# 文档02 前段知识点归纳

## Web标准

不是某一个标准，而是一系列标准的集合。网页主要由三部分组成：**结构（Structure）、表现（Presentation）和行为（Behavior）**。对应的标准也分三方面：结构化标准语言主要包括XHTML和XML，表现标准语言主要包括CSS，行为标准主要包括对象模型（如W3C DOM）、ECMAScript等。

* 结构指的是HTML，通过HTML定义网页中包含的内容。
* 表现指的是CSS，通过CSS可以让HTML标签变得更具美感。
* 行为指的是JavaScript，通过JS的交互编程，让静态的网页动起来。

## HTML技术

**HTML语言是一种标记语言**，其中包含了很多的标记，不同的标记具有不同的作用和展示方式，通过不同的HTML标记可以构造出丰富多彩的网页。

**标记的使用：**

**标记名要求小写**，标记使用的时候需要配合<>一起使用，例如：<div></div>，其中div就是一个标记名称。

**标记的分类：**

**分类方法1：**根据标记的构成，**分为单标记和成对标记**。

单标记只有标记开始无标记的结束，例如:<br />，<hr />，<img src="路径" />

成对标记有开始标记和结束标记，例如:<div>html内容</div>

**分类方法2：**根据标记的展示方式，**分为行内元素，块元素，行内块元素**

行内元素默认一行可以摆放多个元素，块元素默认一行只能排放一个元素，要改变行内元素和块元素的默认展示方式，需要利用CSS的技术实现。

行内元素有：<a href="地址">超链接</a>; <span>html内容</span>

**标记的属性：**

标记上允许携带属性，属性要加在标记的开始位置，**属性名要求小写**，属性名和属性值之间通过=分割，属性值要加上引用，多个属性之间需要空格分割。

例如：<div id="top" class="top" >html内容</div>

有的属性是所有标签公用的

* id 标签标示，不允许重复
* class 为标签定义类选择符
* style 为标签定义CSS样式
* data-xxx 为标签附加额外的数据属性

**标记的嵌套：**

标记允许嵌套，在某个标记内部允许嵌入其它标记，标记和标记之间的嵌套关系，形成了一个倒树状结构，标记的嵌套标记一定要规范，不要出现交叉嵌套。

<div><p></div></p> , 这个是错误的例子。

如上的代码，则无法判断是div里边嵌套了p，还是p里边嵌套了div。

### html文档结构

浏览器中展示的网页就是一个html文档，html文档的代码结构如下：

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>网页标题</title>

</head>

<body>

<div>

</div>

</body>

</html>

第一行是文档代码规范声明，代表该文档内的标记语言所遵循的HTML规范版本。

一个html文档由一个html构成，内部包含了head和body两个部分。

head标记内包含页面相关的描述信息。

body标记内包含的内容会展示在浏览器展示窗口中。

### head中常用标记

1、定义浏览器展示的标题

<title>页面标题</title>

2、在页面内撰写css代码

<style type="text/css">

CSS代码

</style>

3、在页面内撰写JS代码，此标记也可以放在body内。

<script type="text/javascript">

JS代码

</script>

4、引入css代码

<link href="css文件路径" rel="stylesheet" type="text/css" />

5、引入JS代码

<script type="text/javascript" src="js文件路径"></script>

6、meta

meta标签提供关于HTML文档的元数据。元数据不会显示在页面上，但是对于机器是可读的。

1. 页面关键字定义，优化搜索引擎

<meta name="keywords" content="your tags" />

1. 页面描述

<meta name="description" content="150 words" />

1. 页面重定向和刷新

<meta http-equiv="refresh" content="0;url=" />

1. 移动设备

<meta name="viewport" content="width=device-width, initial-scale=1.0,maximum-scale=1.0, user-scalable=no" />

详细内容参考：https://www.w3school.com.cn/tags/tag\_meta.asp

### body中常用标记

1. div容器

块容器，内部包含HTML代码，常用来把其它HTML元素放在一个容器中，通过容器来实现HTML元素的布局和样式控制。

1. p容器

块容器，内部包含一段文本。

1. span容器

行内容器，内部包含HTML代码。将html元素放入容器内，完成元素的布局和样式控制。

1. 列表标记

<ul>

<li>列表项目</li>

<li>列表项目</li>

<li>列表项目</li>

<li>列表项目</li>

</ul>

默认展示为无需列表，通过CSS样式可以让其展示为有序列表。

常通过列表标签实现网页布局，布局的实现方法见CSS部分。

**列表嵌套**：可以在<li></li>内部再定义一个列表标记来实现列表的嵌套。

1. 表格标记

两行两列的表格

<table>

<thead>

<tr>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<tr>

<td>hjhjkhk</td>

<td></td>

</tr>

</tbody>

<tfoot>

<tr>

<td></td>

<td></td>

</tr>

</tfoot>

</table>

表头用thead，表体用tbody，表尾用tfoot，不同的位置用不同的标记方便选择符的定义。

1. 图片标记

<img **src="图片路径"** alt="提示文本" />

1. 超链接标记

<a href="超链接地址" target="\_blank">文本</a>

**target属性常用取值：**

\_blank：通过新开窗口显示超链接地址内容。

\_self：通过本窗口显示超链接地址内容。

***知识扩展（锚点）***

为超链接标记指定name属性后，即可在超链接所在位置创建一个"锚点"，"锚点名"就是name属性的名称，网页上的超链接除了用来打开一个新页面之外，还可以实现锚地的定位。

把光标定位到本页面内的锚点：

<a href="#锚点名">定位到本页内的锚点</a>

打开另一个页面，并把光标定位到锚点：

<a href="网址URL#锚点名">定位到本页内的锚点</a>

1. 浮动框架

<iframe frameborder="0" width="100" height="100" src="页面路径"></iframe>

实现网页嵌入

1. 标题h1~h6

用来定义不同级别的标题。从h1~h6

<h1>标题</h1>

1. 回车

<br />

1. 空格

&nbsp;

