# 1. 页面通用规则

## 1.1 域名规则

### 1.1.1 静态资源，包括JS、CSS、图片、文本等，应使用不携带cookie的域名。

由于安全证书问题，在https访问时，静态资源使用Imgssl.suning.com域名。

具体域名使用规则如下：

|  |  |  |
| --- | --- | --- |
| 文件类型 | 协议 | 域名 |
| JS/CSS/CSS中引用的图片 | http | script.suning.cn |
| 图片（不包括价格图片）,TXT | http | image.suning.cn  image1.suning.cn  image2.suning.cn  image3.suning.cn  image4.suning.cn  image5.suning.cn |
| 价格图片 | http | price1.suning.cn  price2.suning.cn |
| JS/CSS/图片 | https | Imgssl.suning.com |

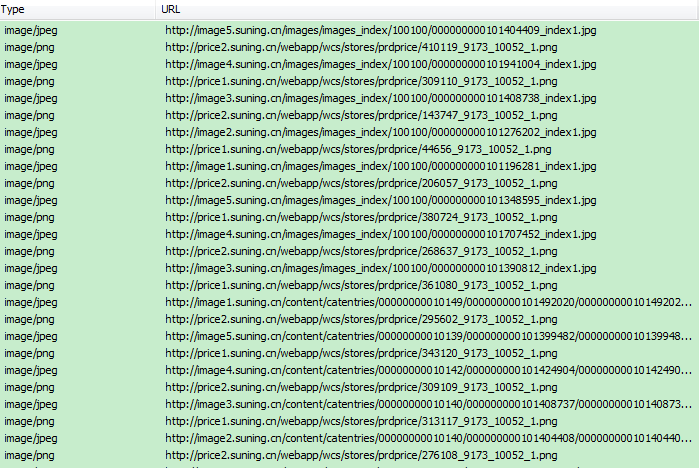
### 1.1.2 在同一页面中，域名应均匀使用。

某些浏览器，对同一域名的资源的同时下载有数量限制。所以，域名应尽量分散，以同步下载更多的资源。

使用方法：

在img标签中定义src属性为${imagesDomain.imageDomianDir}开头，实现域名分散调用

src="${imagesDomain.imageDomianDir}/images/images\_index/100100/imgname.jpg"



## 1.2 Cookie使用规范

Cookie用于在浏览器端记录用户相关信息。

### 1.2.1 Cookie应尽量简短，新建Cookie需要经过评审。

由于目前我们的cookie已经过大，所以非必要的cookie，不允许添加。

在设计文档中写明使用cookie的原因，cookie的各项属性值（包括cookie名称、内容、domain、path、是否ssl、过期时间），估算cookie的长度。

Cookie命名尽量简短，最多6个字节。Cookie内容也应精简。

### 1.2.2 Cookie中的关键业务数据必须进行加密。

建议不要将敏感数据放入Cookie。如用户个人隐私信息。

### 1.2.3 Cookie的有效期

Cookies的Expires属性规定了Cookies的结束作用时间。如果要建立一个会话期间的cookies，即要在客户结束浏览后失效，则一定不能设置该属性。

## 1.3 缓存

### 1.3.1 前台浏览页面，应设置缓存，缓存时间应与业务部门商定。

包括首页、2级、2.5级、3级、4级页面、图书各级页面、各类促销页面、以及其他不涉及用户个人信息且页面HTML一致的页面。

此类页面应静态化，即后缀名为html或htm，以使CDN能够缓存。

### 1.3.2 静态资源，应设置较长时间缓存，目前为10天。

静态资源修改时，建议更换文件名，以便及时生效。

如果不能修改文件名，则需要推送CDN。

nginx设置资源文件参考如下：

location ~\*.(css|js|jpg|jpeg|gif|png|ico)$ {  
root /www/soluo/blog;  
expires 10d;  
}

jsp上做如下设置：

<% response.setDateHeader("Expires",日期); %>

## 压缩

### 1.4.1应在WEB SERVER，对HTML、CSS、JS以及其他文本资源开启Gzip压缩。

Gzip开启以后会将输出到用户浏览器的数据进行压缩的处理，这样就会减小通过网络传输的数据量，提高浏览的速度。

### 1.4.2不应对图片、PDF、视频等其他二进制资源进行Gzip压缩。

### 1.4.3缩减js，应使用工具对js文件及页面内嵌js代码进行缩减。

相关工具，[Closure Compiler](http://code.google.com/closure/compiler/)或[YUI Compressor](http://developer.yahoo.com/yui/compressor/)

