易管理，虽然这两种方法能快速解决浏览器解析差异，但应被视为最后的手段。在长期的项目中，允许使用hack只会带来更多的hack，你越是使用它，你越是会依赖它！   
  
推荐使用下面的：



1. 字体规则

* 为了防止文件合并及编码转换时造成问题，建议将样式中文字体名字改成对应的英文名字，如：黑体(SimHei) 宋体(SimSun) 微软雅黑 (Microsoft Yahei，几个单词中间有空格组成的必须加引号)
* 字体粗细采用具体数值，粗体bold写为700，正常normal写为400
* font-size必须以px或pt为单位，推荐用px（注：pt为打印版字体大小设置），不允许使用xx-small/x-small/small/medium/large/x-large/xx-large等值
* 为了对font-family取值进行统一，更好的支持各个操作系统上各个浏览器的兼容性，font-family不允许在业务代码中随意设置

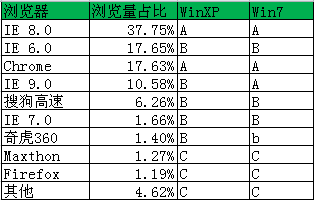
### 测试规范

1. 了解浏览器特效支持

为了页面性能考虑，如果浏览器不支持CSS3相关属性的，则该浏览器的某些特效将不再支持，属性的支持情况如下表所示（Y为支持，N为不支持）： 



1. 设定浏览器支持标准



* A级－交互和视觉完全符全设计的要求
* B级－视觉上允许有所差异，但不破坏页面的整体效果
* C级－可忽略设计上的细节，但不防碍使用

1. 常用样式测试工具

W3C CSS validator：<http://jigsaw.w3.org/css-validator/>

 CSS Lint：<http://csslint.net/>

 CSS Usage：<https://addons.mozilla.org/en-us/firefox/addon/css-usage/>

### 其他规范

* 不要轻易改动全站级CSS和通用CSS库。改动后，要经过全面测试。
* 避免使用filter
* 避免在CSS中使用expression
* 避免过小的背景图片平铺。
* 尽量不要在CSS中使用!important
* 绝对不要在CSS中使用”\*”选择符
* 层级(z-index)必须清晰明确，页面弹窗、气泡为最高级（最高级为999），不同弹窗气泡之间可在三位数之间调整；普通区块为10-90内10的倍数；区块展开、弹出为当前父层级上个位增加，禁止层级间盲目攀比。
* 背景图片请尽可能使用sprite技术, 减小http请求, 考虑到多人协作开发, sprite按照模块、业务、页面来划分均可。

## JS代码规范

### JavaScript文件

JavaScript程序应该作为一个.js文件存储和发布。

JavaScript代码不应该嵌入在HTML文件里，除非那些代码是一个单独的会话特有的。

HTML里的JavaScript代码大大增加了页面的大小，并且很难通过缓存和压缩来缓解。

<script src=filename.js>标签应该在body里越靠后的位置越好。这减少了由

于加载script而导致的其它页面组件的延迟。没有必要使用language或者type属性。由服务器而不是script标签来决定MIME类型

### 缩进

缩进的最小单位是4个空格。不要使用tab键，because (as of this writing in the 21st Century) there still is not a standard for the placement of tabstops. 使用空格会引起文件变大，但是这点大小对局域网无关紧要，而且差别被minification消除了

### 行宽

不要让一行代码超过80个字符。当一条语句不能在单独一行写完时，可能有必要拆分它。在操作符后进行拆分，最好是在逗号后面拆分。 操作符后面进行拆分减少了通过插入分号伪装copy-paste错误的可能性。下一行应该缩进8个空格。

### 注释

慷慨的写注释。留下一些供需要理解你做了什么的人们(可能是你自己)下次阅读的信息是有用的。注释应该书写良好和清晰，就像它们 标注的代码一样。偶尔小幽默一把也是可以的。挫折和怨恨就别写了。 更新注释非常重要。错误的注释让程序更难读懂和理解。 让注释有意义。更多的关注于不能马上可见的东西。不要用如下内容浪费读者的时间:

Java代码 1. i = 0; // Set i to zero.