1. fieldset，legend

用来创建一个容器，并在容器的上边框上展示一个描述。

例如：

<fieldset>

<legend>描述内容</legend>

放在容器里边的内容

</fieldset>

例子：

<fieldset>

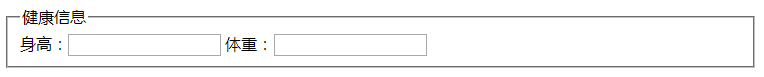
<legend>健康信息</legend>

身高：<input type="text" />

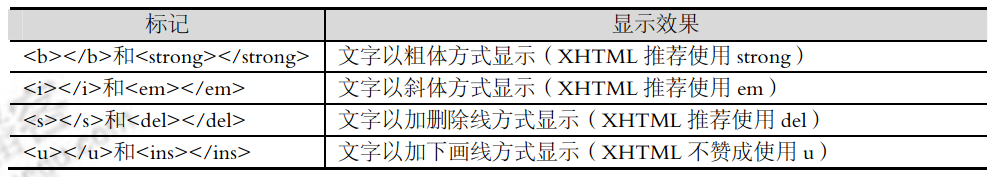
体重：<input type="text" />

</fieldset>

运行效果：



1. 文本格式化标记



### 表单标记

通过表单标记能够实现从客户端向服务器提交数据，表单标记分为表单容器和表单控件。表单控件需要放在表单容器内。

1. 表单容器

<form action=" Url" method="post">

表单控件

</form>

**action是表单数据的提交接收地址。**

**method是表单数据的提交数据的方式，主要有如下的方式：**

* get：把提交的数据附加在URL上，也称为显式提交，适合提交少量数据。

格式如下：url?变量=变量值&变量=变量值

* post：网址上不显示提交的数据，也称为隐式方式，适合提交大量数据。

表单默认采用Post方式提交数据，后端代码通过表单控件的name属性来区分不同的数据项。

1. input表单控件



* 单行文本框

<input type="text" name="userName" />

* 密码输入框

<input type="password" name="userPW" />

* 单选按钮

<input type="radio" name="sex" value="男" />

* 复选框

<input type="checkbox" name="love" value="读书" />

* 文件上传

<input type="file" name="headImg" />

* 隐藏域

<input type="hidden" name="stuID" value="sju" />

* 提交按钮

<input type="submit" value="提交" />

* 重置按钮

<input type="reset" value="清除" />

* 普通按钮

<input type="button" value="点击" />

1. Label标记

<label for="userName">姓名</label>

当光标点击文字"性别"时，光标自动定位到for属性所关联的表单控件上，for属性的属性值为表单控件的ID属性值。

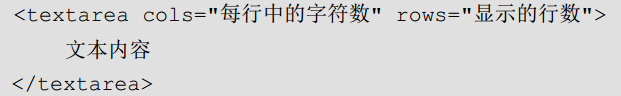
例如：

<label for="userName ">姓名</label>

<input type="text" name="userName" id="userName" />

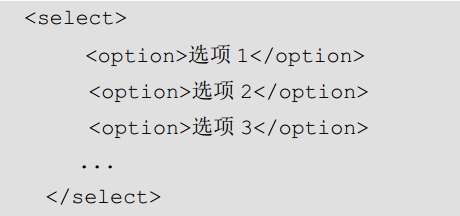
1. textarea控件

用来创建多行文本框控件。

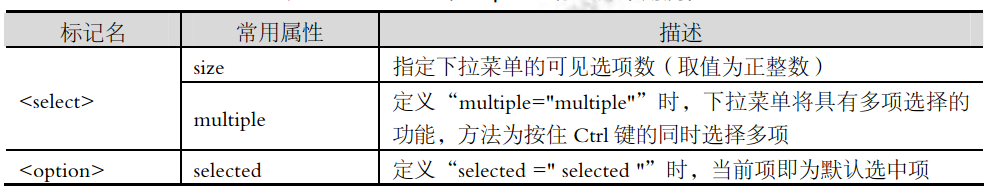


1. select控件

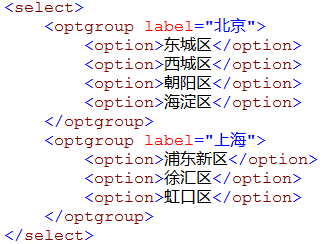
用来创建下拉框控件。



该标记常用属性如下：



在select标记内，可以添加optgroup标记来创建选项组，请参考如下的例子：



运行效果如下：



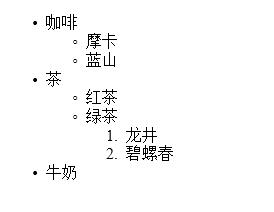
### 路径种类

Html文档中的文件路径分为多种，不同的路径有不同的路径规则，拥有不同的特点，需要理解和掌握不同的路径形式。

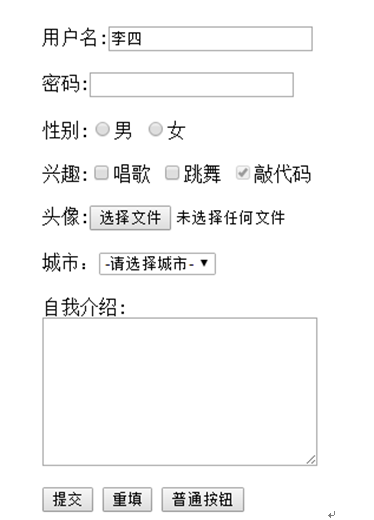
1. **相对路径**：相对路径的值由"路径所在文件"和"路径指向文件"这两个文件的相对位置计算出来的一种路径，开发工具中，选择文件时，默认采用相对路径，这种方式的路径，不同的文件引用同一个资源文件，其所用路径可能不同。
2. **绝对路径**：当路径的开始为协议号，或者盘符之类的，此类路径为绝对路径，一般用来引用站点外部资源时使用。
3. **跟路径**：跟路径是一种网站内部经常使用的路径（需要把网页文件部署在网站服务器内），网站内不建议用相对路径，建议采用跟路径，以"/"开头的路径是跟路径。

### 相关练习

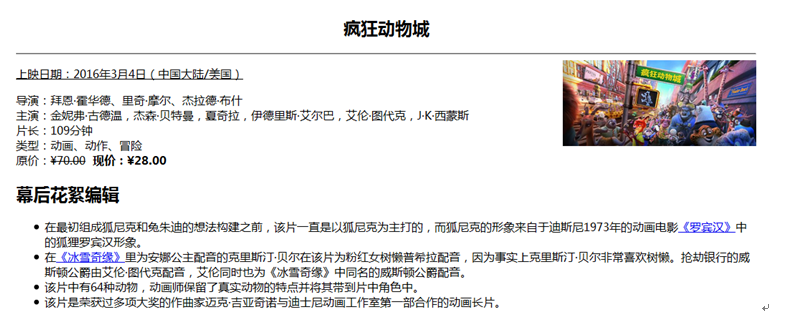
* 1. **完成如下的嵌套无需列表。**



* 1. **完成如下的表单**



* 1. **完成如下的页面排版(学习完CSS之后做)**



## CSS技术

### CSS简介

HTML标记用来完成网页内容的构建，这些内容如何展示主要靠CSS，CSS用来控制网页中标记的展现方式。

**CSS样式规则**：CSS的基本代码单元称为"样式规则"，"样式规则"的代码结构如下：

CSS选择符

{

样式属性:属性值;

样式属性:属性值;

}

大括号内部定义的是CSS属性及其属性值，大括号内定义的这些属性和属性值对那些标记的展示形式生效由大括号上方的"选择符"决定，CSS中都有那些常用的选择符，见CSS选择符。

### CSS基本选择符

1. **id选择符**

选择符名称为"#名称"，以"#"开头的选择符是id选择符，大括号内定义的样式对id属性等于"名称"的标记生效。

#top

{

color: red;

}

id选择符一个页面只能被使用一次，因为标记的id属性值不允许重复。

备注：上边的选择符对如下的div生效。

<div id="top">页面顶部区域</div>

1. **类选择符**

选择符名称为".类名"，以"."开头的选择符是类选择符，大括号内定义的样式对class属性等于"类名"的标记生效。

.nav

{

background-color: blue;

}

类选择符在一个页面中可以多次使用，比较灵活方便，同一个类选择符可以用到不同的标记上。

备注：上边的选择符对如下的div生效。

<div class="nav">页面导航区域</div>

1. **标记选择符**

选择符的名称等于某个标记的名称，大括号内定义的样式对页面内所有该标记生效。

div

{

font-size: 12pt;

}

1. **通用选择符**

选择符名称为"\*"，大括号内定义的样式对本网页内所有的标记生效。

\*

{

padding: 0px;

margin: 0px;

}

通用选择符经常用来定义页面内所有标记的公用属性。

备注：上边的代码是定义所有标记的内边距和外边距为0px，去除标记的默认边距，该行代码经常放在CSS代码的最前边。

### CSS复合选择符

由**两个以上的基本选择符**组合而成的选择符，称为复合选择符，常用的复合选择符有：

1. **包含选择符**

**结构：选择符1 选择符2**

HTML文档中的标记是一种嵌套的树状结构，为了能够选择某个子树内满足某个选择符的标记，需要使用到包含选择符。

例如：.top div，是由类选择符和标记选择符构成的复合选择符，首先".top"选择符会选择一部分标记，".top"后边的"div"选择符只能从".top"选择符选中的标记的所有子树内选择匹配的标记。

1. **并列选择符**

**结构：选择符1，选择符2，选择符3**

如果多个选择符{}内定义的样式是一样的，就不同通过多个样式规则来定义，只需要把多个选择符放在一起，多个选择符之间用","分割即可。

例如：

a:link,a:hover,a:visited,a:active

{

font-size: 12pt;

text-decoration: none;

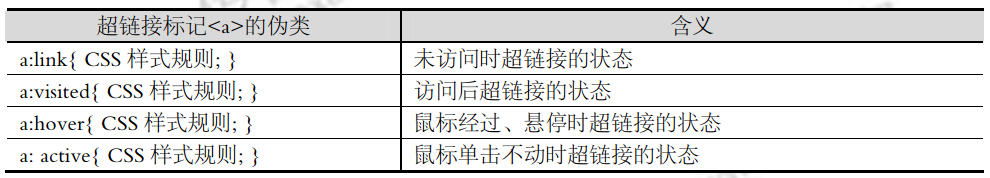
}

1. **链接伪类**

**结构：标签选择符:状态**

同一个标签，根据其不同的种状态，有不同的样式。这就叫做"伪类"。伪类用冒号来表示，这里只介绍超链接的魏磊选择符。

超链接标记拥有不同的状态，如果要对不同的状态定义样式，则需要使用伪类。



1. **标签指定式**

**结构1：标签选择符.类选择符**

**结构2：标签选择符ID选择符**

标签选择符另一个选择符，两个选择符之间没有空格，例如：div.Header，选择的元素为拥有class为"Header"的div元素。

### CSS选择符优先级

在使用CSS选择符的过程中，会出现不同的选择符选中相同标记的情况，那该标记应该使用谁的选择符哪？

在CSS中为不同的选择符定义了不同的优先级别，如果同一个标记有多个选择符选中，并且不同选择符内定义了同一个样式属性，那该标记会应用优先级高的那个选择符内的样式属性。

例如：

.nav

{

background-color: blue;

font-size: 12pt;

}

div

{

font-size: 14pt;

color: red;

}

上边的两个选择符，类选择符的优先级比标签选择符的优先级高，如果同时选中了一个div标记，则div最终的样式为：

{

background-color: blue;

font-size: 12pt;

color: red;

}

由于两个选择符内都有"font-size"这个属性，谁的优先级高用谁的，"color"属性，高优先级的里边没有，则用的底优先级内的。

CSS中针对不同的基本选择符，定义了优先级权重，权重大的，优先级就高。

**ID选择符：100**

**类选择符：10**

**标签选择符：1**

**\*选择符：0**

注意的是，权重值不是10进制的，不同浏览器采用的权重进制不同。

复合选择符是由多个基本选择符构成的，因此复合选择符的权重是多个基本选择符的权重和的累计，例如：.top div 的权重为11。

### CSS代码的使用方式

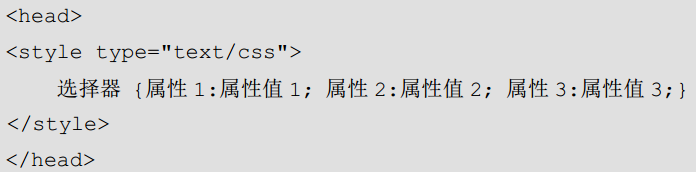
CSS代码提供了多种使用方式，具体方式如下：

1. **行内式**



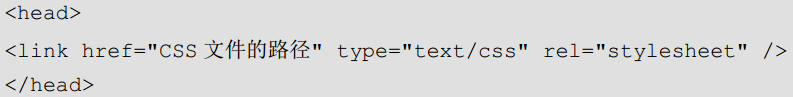
直接在标记上定义样式属性。

1. **内嵌式**



在页面的Head标记内插入<style></style>标记，并在标记内撰写一个个CSS样式规则，撰写的样式规则对本页内所有的标签生效。

1. **外链式**



在单独的CSS文件中撰写一个个样式规则，并在Head标记中把文件引入，定义的样式规则对本页内所有的标签生效。

### CSS常用属性

#### 字体相关

1. **font-family: "微软雅黑"; 字体类型**
2. **font-size: 12pt; 字体大小**

***知识扩展（长度单位）***

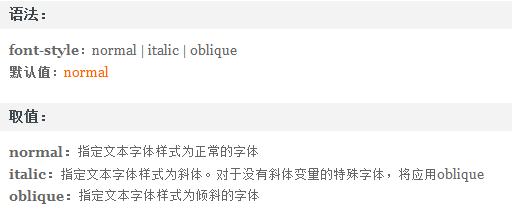
CSS中定义了多种长度单位，分为绝对长度单位和相对长度单位。

绝对长度单位主要有：cm(厘米)，mm(毫米)，in(英寸)，pt(点)