[YUI Compressor快速使用指南](http://www.36ria.com/3948)

### 1.4.4 缩减css，应使用工具对css文件及页面内嵌css代码进行缩减。

相关工具，[YUI Compressor](http://developer.yahoo.com/yui/compressor/) 或[cssmin.js](http://www.phpied.com/cssmin-js/)

## 1.5延迟加载

### 1.5.1 页面中，用户视野看不到的图片、IFRAME，应延迟加载，以节约下载资源，加快页面展示速度。

目前前台已实现延迟加载功能，使用的需引入对应的js文件，并在调用是修改img标签的src属性为lazyload定义的属性src2。具体代码如下（仅供参考），

**function** lazyload(option) {

//取消自定义对象类型，defHeight无用处

//获取所有img并且包含属性src2的元素

**var** defObj = $(option).find("img[src2]");

**var** pageScrollTop = **function**() {

**return** isIPad ? window.pageYOffset : Math.max(document.documentElement.scrollTop, document.body.scrollTop);

};

**var** imgLoad = **function**() {

//遍历之前提取pageTop()

**var** clientHeight = document.body.clientHeight;

**var** ot = pageScrollTop();

defObj.each( **function**() {

**var** vTop = $(**this**).offset().top;

//只在当前屏能显示出的图片

**if** (vTop <= (ot + clientHeight) && vTop >= ot) {

**var** src2 = $(**this**).attr("src2");

**if** (src2) {

$(**this**).attr("src", src2).removeAttr("src2");

//集合中删除此元素，之后的遍历不会与之有关，减少遍历次数

defObj=defObj.not($(**this**)[0]);

}

}

})

};

$(window).bind("scroll", **function**() {

imgLoad();

})

};

### 1.5.2 页面中引入的JS文件和JS片段，应尽量在页面后部，以避免阻塞页面加载。

## 1.6 HTTP STATUS

### 1.6.1 出现HTTP 4XX 和5XX 错误的时候，不应出现原始报错页面，而应指向特定的页面。





### 1.6.2 应尽量避免页面重定向。

尽量不要使用Response.Redirect("Default.jsp")等重定向语句。

### 1.6.3 有些HTTP请求无需返回内容，应使用HTTP 204替代HTTP 200。

## 1.7 SEO规范

### 1.7.1 URL应符合面包屑（bread crumb）格式。

面包屑是导航的一种形式，它清楚的表明当前用户在网站的位置。

面包屑应用于URL的结构设计，有助于用户和搜索引擎的体验和收录。

例如：

### 1.7.2 前台浏览页面，不允许使用javascript做链接。

包括首页、2级、2.5级、3级、4级页面、图书各级页面、各类促销页面、以及其他不涉及用户个人信息且页面HTML一致的页面。

搜索引擎蜘蛛不能抓取JavaScript的link链接。

### 1.7.3 图片加alt注释，图片链接和超链接尽量使用绝对路径。

Alt内容，应跟图片展现的意义一致。

。

# 2. JSP规范

## 2.1 JSP页面中，应只有页面展示逻辑，不应有业务逻辑

## 2.2 不应出现HTML注释

例如：<!-- Start - JSPF File Name: JSTLEnvironmentSetup.jspf -->

## 2.3 多余的换行、空格、TAB应当被删除

## 2.4 import应指定具体类名，不应使用通配符\*

例如不能使用import java.lang.\* 应该使用 import java.lang.String

## 2.5 应当避免设计既显示表单又处理结果的页面

## 2.6 在jsp中避免代码重复。把要重复的功能放在一个包含的jsp、bean或标记扩展中，使得它能够被重用。

## 2.7 jsp bean应当永远不要去直接生成HTML

## 2.8 在jsp中应该避免使用out.println()来产生页面内容。

## 2.9 jsp层不应该直接访问数据，这包括JDBC数据库访问和EJB访问。

## 2.10 应尽量使用标签，避免使用jsp scriptlet。

尽量避免：

<% if (user.getRole() == "member")) { %>

Welcome, member!