一般使用行注释。把块注释用于正式文档或外部注释。

### 变量声明

所有的变量应该在使用前声明。JavaScript不强求这点，但是这样做会让程序更易读，并且会让探测未声明的可能变成隐式的globals的 变量更容易。 var语句应该为方法体内的第一个语句。 每个变量声明应该自己占一行并有注释。它们应该按字母顺序排列。

Java代码

1. var currentEntry; // currentyly selected table entry

2. var level; // indentation level

3. var size; // size of table

JavaScript没有块作用域，所以在块里定义变量可能会让有其它C家族语言经验的程序员迷惑。在方法顶端定义所有变量。 尽量少使用全局变量。隐式的全局变量应该从来不使用。

### 方法声明

所有的方法应该在它们使用前声明。内部方法应该位于var语句后面。这让哪些变量包含在它的scope里更清楚。 方法名和参数列表的“(”(左圆括号)之间不应该有空格。在“)”(右圆括号)和“{”(左大括号)之间有一个空格。 方法体本身缩进4个空格。“}”(右大括号)应该和方法声明处对齐。

Java代码

1. function outer(c, d) {

2. var e = c \* d;

3.

4. function inner(a, b) {

5. return (e \* a) + b;

6. }

7.

8. return inner(0, 1);

9. }

这个规范可以和JavaScript很好的工作，因为在JavaScript里方法和对象变量可以放在允许表达式的任何位置。它使用内部方法和复杂

结构提供最好的可读性。

Java代码

1. function getElementsByClassName(className) {

2. var results = [];

3. walkTheDOM(document.body, function (node) {

4. var a; // array of class names

5. var c = node.className; // the node's classname

6. var i; // loop counter

7. // If the node has a class name, then split it into a list of simple names.

8. // If any of them match the requested name, then append the node to the set of results.

9. if (c) {

10. a = c.split(' ');

11. for (i = 0; i < a.length; i += 1) {

12. if (a[i] === className) {

13. results.push(node);

14. break;

15. }

16. }

17. }

18. });

19. return results;

20. }

如果一个方法字面量为匿名的，则在“function”和“(”(左圆括号)之间应该有一个空格。如果省略空格，则它可能看起来方法名是

“function”，而这是错误的。

Java代码

1. div.onclick = function (e) {

2. return false;

3. };

4. that = {

5. method: function () {

6. return this.datum;

7. },

8. datum: 0;

9. };

尽量少用全局方法。

### 命名

命名应该由26个大小写字母(A .. Z, a .. z)，10个数字(0 .. 9)和\_(下划线)组成。不要使用国际字符，因为它们可能不易读或者不能在任何地方都能容易理解。不要在名字里使用$(美元符号)或\(反斜线符号)。不要使用\_(下划线)作为名字的首字母。它有时被用来表示私有，但是它实际上不提供私有性。

如果私有性很重要，使用提供私有性的[private members](http://javascript.crockford.com/private.html)。   
Avoid conventions that demonstrate a lack of competence.   
  
大多数变量和方法名应该以小写字母开始。   
  
必须使用[new](http://yuiblog.com/blog/2006/11/13/javascript-we-hardly-new-ya/)前缀的构造函数应该以大写字母开始。JavaScript不会在省略new时报编译期警告或运行时警告。   
不使用new时会发生坏事情，所以大写首字母规范是我们拥有的唯一的防御。   
  
全局变量应该全部使用大写字母。(JavaScript没有宏或常量，所以没有多少要求使用大写字母来表示JavaScript的特性的场景)