1in = 2.54cm = 25.4 mm = 72pt = 96px

相对长度单位有：px(像素)，em(一个文字的宽度大小)，ex(一个文字高度的一半)

1. **font-weight: 400; 字体加粗**
2. **font-style: inherit; 字体样式**

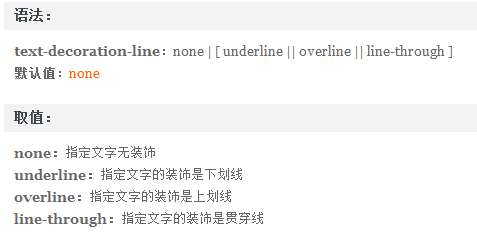


1. **color: red; 字体颜色**

可以使用Color Name(颜色名称), RGB, transparent来指定color。

1. **text-decoration: none; 字体装饰**









#### 文字相关

1. **text-align: center; 文本水平对齐**



1. **vertical-align: middle; 文本垂直对齐**



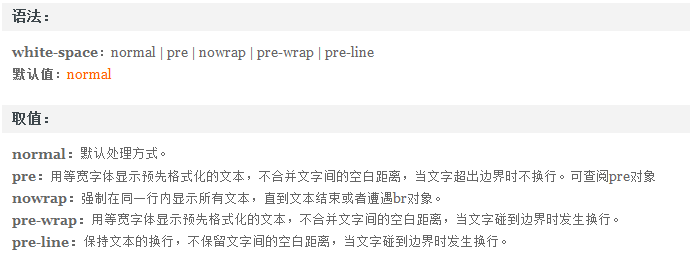
1. **line-height: 3ex; 文本行高**

* 备注：*当单行文本时，设置行高等于容器的高度时，可以让文字垂直居中，*"3ex"就是1.5倍行高。

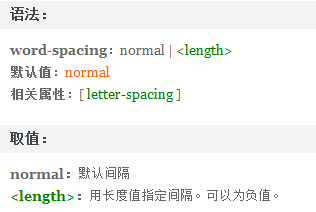
1. **text-indent: 2em; 文字首行缩进**

* 备注："2em"就是首行缩进两个字体。

1. **white-space:normal; 文本空格处理方式**



1. **word-spacing: normal; 字间距**



1. **letter-spacing: initial; 单词间距**



#### 背景相关

1. **background-color:#FF0000; 背景颜色**

可以使用Color Name(颜色名称), RGB, transparent来指定color。

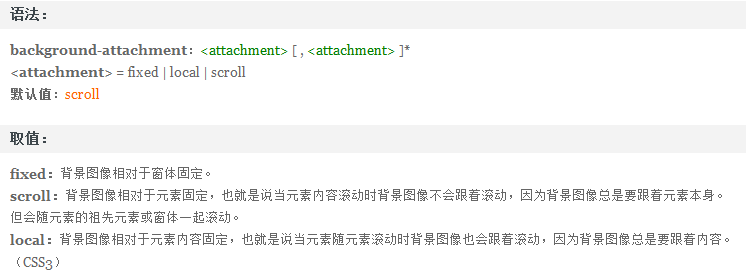
1. **background-image: url(img/HBuilder.png); 背景图片**



1. **background-repeat: repeat-x; 背景平铺方式**



1. **background-attachment: scroll; 背景附着方式**



1. **background-position: 0px 0px; 背景位置**

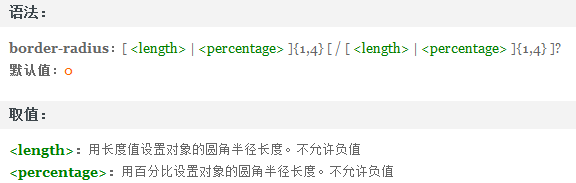


#### 边框相关

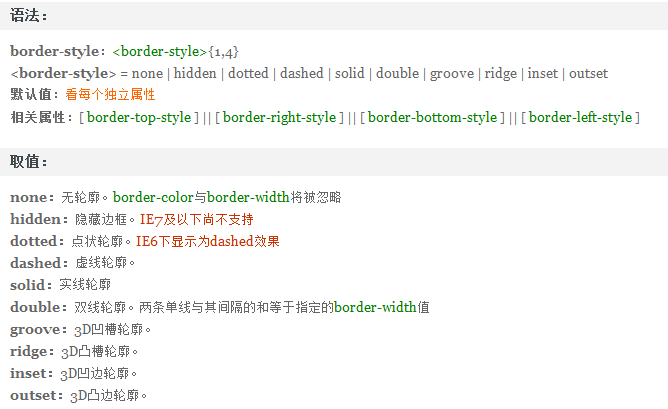
1. **border-color: #FF0000; 边框颜色**



1. **border-radius: 10px; 边框圆角**



1. **border-style: dotted; 边框样式**

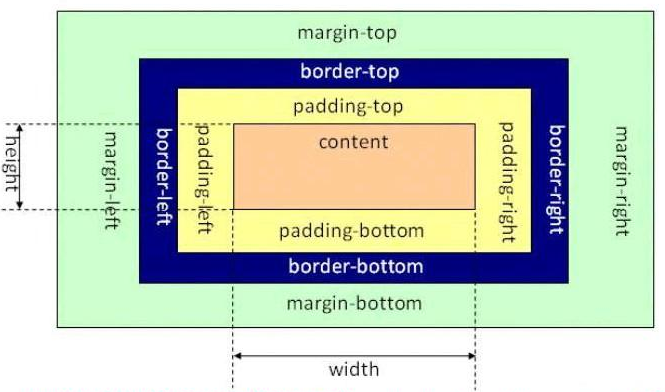


1. **border-width: 1pt; 边框宽度**

#### 方框相关

1. **padding: 2px; 内边距**
2. **margin: 1px; 外边距**

***知识扩展（盒子模型）***



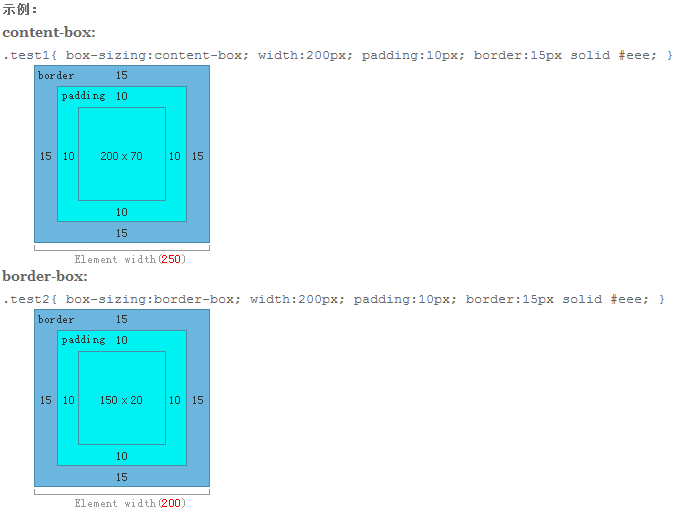
width和height属性默认指的是其内容的尺寸，加上内边距，边框宽度之后，才是其在网页中所占的尺寸。

1. **box-sizing: content-box; 盒子模型尺寸规则**



默认情况下元素的width和height属性设置的是元素内容的尺寸，元素在浏览器中所占的尺寸还需要加上内边距和边框宽度。

如果要设置元素的所占尺寸等于width和height所设数值，需要把box-sizing属性设置为“border-box”属性值。



#### 定位相关

1. **width: 100px; 元素宽度**
2. **height: 200px; 元素高度**
3. **position: absolute; 元素定位方式**



1. **top: 200px; 离参考物顶边的距离**
2. **left: 100px; 离参考物左边的距离**
3. **right: 100px; 离参考物右边的距离**
4. **bottom: 100px; 离参考物底部的距离**
5. **z-index: 2; 元素在Z轴的层高**

***知识扩展（盒子模型）***

定位方式介绍：

**static(流定位)：**默认定位方式，元素按照从左到右, 从上到下的规则来摆放元素，流定位的元素不会出现叠加的现象。

流定位方式支持通过text-align；margin-left:auto，margin-right:auto来设置居中，支持float,clear来设置元素浮动与清除，绝对定位方式不支持。

**absolute(绝对定位)：**元素在网页上的位置通过width，height， top，left四个属性的值来决定，top和left的默认参考物是浏览器窗口。

**relative(相对定位)：**相当于流定位方式下所在位置进行的位置偏移，原位置无法再摆放其它元素，偏移的大小通过top和left属性来设置，参考位置是流定位方式下所在的位置，相对定位主要是用来为绝对定位元素做参考物，当绝对定位的元素放在相对定位元素内的时候，绝对定位元素的参考物就变成了离其最近的相对定位的元素。