<% }

%>

**尽量使用标签：**

<c:if test="member">

<c:out value=" Welcome, member!"/>

</c:if>

## 2.11 jsp bean中不应当包含大量的数据。

## 2.12 避免在jsp中进行异常处理

## 2.13 避免在jsp中定义方法和内部类。

## 2.14 应通过pageEncoding指定编码，此编码需要与jsp文件本身的编码一致

例如：

<%@ page pageEncoding=*"UTF-8"*%>

## 2.15 无需session的页面，应指定不生成session。

例如：<%@ page session=*"false"*%>

## 2.16 Form提交数据，尽可能使用POST，而避免使用GET方法（容易被人模拟）

避免使用GET方法，GET方法提交的数据会以明文显示在URL上，容易被模拟。POST方法提交的数据在URL上不显示。

## 2.17 关键信息保密，关键保密信息不允许直接显示在页面上。例如用户的证件号或者登录银行信息等

例如：138\*\*\*\*3122

## 2.18 单个JSP文件无特殊需要不得超过1000行，如超出需要注释写明原因。

如果一个JSP文件太大，可以采用拆分的方法，一个文件太大，不利于维护。

# 3. HTML规范

## 3.1正确处理缩进

通过合理缩进能使标签的嵌套关系更清晰，有利于维护。

## 3.2HTML标记必须使用小写

## 3.3每个标签后必须要闭合

例如<input />或是<input></input>，<a href=”#”></a>

## 3.4对于属性值，使用双引号包围

例如width=”100px”

## 3.5在W3C HTML4.0和XHTML1.0不可以使用以下HTML标签：

<applet><basefont><dir><font><isindex><menu><s><center><u><strike><xmp>

## 3.6<span><p><h1><div>的区别使用

* <span>，只用来修改一小部分文字的属性的时候，使用span，比如改变某小段自己的颜色
* <p>，整体的段落部分，适合大段文字，一般情况下不用在span和p中嵌入div
* <h1>,<h2>…<h6>，表示标题。一个页面内h1只可以用一次；
* <div>，一个块结构，块状元素

## 3.7<i><em><b><strong>的区别使用

* <i>，标签呈现斜体的文本
* <em>，把文本定义为斜体强调的内容，表示内容着重点(stess emphasis)
* <b>，标签定义粗体的文本
* <strong>，把文本定义为粗体语气更强的强调内容，表示内容的重要性(strong importance)

## 3.8<form>中必须加action属性，并且不能为空，如果不需要使用action属性也必须定义action=”no”

## 3.9<img>中的alt属性不可以缺少，来表示图像代替文本<img alt=””/>

## 3.10<head>和</head>之间必须有<title></title>

## 3.11<tr>、<td>必须定义在<table>之间

## 3.12<button>标签必须定义在<form>之间，否则会出现浏览器不支持的情况

## 3.13HTML的标准头部，只能在同一个页面出现一次：

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

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

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

<title>无标题文档</title>

</head>

## 3.14如果引用主站上服务器的资源，尽可能使用相对路径

## 3.15页头title、meta标签正确的顺序应该为：titile->description->keywords

## 3.16相同的css、js文件在同一个页面中只能被引入一次

# 4.JavaScript 规范

## 代码风格

### 4.1.1命名规则

* + 1. 函数命名：functionName
    2. 变量命名：variableName
    3. 类命名：ClassName
    4. 常量名：CONSTANT\_NAME

### 4.1.2缩进与大括号

如：

if (b) {

//前面是四个空格

}

else {

//前面是四个空格

}

### 4.1.3务必用var声明变量

如果尝试读一个未声明的变量的值,JavaScript会生成一个错误.如果尝试给一个未用var声明的变量赋值,JavaScript会隐式声明该变量.但要注意的,隐式声明的变量总是被创建为全局变量,即使该变量只在一个函数体内使用.局部变量是只在一个函数中使用,要防止在创建局部变量时创建全局变量(或采用已有的全局变量),就必须在函数体内使用var语句.无论是全局变量还是局部变量,最好都使用var语句创建.

function doSth() {

//变量a会理解成全局变量，在函数外面也可以调用

a = "AAA";