### 语句

**简单语句**   
每行应该包含至少一个语句。在每个简单语句末尾添加一个“;”(分号)。注意一个给方法字面量或对象字面量赋值的赋值语句仍然是   
一个赋值语句，所以也必须以分号结尾。   
  
JavaScript允许任何表达式作为语句使用。这可能产生一些错误，特别是在插入分号时。唯一可以当作语句使用的表达式是赋值表达式和   
调用表达式。   
  
**复合语句**   
复合语句是包含一个用“{}”(大括号)包围语句列表的的语句。   
1，包围的语句应该再缩进4个空格。   
2，“{”(左大括号)应该位于开始复合语句的行的末尾。   
3，“}”(右大括号)应该新起一行并且和相匹配的“{”所在那行的起始位置对齐   
4，当语句是控制结构的一部分时，所有语句都应该用括号包围，即使是单行语句，例如if或for语句。这让添加语句更容易而且不会引起八哥。   
  
**标签**   
语句标签是可选的。只有如下语句需要被标签标识: while，do，for，switch。   
  
**return语句**   
具有值的return语句不应该使用“()”(圆括号)包围值。返回值表达式必须和return关键字在同一行从而避免插入分号。   
  
**if语句**   
if语句应该使用如下格式:

Java代码  [收藏代码](javascript:void())

1. if (condition) {
2. statements;
3. }
5. if (condition) {
6. statements;
7. } else {
8. statements;
9. }
11. if (condition) {
12. statements;
13. } else if (condition) {
14. statements;
15. } else {
16. statements;
17. }

**for语句**   
for语句应该使用如下格式:

Java代码  [收藏代码](javascript:void())

1. for (initialization; condition; update) {
2. statements;
3. }
5. for (variable in object) {
6. statements;
7. }

第一种格式应该和数组使用。   
  
第二种格式应该和对象使用。注意添加到对象的prototype中的成员将被包含在遍历中。通过使用hasOwnProperty方法来区分对象的   
成员是明智的:

Java代码  [收藏代码](javascript:void())

1. for (variable in object) {
2. if (object.hasOwnProperty()) {
3. statements;
4. }
5. }

**while语句**   
while语句应该使用如下格式:

Java代码  [收藏代码](javascript:void())

1. while (condition) {
2. statements;
3. }

**do语句**   
do语句应该使用如下格式:

Java代码  [收藏代码](javascript:void())

1. do {
2. statements;
3. } while (condition);

不像其它复合语句，do语句始终使用“;”(分号)结尾。   
  
**switch语句**   
switch语句应该有如下格式:

Java代码  [收藏代码](javascript:void())

1. switch (expression) {
2. case expression:
3. statements;
4. default:
5. statements;
6. }

每个case和switch对齐，这避免了缩进过度。   
  
每组语句(除了default)应该以break，return或者throw结束。不要fall through。   
  
**try语句**   
try语句应该使用如下格式:

Java代码  [收藏代码](javascript:void())

1. try {
2. statements;
3. } catch (variable) {
4. statements;
5. }
7. try {
8. statements;
9. } catch (variable) {
10. statements;
11. } finally {
12. statements;
13. }

**continue语句**   
不要使用continue语句。它会让方法的控制流程模糊。   
  
**with语句**   
[不要使用with语句](http://yuiblog.com/blog/2006/04/11/with-statement-considered-harmful/)。

### 空格

空行通过将逻辑相关的代码放到一起来增加可读性。   
  
空格应该用于如下情况:   
1，关键字后面跟“(”(左圆括号)时应该用一个空格隔开。

Java代码  [收藏代码](javascript:void())

1. while (true) {

2，方法名和方法的“(”(左圆括号)之间不要有空格。这利于区分关键字和方法调用。   
3，所有的二元操作符，除了“.”(圆点)、“(”(左圆括号)和“[”(左中括号)，都应该使用一个空格来和操作数隔开。   
4，一元操作符和操作数之间不应该使用空格隔开，除了操作符是一个单词时，如typeof。   
5，for语句控制部分的每个“;”(分号)应该在后面跟一个空格。   
6，每个“,”(逗号)后面应该跟一个空格。