**fixed(固定定位)：**绝对定位的一种特例，偏移一定是相对于浏览器窗口，依靠top和left来实现相对于浏览器左上角进行定位，依靠right和bottom来实现相对于浏览器右上角进行定位。

z-index：当多个绝对定位元素叠加在一起的时候，用来设置元素在Z轴上的位置，值越大，越靠上。

#### 布局相关

1. **float: left; 元素浮动**



块元素默认一行只能展示一个元素，为了能够让多个块元素展示在一行，可以通过float属性，最左边的元素也需要浮动。

如果某个容器里边的子元素浮动了，一般情况下要求里边所有子元素都浮动起来，否则会出现浮动元素把不浮动元素盖住的现象（不浮动元素还遵循流定位）。

1. **clear: both; 元素浮动清除**



1. **display: block; 元素展示方式**



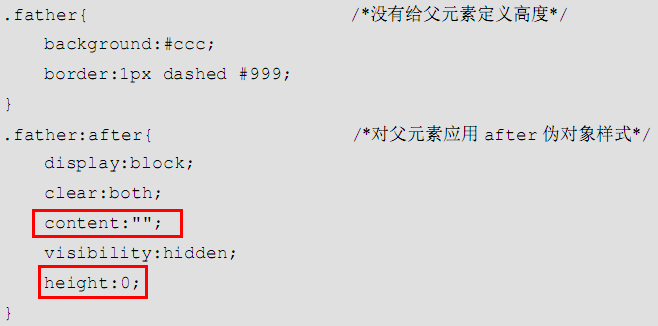
1. **overflow: hidden; 内容溢出处理方式**

该属性主要用来控制内容超出元素后的处理方式，但是它还可以用来解决子元素浮动对父元素的影响（overflow:hidden），当子元素浮动的时候让父元素的高度随着子元素的高度一起变化。



1. **: after; 解决内容浮动对元素的影响**

如果某元素内的子元素浮动了，父元素的高度不会随着子元素高度变化，我们可以通过overflow:hidden这个方式来解决，但是这种方式有一个问题，如果元素内部存在绝对定位的元素，并且绝对定位元素会超出父元素，用overflow:hidden这种方式，超出的内容就会被隐藏，这个时候如果想让绝对定位的元素超出部分能够显示出来，就需要使用:after，实例如下：



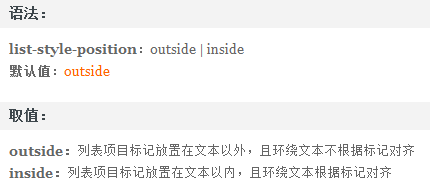
content属性和height必须按照如图所示使用。

#### 列表相关

1. **list-style-type: none; 列表符号类型**



1. **list-style-image: url(img/HBuilder.png); 列表符号图片**
2. **list-style-position: outside; 列表符号位置**

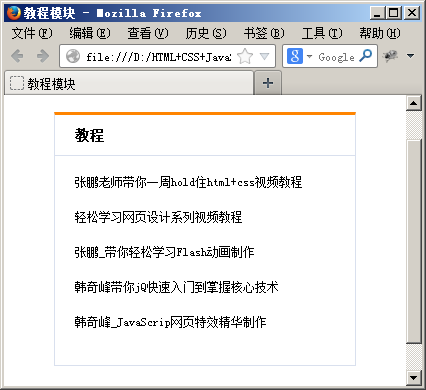


### 相关练习

1. **完成如下效果的对联**



1. **完成如下效果的制作**



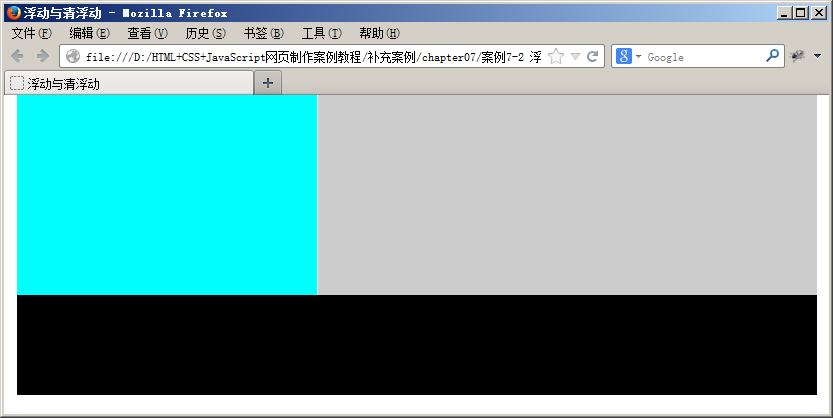
1. **完成如下效果的制作**



1. **完成如下效果的制作**



1. **完成如下效果的制作**



## JavaScript技术

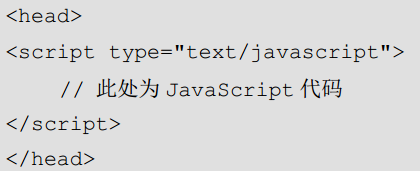
### JS简介

JS是Web前段一种功能强大的编程语言，用于开发交互式的Web页面，它不需要进行编译，而是直接嵌入在HTML页面中，把静态页面转变为支持用户交互病响应时间的动态页面。

### JS代码的使用方式

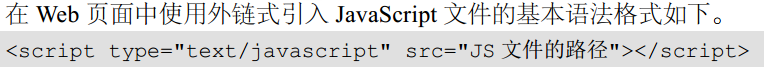
JavaScript提供了多种使用方式，具体使用方式如下：

1. **内嵌式**



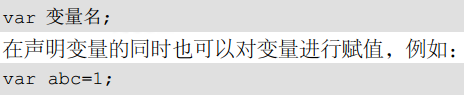
在Head标记或者Body标记内插入<script>标记，并在标记内编写合法的JS代码。

1. **外链式**



### JS的变量声明与赋值

JavaScript中变量在使用之前，必须进行声明，所有的变量都是通过var声明，语法格式如下：



如果只声明了变量，未对其赋值，则其默认为undefined。

可以使用一个关键字var同时声明多个变量，只需用逗号"，"分割变量名即可。



可以在声明的同时对其赋值，即初始化：



### JS的基本数据类型

1. **Number类型**

Number类型包含整数和浮点数（浮点数数值必须包含一个小数点，且小数点后面至少有一位数字）两种值。

JavaScript中提供了parseFloat() 和 parseInt() 等方法用来把其它类型的值转换为数值类型，并且提供了isNaN() 函数用于检查其参数是否是非数字值。

isNaN(123)   //false

isNaN("hello")   //true

1. **String类型**

在JavaScript中，字符串和字符类型都成为字符串型。

字符串对象有length属性来获取字符串的长度。

1. **Boolean类型**

该类型只有两个值，true和false

1. **Undefined类型**

只有一个值，即undefined值。使用var声明了变量，但未给变量初始化值，那么这个变量的值就是undefined。

1. **Null类型**

null类型被看做空对象指针，前文说到null类型也是空的对象引用。

1. **JS转义字符：**



### JS的引用数据类型

1. **Object类型**

我们看到的大多数类型值都是Object类型的实例，创建Object实例的方式有两种。

第一种是使用new操作符后跟Object构造函数，如下所示

var person = new Object();

person.name = "Micheal";

person.age = 24;

第二种方式是使用对象字面量表示法，如下所示

var person = {

　　name : "Micheal",

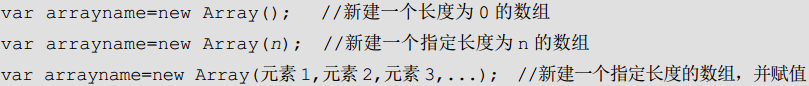
　　age : 24

};

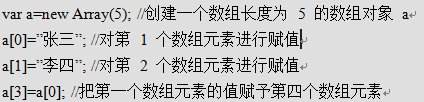
1. **Array类型**

数组的每一项可以用来保存任何类型的数据，也就是说，可以用数组的第一个位置来保存字符串，第二个位置保存数值，第三个位置保存对象，另外，数组的大小是可以动态调整的。

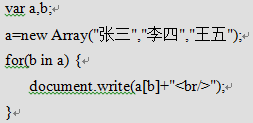
在JavaScript中，使用内置对象类Array可以创建数组对象，其语法格式如下：



使用举例：



使用举例：



数组对象拥有length属性，能够返回数组元素的个数，即数组长度。

数据对象拥有.toString()方法，该方法能够返回一个字符串，字符串内包含数组中的所有元素，各个元素用逗号隔开。

二维数组的使用例子：

var students,i,j;

students=new Array();

students[0]=new Array("张三","18 岁","男");

students[1]=new Array("李四","17 岁","男");

students[2]=new Array("王五","18 岁","女");

for(i in students){

for(j in students[i]){

document.write(students[i][j] +"\t");

}

document.write("<br/>");

}

1. **function类型**

每个函数都是Function类型的实例，而且都与其他引用类型一样具有属性和方法。函数通常是使用函数声明语法定义的，如下所示

function sum(num1,num2){

　　return num1 + num2;

};

这和使用函数表达式定义函数的方式相差无几。

var sun = function (sum1,sum2){

　　return sum1 + sum2;

};

1. **typeof 操作符**

由于js中的变量是松散类型的，所以它提供了一种检测当前变量的数据类型的方法，也就是typeof关键字.