}

function doSth() {

//变量a会理解成局部变量，在函数外面不能调用

Var a = "AAA";

}

### 4.1.4尽量使用===/!==，必要时候使用==/!=

==和!=操作符会做类型强制转换。特别是，不要使用==来和“假”值（false, null, undefined, “”, 0, NaN）做比较。

===表示恒等于，比较的两边要绝对的相同   
运行体会一下以下的代码就会清楚了：   
alert(0 == ""); // true   
alert(0 == false); // true   
alert("" == false); // true   
  
alert(0 === ""); // false   
alert(0 === false); // false   
alert("" === false); // false

### 4.1.5不要省略末尾的分号

### 4.1.6不要使用with

With的使用方法如下，但有可以引起与局部变量名的冲突，例如name。

function Lakers() {

this.name = "kobe bryant";

this.age = "28";

this.gender = "boy";

}

var people=new Lakers();

with(people)

{

var str = "姓名：" + name + "<br>";

str += "年龄：" + age + "<br>";

str += "性别：" + gender;

document.write(str);

}

### 4.1.7不要在代码块中声明Function

不要：

if (x) {

function foo() {}

}

这样写的问题 foo可能与if块中其他定义

可以：

if (x) {

var foo = function() {}

}

### 4.1.8正确使用异常

1.抛出捕获 try {…} catch(exception) {…}

2.自定义异常

### 4.1.9不要使用for(elem in array)这种语法来循环数组，使用普通的for（与java中的概念是不一样的，在js中建议用普通for）

不要：

var aNames=["sun","bao","tao"]

for(var sName in aNames) {...}

可以：

var aNames=["sun","bao","tao"]

for(var i=0; i < aNames.length; i++) {...}

1）function printArray(arr) {

for (var key in arr) {

alert(arr[key]);

}

}

2）function printArray(arr) {

var l = arr.length;

for (var i = 0; i < l; i++) {

alert(arr[i]);

}

}

var a = new Array(10);

a[3] = 3;

//如果用1）的写法，只会循环1次；如果用2）的写法，会循环10次

printArray(a);

### 4.1.10不要使用多行的字符串常量

不要：

var TEST\_DATA = "AAAAAAAAAAAA\

BBBBBBBBBBBBB\

CCCCCCCCCCCCC"; // “\”为分行符

可以：

var TEST\_DATA\_A = "AAAAAAAAAAAA"

+ "BBBBBBBBBBBBB"

+ "CCCCCCCCCCCCC";

### 4.1.11不要使用Array、Object和Function的构造函数，而是使用它们的常量表达形式

1.不要：var arr = new Array(); var obj = new Object();

2.而是：var arr = []; var obj = {};

## 4.2 常见模式

### 4.2.1使用prototype扩展已有的类

Number.add(num)：作用，数字相加  
实现方法：Number.**prototype**.add = function(num){return(this+num);}  
试验：alert((3).add(15)) -> 显示 18

### 4.2.2可以传递function时候不要传递string（如：setTimeout时候）

function dolaert() {

alert("OK");

}

setTimeout("dolaert()"); //不建议

setTimeout(dolaert); //建议

setTimeout(function dolaert(){});//建议

### 4.2.3不要使用eval（反序列化JSON例外）

不同浏览器对eval处理是有差异的，尽量避免使用。

### 4.2.4不要使用+拼接大量字符串

可以使用数组方式

var buffer = [];

for(var i = 0; i < 2; i++) {

buffer.push("str");

}

document.writeln(buffer.join(""));

### 4.2.5尽量不要在HTML代码中内嵌JS

1.不要：<a onclick="doSomething()" href="#">Click!</a>

2.应该：<a href="backuplink.html" class="doSomething">Click!</a>然后在单独的JS文件中：$('a.doSomething').click(function(){ doSomething();});

### 4.2.6不要对基础类型使用封装对象

1.不要：new Boolean(false)

2.使用：false

3.备注：typeof false == ‘boolean’; typeof new Boolean(false) == ‘object’;

### 4.2.7注意closure（闭包）和DOM元素共用引起的内存泄露

1.Leak：

function foo(DOMElement, a, b) {

element.onclick = function() { /\* uses a and b \*/ };