### 额外的建议

**{}和[]**   
使用{}替代new Object()。使用[]替代new Array()。   
当成员名字为连续的整数时使用数组。当成员名字为任意的字符串或名字时使用对象。   
  
**逗号操作符**   
不要使用逗号操作符，除了for语句的控制部分的严格使用。(这不适合逗号操作符，它应该用于对象字面量，数组字面量，var语句和参数   
列表。)   
  
**块作用域**   
在JavaScript里块没有作用域，只有方法有作用域。不要使用块，除了复合语句一定需要用到外。   
  
**赋值表达式**   
不要在if和while语句的条件部分做赋值。不要写不易懂的代码。   
  
**===和!==操作符**   
始终使用===和!==操作符会更好。==和!=操作符会做类型强制转换。特别是，不要使用==来和“假”值做比较。   
  
**令人混淆的加和减**   
注意不要在“+”后面跟“+”或“++”。这种模式令人混淆。在它们之间插入圆括号来让你的意图更清晰。

Java代码  [收藏代码](javascript:void())

1. total = subtotal + +myInput.value;
3. // is better written as
5. total = subtotal + (+myInput.value);

这样“+ +”就不会被读错成“++”。   
  
**邪恶的eval**   
eval方法是JavaScript里最滥用的特性。不要使用它。   
eval有别名。不要使用Function构造函数。不要传递字符串给setTimeout或者setInterval。

## HTML代码规范

### 基本规范

* 使用符合语义的标签书写HTML文档，选择恰当的元素表达所需的含义；
* 元素的标签和属性必须小写，属性值必须加双引号。
* 元素嵌套遵循（X）HTML Strict 嵌套规则，推荐使用FireFox 插件HTML Validator进行校验
* 正确区分自闭合元素和非闭合元素。非法闭合包括<br>…</br>、<script/>、<iframe/>非法闭合会导致嵌套错误问题
* 通过给元素设置自定义属性来存放与JavaScript交互的数据，属性名格式为data-xx(例如：data-lazyload-url)

### 文档说明

(1)XHTML1.0 过渡型

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

(2)HTML5

<!DOCTYPE html>

### 页面编码

(1)XHTML1.0 过渡型

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

(2)HTML5

<meta charset="utf-8" />

### 关键字与描述

<meta name="keywords" content="关键词一,关键词二," />

<meta name="description" content="页面描述" />

查看规范模板：[XHTML页面模板](http://www.56.com/style/-doc-/v1/tpl/css_dev_spec/w3c_html.html)、[HTML5页面模板](http://www.56.com/style/-doc-/v1/tpl/css_dev_spec/w3c_html.html)

### 其他

* 页面中**不要**使用&nbsp进行缩进，如需缩进，使用CSS控制
* HTML页面中**不要**使用行内样式
* 当空标签使用背景图时，必须加上相应该的文字说明，并按需隐藏文字
* 代码缩进，必须使用四个空格的TAB键
* HTML标签名、属性名必须全部采用小写，属性必须加引号，并且必须闭合，单标签也必须闭合，如：<input type=”text” />、<br />
* 内联元素禁止包裹块级元素
* HTML标签使用必须语义化
  + 如：标题根据重要性用h(h1、h2……)，段落用p标签，无序列表用ul标签，链接用a标签等
* 书写链接地址时, 必须避免重定向，如：href="http://www.56.com/", 即须在URL地址后面加上“/”
* HTML字符需使用转义符：

|  |  |  |
| --- | --- | --- |
| &lt; | < | 小于号 |
| &gt; | > | 大于号 |
| &amp; | & | 连接符 |
| &quot; | " | 引号 |
| &reg; | ® | 注册 |
| &copy; | © | 版权 |
| &trade; | ™ | 商标 |
| &middot; | · | 中点 |

## JSP编码规范

### 编程规范确立的原因：

1：增加代码可读性

2：减少培训成本

3：确定开发小组的规范文档 文件名和文件放置位置：

文件名和文件放置位置：

|  |  |  |
| --- | --- | --- |
| File Type | File Suffix | Recommended Location |
| JSP technology | .jsp | <context root>/<subsystem path>/ |
| JSP fragment | .jsp | <context root>/<subsystem path>/ |
| .jspf | <context root>/WEB-INF/jspf/<subsystem path>/ |
| cascading style sheet | .css | <context root>/css/ |
| JavaScript technology | .js | <context root>/js/ |
| HTML page | .html | <context root>/<subsystem path>/ |
| web resource | .gif, .jpg, etc. | <context root>/images/ |
| tag library descriptor | .tld | <context root>/WEB-INF/tld/ |