typeof (123)　　 //Number

typeof('abc')　　 //String

typeof( true)       //Boolean

typeof( undefined) //Undefined

typeof( null)         //Object

typeof( { } )          //Object

typeof( [ ] )           //Object

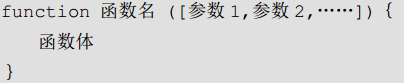
typeof( console.log())  //Function

null类型进行typeof操作符后，结果是object，原因在于，null类型被当做一个空对象引用。

### JS函数定义与调用

1. 函数的定义

JavaScript函数是由多条语句组成的逻辑单元，在JavaScript中，函数使用关键字function来定义，其语法格式如下所示：



1. 函数的调用

函数定义后不会自动执行，需要在特定的位置调用，函数调用的语法格式如下：



1. 函数中变量的作用域

函数中的变量需要先定义后使用，变量需要在它的作用范围内才可以使用。不同作用域决定了变量的类型。

* 全局变量：定义在所有函数之外，作用于整个程序的变量。
* 局部变量：

### JS的常用函数

1. **prompt()函数**

prompt()方法是 JavaScript 中窗口 window 对象的一个常用方法，用于显示和提示用户输入信息的对话框，其语法格式如下。



如果用户单击提示框中的取消按钮，则返回 null。单击确认按钮，则返回输入字段当前显示的文本。

1. **alert函数**

alert()函数主要用于弹出警示对话框，通常用于对用户进行提示，其语法格式如下。



alert()括号内的文本信息用于显示在警示对话框中，该对话框还包含一个"确认"按钮，单击该按钮可以关闭对话框。

1. **console.log()函数**

console.log 方法用于标准输出流的输出，即在控制台中显示输出结果，JavaScript程序开发过程中，输出调试信息时经常用到此函数。

1. **confirm()函数**

用来弹出一个确认框，当单击"确认"后，返回true，否则返回false，当某个操作需要用户确认后方可执行时，需要用到此函数。

1. **eval()函数**

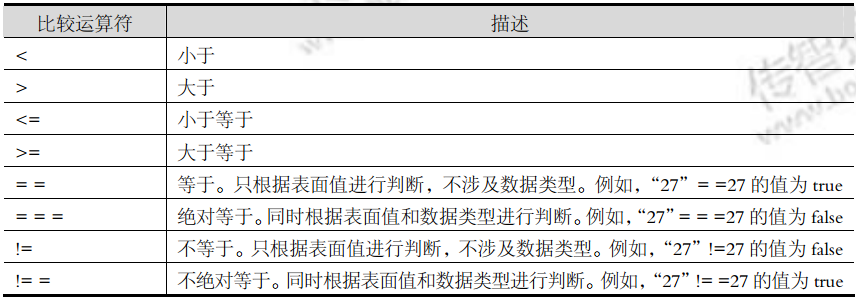
用来计算某个字符串表达式的值，例如eval(‘2+3’)计算的结果就是5。

### JS的运算符

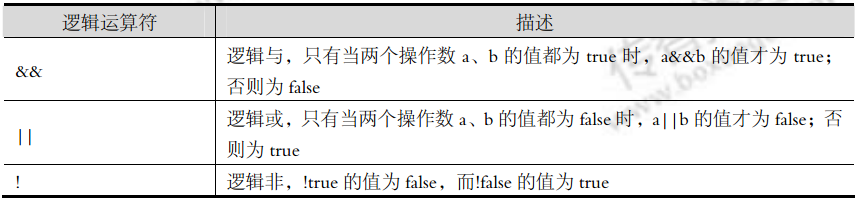
1. **算数运算符**



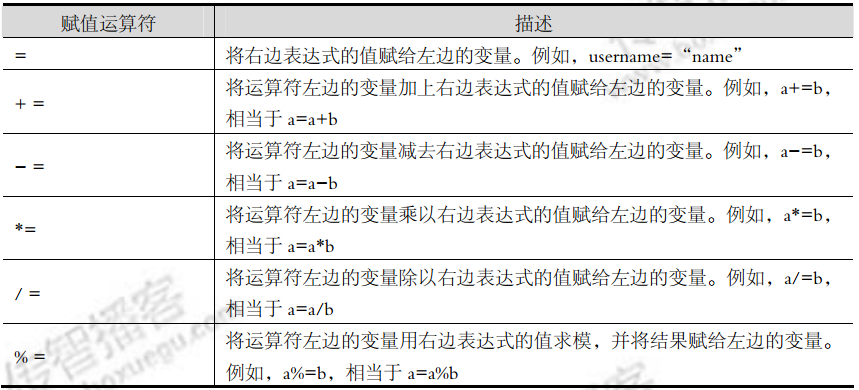
1. **比较运算符**



1. **逻辑运算符**



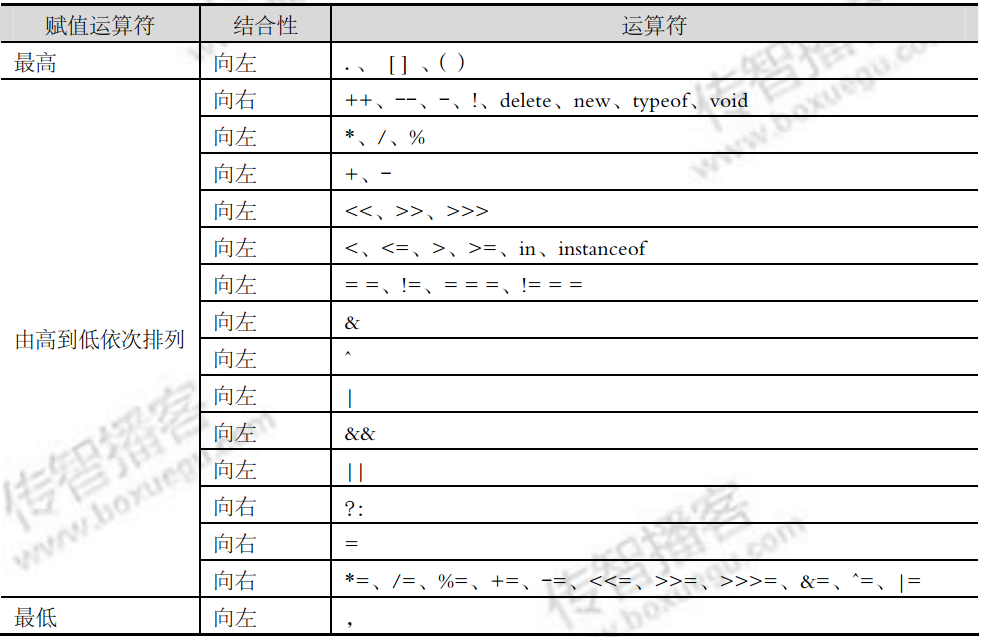
1. **赋值运算符**



1. **条件运算符**



1. **运算符优先级**



### JS表达式

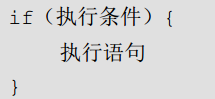
1. **表达式**

表达式是一个语句集合，像一个组一样，计算结果是一个单一的值，该值可以是 boolean、

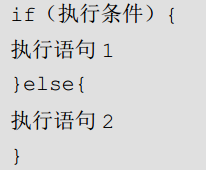
number、 string、 function 或者 object 数据类型之一。一个表达式本身可以很简单，如一个数字或者变量。另外，它还可以包含许多连接在一起的变量关键字以及运算符。