} 构成了闭包.a,b在foo返回后在内存中也不会释放

2.OK：

function foo(DOMElement, a, b) {

element.onclick = (function(c, d) {

return function(){ /\* uses c and d \*/ };

})(a, b); a,b在foo返回后在内存中会释放

}

3.解释：之前的leak版本，DOM Element引用了匿名的function，匿名的function引用了foo，而foo又引用了DOM Element，形成了循环引用，所以closure里面的a, b包括foo，都是无法被GC的。闭包有可能引起内存泄露，使用时应注意。

### 4.2.8除非必要，不要修改内建对象的prototype，严禁修改Object和Array。

## 4.3 jQuery

### 4.3.1版本库的控制统一b2c整个项目版本号，如需升级版本需讨论后决定，任何人不得擅自升级版本。全站统一采用jquery.js 1.42版本。

### 4.3.2不要选择不必要的元素

比如用属性选择器选了很多元素，再过滤出自己用的。这样效率会低。

### 4.3.3能使用id选择器时候不要使用class选择器，能使用class选择器不要使用属性选择器

目的是为了尽量缩小选择的范围，提高效率。

### 4.3.4尽量使用jQuery的链式语法：

1. 不要：var e = $(‘#someId’); e.attr(‘class’, ‘someClass’); e.hide(); e.html(‘new content’);
2. 而是：$(‘#someId’).attr(‘class’, ‘someClass’).hide().html(‘new content’);
3. 更不要：$(‘#someId’).attr(‘class’, ‘someClass’); $(‘#someId’).hide(); $(‘#someId’).html(‘new content’);

### 4.3.5尽量使用find来查找层次结构的元素

1. 不要：$(‘#someOuter div.someInner’)
2. 而是：$(‘#someOuter’).find(‘div.someInner’)

### 4.3.6尽量使用原生DOM API

* + - 1. 不要：

$('#someDiv').click(function() {

alert($(this).attr('id'));

});

* + - 1. 而是：

$('#someDiv').click(function() {

alert(this.id);

});

* + - 1. 例外：存取属性src、href、style时候要使用jQuery。

## 4.4Ajax

### 4.4.1一般请求使用jQuery的get和post方法。

1. $.get("AJAXServer",{name:$("#userName").val(),test:"test123"},function(data){$("#result").html(data)});
2. $.post("AJAXServer",{name:$("#userName").val(),test:"test123"},function(data){$("#result").html(data)});

### 4.4.2GET获取JSON采用jQuery的json方法。

### 4.4.3GET获取HTML采用jQuery的load方法。

## 4.5浏览器端的MVC与Template

### 4.5.1采用JSON+Template优于GET获取HTML。

### 4.5.2采用jQuery的Template类库。

## 4.6js引用：基础模块放在 head 区域，扩展类或者专项类的可以放在尾部

## 4.7js压缩：所有js文件，在发布之前都要进行压缩

## 4.8js文件命名：根据项目命名，例如：图书馆为 book.js ；避免同一命名

# CSS规范

## 5.1引用

## 5.1.1所有外链引用的css文件，全部放在<head>区域

<link rel="stylesheet" type="text/css" href="css/common.css" />

//common.css中包含css reset、公用模块及整站公用头尾的样式内容

<link rel="stylesheet" type="text/css" href="css/index2011.css" />

//本页面相关的样式

## 5.2CSS书写要求

### 5.2.1公共组件需高度重用，命名从简，不要加前缀；

### 5.2.2各栏目的相应代码，需加前缀，例如：苏宁缀为“sn”，分隔符为下划线“-”

### 5.2.3模块组件化，组件中的class名采用“-”相结合的方式，单词之间的分隔靠大写字母分开，从属关系靠下划线分隔。

### 5.2.4id名采用驼峰命名的方法，方便后台调用。

### 5.2.5命名要有意义，不要使用没有意义的命名。用英语命名，不可用拼音。

## 5.3书写顺序：布局定位属性-->自身属性-->文本属性-->其他属性

.sn\_list {

//显示属性

display:; position:; float:; clear:; cursor:;

//盒模型

width:; height:; margin:; padding:;

//文字

color:; font:; content:; text-align:; text-decoration:; white-space:; vertical-align:;

//边框和背景(容易修改)

border:; background:;z-index:;

}

## 5.4所有css文件，在发布之前都要进行压缩