### jsp/jsp fragment文件

开头注释

服务器端开头注释注释范例之一

『比较完整的版本』：

<%-- - Author(s):

- Date:

- Copyright Notice:

- @(#): - Description:

--%>

『注』@(#)可被某些程序识别为说明标识符开始的标志。尽管并不是所有程序都能识别，但是加上也没有害处。

当然，上面的开头注释比较复杂，较简单的开头注释范例如下：

<%-- - Author(s):

- Date:

- Copyright Notice:

--%>

有些情况下，需要将开头注释说明部分传送到客户端。

客户端开头注释范例

<!-- - Author(s):

- Date:

- Copyright Notice:

-->

### jsp指令

两种选择，

选择一：

<%@ page session=”false” %>

<%@ page import=”java.util.\*”%>

<%@ page errorPage=”/common/errorPage.jsp”%>

选择二

<%@ page session=”false”import=”java.util.\*” errorPage=”/common/errorPage.jsp” %>

选择一和选择二的效果一样，除了选择一会引入两个空行。

一般情况下，选择二是推崇的选择，但是当导入多个java包时可能会导致选择二过长。

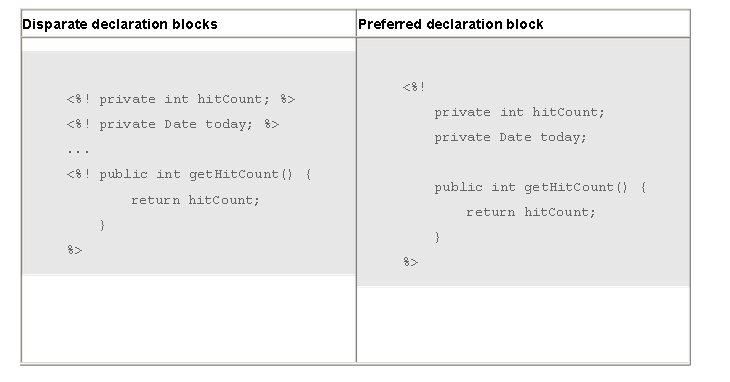
### tag library directives 标签库指令

<%@ tablib uri=”URI1” prefix=”tagPrefix1” %>

<%@ tablib uri=”URI2” prefix=”tagPrefix2” %>

从jsp1.2 开始，JSTL(JSP Standard Tag Library) JSP标准标签库被推荐使用，以减 少页面代码中jsp scriptlets中代码的数量，而且使用JSTL的代码更容易读写和维护。

### jsp 声明



右边的声明是推荐的格式，将一个网页的声明写到一个<%! %>中。

其编码规范和普通 java程序的编码规范没有区别。

### Html 和 jsp 代码

一个jsp页面包含以下几部分：html代码、jsp代码、jsp表达式、scriptlets、JavaBeans 指令。

TLD – Tag library Descriptor 标签库描述符

TLD文件必须以一个合适的XML声明和正确的DTD声明开头。

例如，一个jsp1.2 TLD 文件必须的开头是：

<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN" "http://java.sun.com/dtd/web-jsptaglibrary\_1\_2.dtd">