1. **if条件语句**

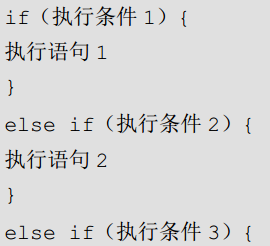
单向判断语句

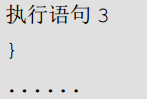


双向判断语句

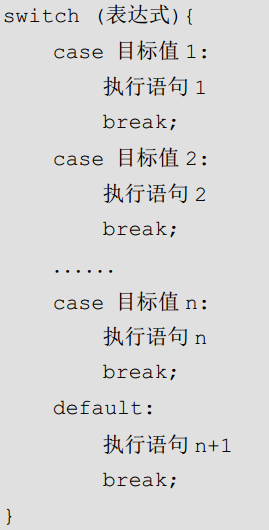


多向判断语句

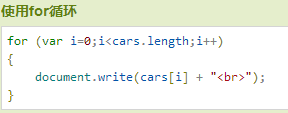


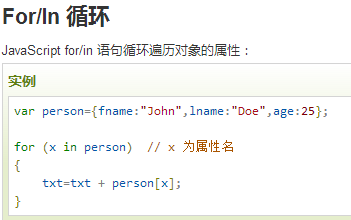


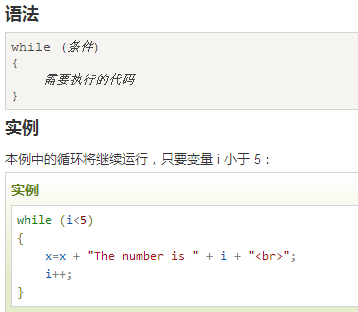
1. **switch语句**

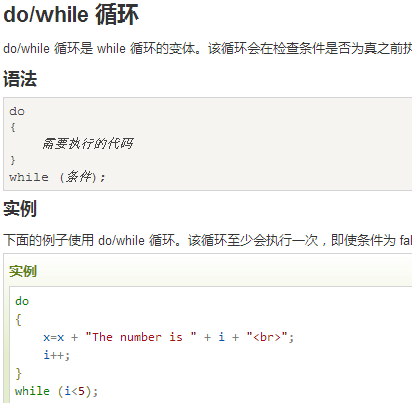


1. **循环语句**

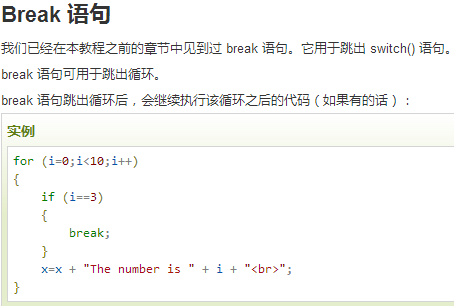


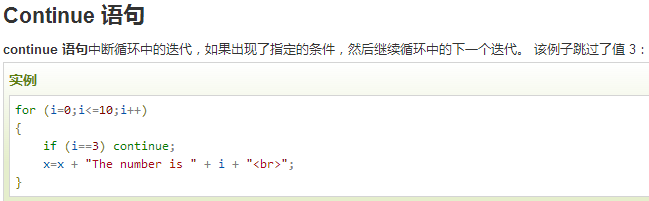






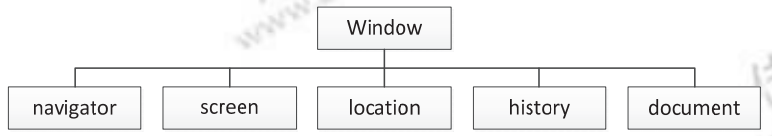
1. **循环控制**



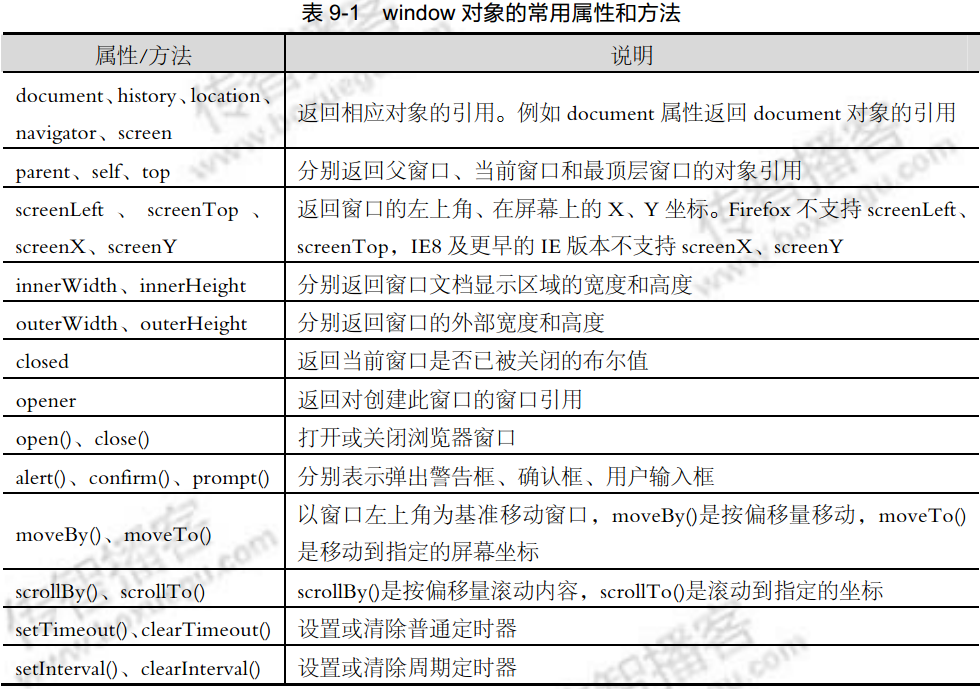


### JS的常用Bom对象

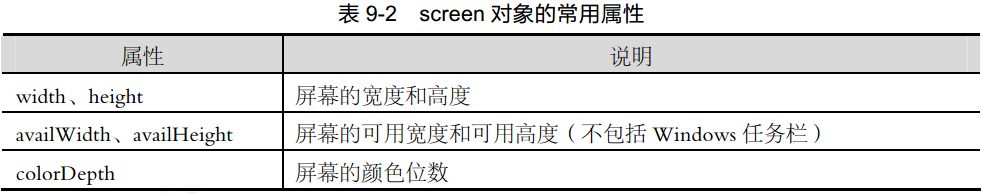
BOM（Browser Object Model）是浏览器对象模型，它提供了一系列对象用于与浏览器窗  
口进行交互。 BOM 对象包括 window（窗口） 、 navigator（浏览器程序） 、 screen（屏幕） 、 location（地址） 、 history（历史）和 document（文档）等对象。其中， window 对象是浏览器的窗口，它是整个 BOM 的核心，位于 BOM 对象的最顶层。



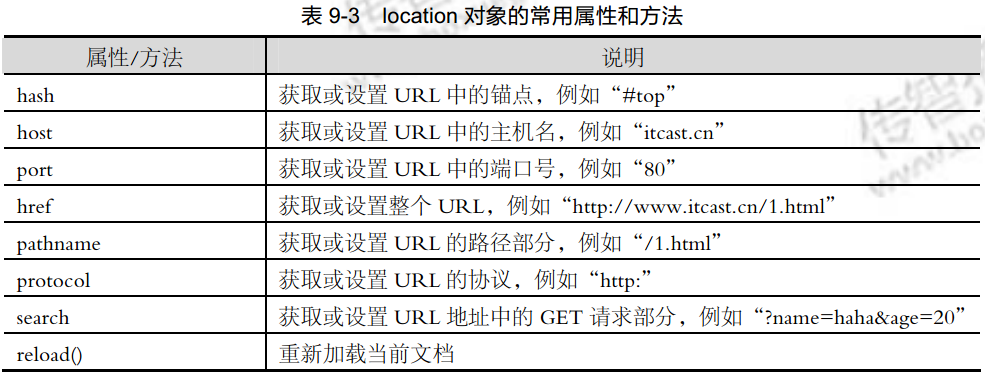
1. **windows对象**



1. **screen对象**



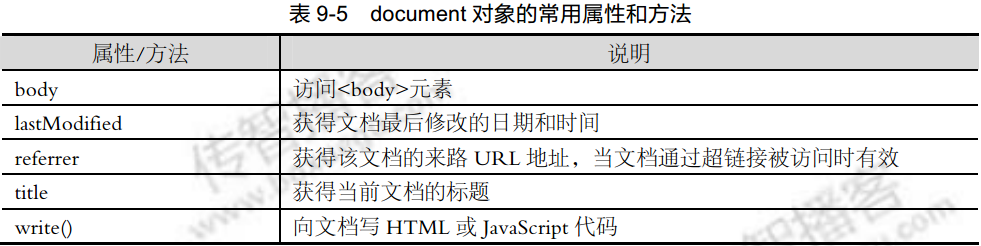
1. **location对象**



1. **history对象**

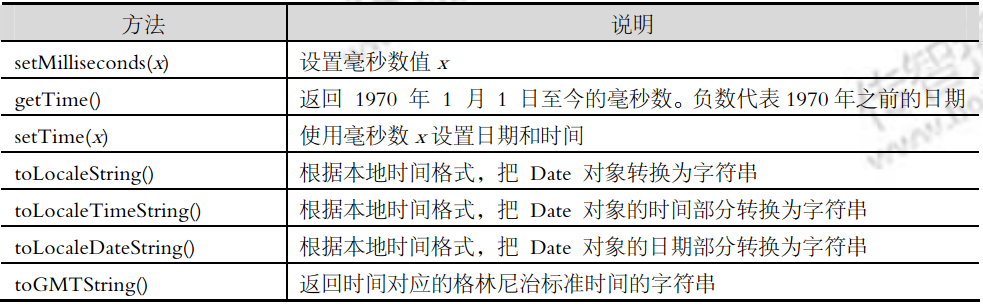


1. **document对象**



1. **date对象**





### JS的事件

用户与网页交互时产生的操作称为事件。事件可以由用户引发，也可能是页面发生改变，甚至还有用户看不到的事件，如AJAX交互进度改变。绝大多数事件都是由用户的动作所引发。

JavaScript语言是一种基于事件的开发语言，在页面内编写的JavaScript代码，当JavaScript代码和HTML的某个事件建立关联后，就可以通过HTML的事件来触发相关代码的执行。

常见的JavaScript事件有：

onLoad ：当页面加载完毕后触发，常用于body元素

onUnload ：当页面关闭后触发，常用于body元素

onBlur ：当失去焦点时触发，常用于input表单元素

onFocus ：当获得焦点时触发，常用于input表单元素

onClick ：当点击时触发，可以用于任何元素

onMouseOver ：当鼠标悬浮时触发

onMouseOut ：当鼠标离开时触发

onMouseDown ：当鼠标按下时触发

onMouseUp ：当鼠标弹起时触发

onMouseMove ：当鼠标移动时触发

onChange ：当状态改变时触发，常用于select下拉选框

onSelect ：当文本框中的文本选中时触发

onkeypress ：当键盘按下时触发（要快于onkeydown）

onkeydown ：当键盘按下时触发（可能捕获功能键，如上下左右）

onkeyup ：当键盘弹起时触发

onSubmit ：当表单提交时触发，常用于form表单元素

onReset ：当表单重置时触发，常用于form表单元素

JavaScript代码除了可以通过事件来调用之外，还存在其它的执行方式，写在页面内的JavaScript代码，如果这些代码未放在某个函数内，则页面解析过程中，解析到这些代码的时候就会被执行。

备注：function内的代码不会被解析执行，一定要通过函数调用才可以执行。

### 相关练习

1. 定义一个函数，函数内定义一个数组，用来保存江苏省内所有的城市，并通过循环输出城市名称，页面加载的时候，执行该函数内的代码。
2. 在页面上增加一个开始和结束按钮，单击"开始"后，能够创建一个定时器，每隔5秒钟，输出当前时间，当点击"结束"后，能够终止。
3. 在页面上添加一个按钮，单击后，能够弹出一个输入框，输入内容后，能够把输入的值弹窗输出。

## jQuery技术

系统学习参考书：《锋利的jQuery（第2版）》，单东林著，人民邮电出版社，2012年

API参考手册：<http://jquery.cuishifeng.cn/>

### jQuery简介

jQuery是一个优秀的JavaScript库，jQuery凭借简介的语法和跨平台的兼容性，极大地简化了JavaScript开发人员遍历HTML文档，操作DOM，处理事件，执行动画和开发AJAX的操作，其独特而又优雅的代码风格改变了JavaScript程序员的设计思路和编写程序的方式。

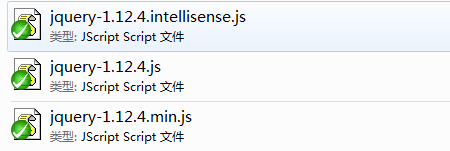
jQuery的优势

1. 轻量化
2. 强大的选择器
3. 出色的DOM操作的封装
4. 可靠的事件处理机制
5. 完善的AJAX
6. 出色的浏览器兼容性
7. 链式操作方式
8. 行为层和结构层的分离
9. 丰富的插件支持
10. 完善的文档
11. 开源

### jQuery开发库

jQuery是以JavaScript库的形式存在，要进行jQuery开发，只需要下载对应版本的jQuery库文件，并且在HTML页面中引入jQuery库文件，就可以采用jQuery的代码风格和代码语法进行代码的编写。

jQuery提供了不同的库格式，有源代码格式的，有压缩格式的，开发过程中建议引入源代码格式的，打包发布给客户的时候，建议换成压缩格式的。



在页面中通过script标签，引入jQuery脚本库之后，即可编写jQuery代码，jQuery代码一定要写在jQuery库引入代码的下方。

<script type="text/javascript" src="/scripts/jquery-1.12.4.js">

</script>

<script type="text/javascript">

编写jQuery代码

</script>

jQuery中的变量定义，函数定义，函数调用，运算符，表达式等内容的语法规则和JavaScript是一样的，只是其中多了一些jQuery的代码。

### jQuery代码风格

jQuery中有一个特殊符号$，其是jQuery的简写，$("#foo")，等价于jQuery("#foo")。

jQuery中的代码基本上以$(选择符).xxx(参数)的形式出现，$()内部是一个选择符，代表要操作的HTML元素，选择符后边携带一个"xxx"，xxx可能是一个方法，也可能是一个事件。

例如：$("div").hide()，这行代码的作用就是让页面内所有的div元素隐藏。

例如：$("div").click(function(){ 代码 })，这样代码的意思是给页面内所有的div元素绑定一个click事件，当click事件触发的时候，会执行function函数内的代码。

如果要想让jQuery代码在页面加载完毕之后才执行，需要把撰写的jQuery代码放入如下的结构内：

$(document).ready(function () {

//编写页面加载完毕之后才执行的代码，类似Javascript的load事件

});

$(document)代表选择本页的HTML文档，ready是页面加载完毕之后触发的事件，事件触发后，会执行参数内的function代码。

jQuery中提供了另一种简化的写法：

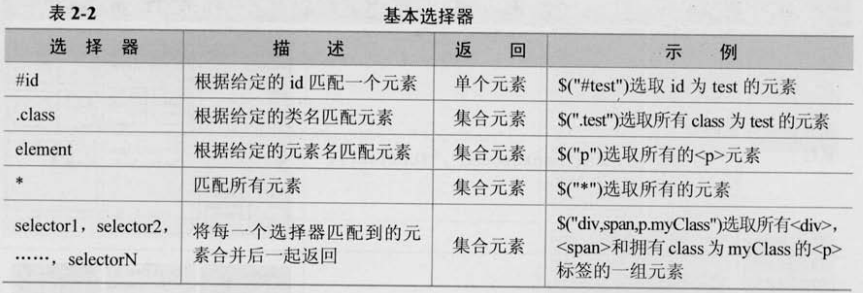
$(function () {

//编写页面加载完毕之后才执行的代码，类似Javascript的load事件

});

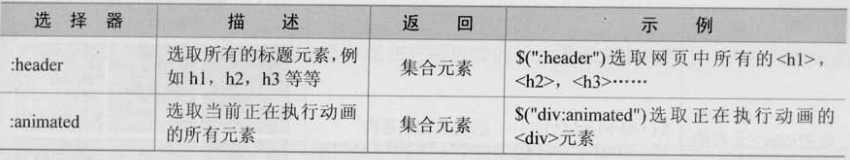
### jQuery提供的选择器

jQuery的选择器特别丰富，通过jQuery选择器能够很方便的进行网页元素的选择。





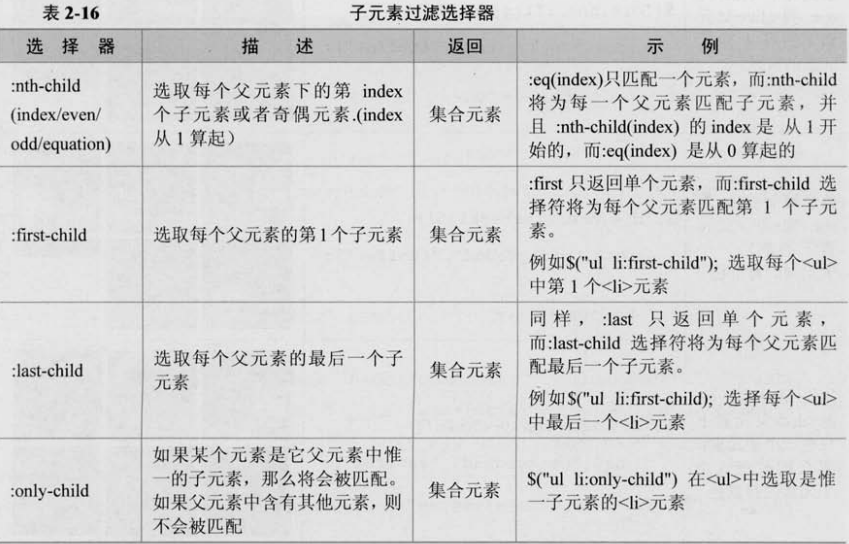




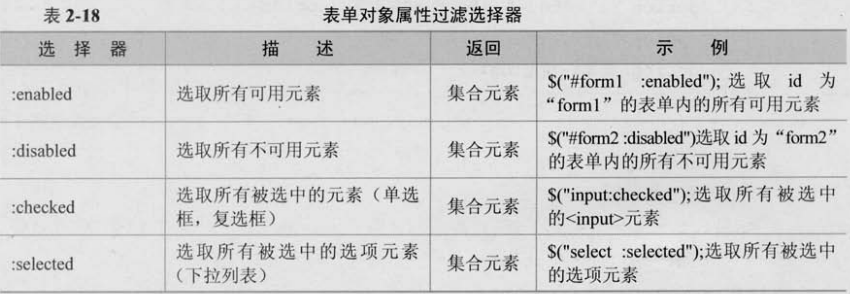








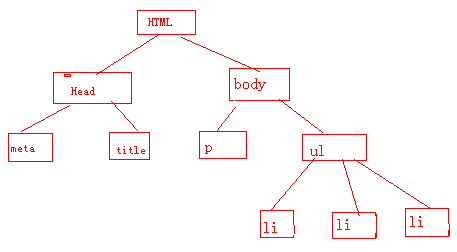




### jQuery的Dom操作

Dom是Document Object Model的缩写，意思是文档对象模型。

每一个网页都可以用DOM表示出来，每个DOM都可以看作是一棵DOM树。



Dom操作就是对HTML文档内容进行的操作，包括：**查找元素，创建与添加元素，移除元素，包裹元素，清空元素内容，为元素添加属性值，修改元素的属性，为元素附加或移除样式，获取元素内部的Html内容，获取元素内的文本内容，获取表单控件的值**等等操作。

#### 查找DOM节点

查找节点非常容易，使用选择器就能轻松完成各种查找工作。

例：查找元素节点p返回p内的文本内容：$("p").text();

例：查找元素节点p返回p的属性title的属性值：$("p").attr("title");

#### 新建DOM节点

创建元素节点并且把节点作为<ul>元素的子节点添加到DOM节点树上。先创建元素点，创建元素节点使用Jquery的工厂函数$()来完成，格式如下：$(html),该方法会根据传入的html字符串返回一个DOM对象，并将DOM对象包装成一个jQuery对象后返回。创建一个元素节点jQuery代码如下：

$li1=$("<li title='榴莲'>榴莲</li>")

代码返回$li1就是一个由DOM对象包装成的jQuery对象。把新建节点添加到DOM树中jQuery代码如下：

$("ul").append($li1);

#### 添加DOM节点

将新建的节点插入到文档中有多个方法，如下：append()、appendTo()、prepend()、prependTo()、after()、insertAfter()、before()、insertBefore()。

1. **append()方法**

append()方法向匹配的元素内部追加内容，方法如下：

$("target").append(element);

参考代码：

$("ul").append("<li title='香蕉'>香蕉</li>");

该方法查找ul元素，然后向ul中添加新建的li元素。

1. **appendTo()方法**

appendTo()方法将所有匹配的元素追加到指定的元素中，该方法是append()方法的颠倒操作。方法如下：

$(element).appendTo(target);

参考代码：

 $("<li title='荔枝'>荔枝<li>").appendTo("ul");

该方法新建元素li，然后把li添加到查找到的ul元素中。

1. **prepend()方法**

prepend()方法将每匹配的元素内部前置要添加的元素，方法如下：

$(target).prepend(element);

参考代码：

$("ul").prepend("<li title='芒果'>芒果</li>")

该方法将查找元素ul然后将新建的li元素作为ul子节点，且作为ul的第一个子节点插入到ul中。

1. **prependTo()方法**

prependTo()方法将元素添加到每一个匹配的元素内部前置，方法如下：

$(element).prependTo();

参考代码：

$("<li title='西瓜’>西瓜</li>").prependTo("ul");

该方法将新建的元b素li插入到查找到的ul元素中作为ul的第一个子节元素。

1. **after()方法**

after()方法向匹配的元素后面添加元素，新添加的元素做为目标元素后的紧邻的兄弟元素。方法如下：

$(target).after(element);