这个后面紧跟的才是开头注释部分。   
上面这些规则对jsp文件和jsp fragment文件都是适用的。

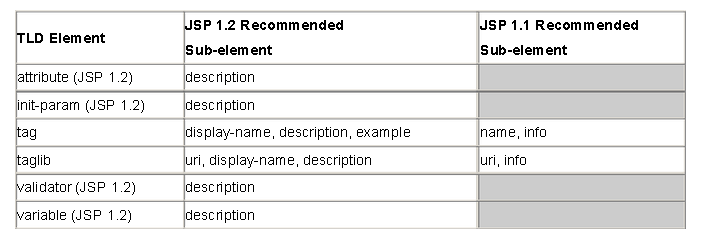
除了以上几部分，在文件中还可以包括以下几部分：

一个tag library validator 『可选』

declaration of event listeners 『可选』

Declaration of one or more available tags

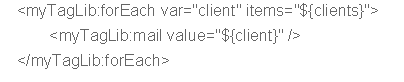
建议在你的TLD文件中包含这些可选的子元素



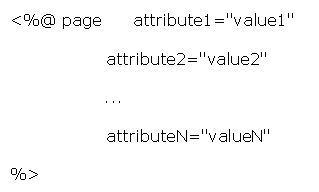
关于缩进

建议使用4个空格而不是Tab进行缩进，因为不同的编辑器对Tab代表的空格数解释不同。

示例一『只包含一个缩进』：



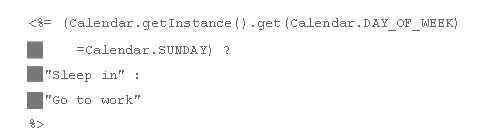
示例二『包含多个缩进的代码』： WEB业务实现开发规范



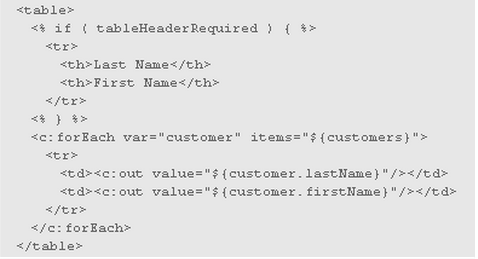
Scripting元素的缩进

Scripting元素包括声明、scriptlet、表达式等，

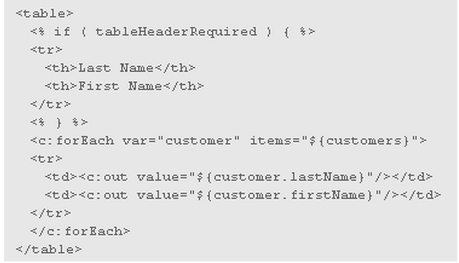
当这些元素在页面中占用的地方超过一行 时进行换行，换行后的示例如下



jsp html java 混合在一起时的缩进： 较好的模式



较差的模式：



『上面两中模式的区别』较好的模式即上面那种，脚本语言前空两格，并不影响html语言的缩进，后者都缩进四格，代码没有区别，比较混乱。

普通注释 jsp注释

|  |  |  |
| --- | --- | --- |
| Line | JSP scriptlet with scripting language comment | Pure JSP comment |
| single | <% /\*\* ... \*/ %>  <% /\* ... \*/ %>  <% // ... %> | <%-- ... --%> |
| multiple | <% /\*  \* ... \* \*/ %>  或  <%  //  // ... // %> | ... - -- %> |

html注释：

<!-- toolbar section -->

...

<!-- left-hand side navigation bar -->

...

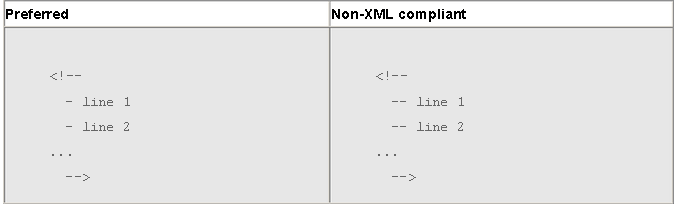
<!-- main body -->

...

<!-- footer -->

『说明』html注释的两种情况：1）版本和权限说明 2）Html开发人员用来对代码的注释说明

多行注释时需注意的情况

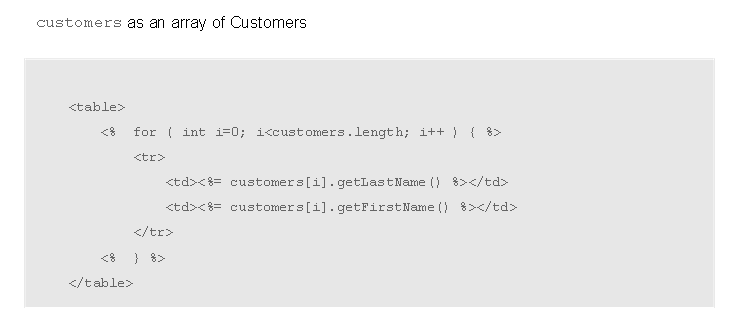


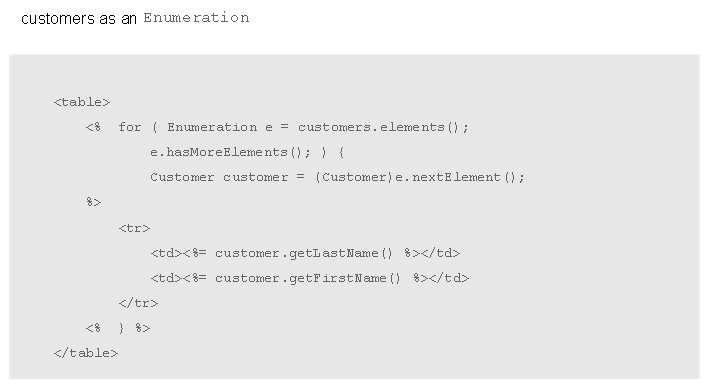
『注』推荐前一种注释方法

JSP Scriptlets

在可能的情况下，当tag libraries能提供相同的功能，尽量使用tag libraries。这样做的 好处：这样写的代码容易读写和维护、使业务逻辑和页面表现相分离、让你的代码更容易向jsp 2.0规范的代码靠拢。

举例：



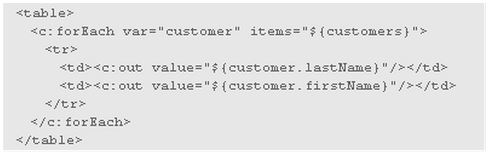


对于不同数据类型的customers，必须对应不同的scriptlet代码。如果使用Tag library，当customers为不同的数据类型时，代码能提供更高的可扩展性。

例如在

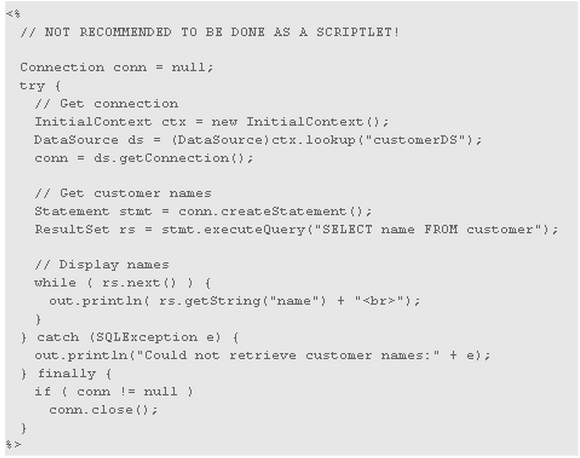
JSTL

中，下面的代码既支持数据类型又支持枚举类型的customers：

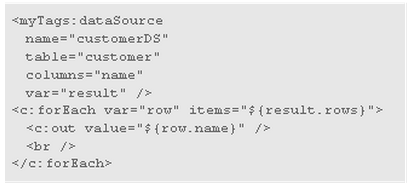


例如在JSTL中，下面的代码既支持数据类型又支持枚举类型的customers：

按照MVC设计模式的思想－降低业务逻辑和表现层的强耦合，jsp scriptlets最好不要用 来写业务逻辑。尽管可以用jsp scriptlet来传递服务器处理客户请求后返回客户端的数据，但是最好也不要这么做，因为用一个前端的控制器servelet或者是客户tag标签可能会更好。 例如，如下代码展示如何将从服务器中取得的客户名字直接在客户端显示，首先是用jsp scriptlet『jsp 脚本』来实现



这种方法将业务逻辑层和表现层强耦合，不如下面的代码



『对第二种代码的解释』：将于数据库的交互委托给tag myTags:dataSource，隐藏了具体实现细节。