参考代码：b

$("p").after("<span>新加段新加段新加段新加段新加段</span>");

方法将查找节点p，然后把新建的元素添加到span节点后面做为p的兄弟节点。

1. **insertAfter()方法**

insertAfter()方法将新建的元素插入到查找到的目标元素后，做为目标元素的兄弟节点。方法如下：

$(element).insertAfter(target);

参考代码：

$("<p>insertAfter操作</p>").insertAfter("span");

方法将新建的p元素添加到查找到目标元素span后面，做为目标元素后面的第一个兄弟节点。

1. **before()方法**

before()方法b在每一个匹配的元素之前插入，做为匹配元素的前一个兄弟节点。方法如下:

$(target).before(element);

参考代码：

$("p").before("<span>下面是个段落</span>");

before方法查找每个元素p，将新建的span元素插入到元素p之前做为p的前一个兄弟节点。

1. **insertBefore()方法**

insertBefore()方法将新建元素添加到目标元素前，做为目标元素的前一个兄弟节点，方法如下:

$(element).insertBefore(target);

参考代码：

$("<a href='#'>锚</a>").insertBefore("ul");

insertBefore()新建a元素，将新建的a元素添加到元素ul前，做为ul的前一个兄弟节点。

增加元素的方法前四个是添加到元素内部，后四个是添加到元素外部的操作，有这些方法可以完成任何形式的元素添加。

#### 删除DOM节点操作

如果想要删除文档中的某个元素jQuery提供了两种删除节点的方法：remove()和empty();

1. **remove()方法**

remove()方法删除所有匹配的元素，传入的参数用于筛选元素，该方法能删除元素中的所有子节点，当匹配的节点及后代被删除后，该方法返回值是指向被删除节点的引用，因此可以使用该引用，再使用这些被删除的元素。方法如下：

$(element).remove();

参考代码：

$span=$("span").remove();

$span.insertAfter("ul");

该示例中先删除所有的span元素，把删除后的元素使用$span接收，把删除后的元素添加到ul后面做为ul的兄弟节点。该操作相当于将所有的span元素以及后代元素移到ul后面。

1. **empty()方法。**

empty()方法严格来讲并不是删除元素，该方法只是清空节点，它能清空元素中的所有子节点。方法如下:

$(element).empty();

参考代码：

$("ul li:eq(0)").empty();

该示例使用empty方法清空ul中第一个li的文本值。

#### 修改DOM节点操作

修改文档中的元素节点可以使用多种方法:复制节点、替换节点、包裹节点。

1. **复制节点$(element).clone()**

复制节点方法能够复制节点元素，并且能够根据参数决定是否复制节点元素的行为。方法如下:

$(element).clone(true);

参考代码：

$("ul li:eq(0)").clone(true);

该方法复制ul的第一个li元素，true参数决定复制元素时也复制元素行为，当不复制行为时没有参数。

1. **替换节点$(element).repalcewith()、$(element).repalceAll()**

替换节点方法能够替换某个节点，有两种形式形式实现：replaceWith()和replaceAll()，

**（1）**使用replaceWith方法使用后面的元素替换前面的元素。

$(oldelement).replaceWith(newelement);

参考代码：

$("p").replaceWith("<strong>我要留下</strong>");

该方法使用strong元素替换p元素。

**（2）**replaceAll方法使用前面的元素替换后面的元素。

参考代码：

$("<h3>替换strong</h3>").replaceAll("strong");

该例使用h3元素替换所有的strong元素。

1. **包裹节点$(element).wrap()、$(element).wrapAll()、$(element).wrapInner()**

包裹节点方法使用其他标记包裹目标元素从而改变元素的显示形式等，并且该操作不会破坏原始文档的词义。包裹节点有三种实现形式：wrap();wrapAll();wrapInner();

**（1）wrap()方法如下：**

$(dstelement).wrap(tag);

代码参考：

$("p").wrap("<b></b>");

该示例方法使用b标签包裹所有的p元素每个元素都使用b标签包裹。

**（2）wrapAll()方法如下：**

$(dstelement).wrapAll(tag);

代码参考：

$("p").wrapAll("<b></b>");

访示例方法使用b标签包裹所有的p元素，所有的p元素标签用一个b标签包裹。

**（3）wrapInner()方法如下：**

$(dstelement).wrapInner(tag);

代码参考：

$("strong").wrapInner("<b></b>");

该示例使用b标签包裹每个一strong元素的子元素。

#### 属性/样式/内容/遍历/CSS DOM

1. **属性操作attr()和removeAttr()**

**（1）**attr()方法能够获取元素属性，也能够设置元素属性。

当attr(para1)方法有一个参数时候用于获得当前元素的para1的属性值。

当attr(para1,attrValue)有两个参数时候设置当前元素的属性名为para1的属性值为attrValue。

参考代码：

$("p").attr("title");

该示例用于获得p元素的title属性值。

 $("p").attr("title","你最喜欢的水果");

该示例设b置p元素的title属性值为"你最喜欢的水果";

如果一次设置多个属性值可以使用"名/值"对形式，

参考代码：

$("p").attr({"title":"你最喜欢的水果","name":"水果"})。

该示例一次设置两个属性值。

**（2）**removeAttr()方法用于删除特定的属性，方法是在参数中指定属性名。

$("p").removeAttr("name");

该方法就是移除p元素的name属性。

1. **样式操作addClass()、removeClass()、toggleClass()和hasClass()**

**（1）**添加样式addClass()方法，使用该方法对目标元素添加相应的样式，方法如下：

$(element).addClass();

参考代码：

$("p").addClass("active");

该示例设置元素p的样式为active。

**（2）**移除样式removeClass()方法，使用该方法移除目标元素的指定样式，方法如下：

$(element).removeClass();

参考代码：

$("p").removeClass("active");

该实例去除掉p元素的active类样式。

**（3）**切换样式toggleClass()方法，使用该方法切换目标元素的样式，方法如下：

$(element).toggleClass();

参考代码：

$("p").toggleClass("active");

该方法来回切换【添加/删除实现切换】元素p的样式active.

**（4）**判断元素是否使用了样式$(element).hasClass(),方法如下：

$(element).hasClass(class);

参考代码：

alert($("p").hasClass("red"));

显示出p元素是否有red样式。

**备注：**addClass()和attr()方法设置样式的不同，attr方法把元素的属性名对应的属性值设为方法中的参数值，addClass()则把属性值添加到属性名对应的属性值中。

例：已有元素<p class='class1'>元素样式</p>,使用attr()和addClass()分别添加新样式。

$("p").attr("class","another")

结果是：<p class='another'>元素样式</>

$("p").addClass("class","another")

结果是：<p class='class1 another'>元素样式</p>

1. **设置和获取HTML,文本和值**

**（1）html()方法获得或设置某个元素的html元素。**

方法如下:

$(selector).html();

参考代码：

$("p").html();

该示例获得元素p的html内容。

$("p").html("<strong>添加html内容</strong>");

该示例设置p的html内容为"<strong>添加html内容</strong>";

**（2）text()方法获得或设置某个元素的文本值。**

方法如下:

$(selecotr).text();

参考代码：

$("p").text();

该示例获得元素p的text文本内容。

$("p").text("重新设置的文本内容");

该示例设置元素p的text文本为"重新设置的文本内容";

**（3）val()方法获得或设置某个元素的值，如果元素值是多选则以数组形式返回。**

方法如下：

$(selector).val();

例:文本元素 <input type="text" id="userName" value="请输入用户名" />

$("#userName").val();

获得input元素的值。

$("#userName").val('响马');

设置input元素的值为'响马'。

val()方法的不仅能操作input，最重要的一个用途用于select【下拉列表框】、checkbox【多选框】、radio【单选框】。

例：在下拉框下的多选赋值应用<select id="fruits" multiple="multiple"><option>苹果</option><option>香蕉</option><option>西瓜</option></select>

$("#fruits").val('苹果');

该示例使select中苹果选中。

$("#fruits").val(['苹果','香蕉']);

该示例使select中苹果和香蕉两项被选中。

1. **遍历节点操作children()、next()、prev()、siblings()和closest()**

（1）children()方法

用于取得匹配元素的子元素集合，只匹配子元素不考虑任何后代元素。方法如下：

$(selector).children();

参考代码：

$("$("body").children().length;

该示例获得body元素的子元素个数；

（2）next()方法

用于匹配元素的下一个兄弟节点，方法如下:

$(selector).next();

参考代码：

$("p").next().html();

该示例获得p元素的下一个兄弟节点的html内容。

（3）prev()方法

用于匹配元素的上一个兄弟节点，方法如下：

$(selector).prev();

参考代码：

$("ul").prev().text();

该示例获得ul元素的上一个兄弟节点的文本内容。

（4）siblings方法()

用于匹配目标元素的所有兄弟元素，方法如下：

$(selector).siblings();

参考代码：

$("p").slibings();

示例获得p元素的所有兄弟节点元素。

（5）closest()方法()

用来取得最近的匹配元素，首先检查当前元素是否匹配如果匹配则直接返回，否则继续向上查找父元素中符合条件的元素返回，如果没有匹配的元素则返回空jQuery对象。

1. **CSS-Dom操作css()、offset()、position()、scrollTop()和scrollLeft()**

（1）css()方法

用于获取、设置元素的一个或多个属性。方法如下：

$(selector).css();

代码参考：

$("p").css("color","red");

该示例用于设置元素的颜色属性为红色;

$("p").css("color");

该示例用于获得元素的color样式值;

$("p").css({"font-size":"30px","backgroundColor":"#888888"});

该示例用于设置元素的多个样式。

（2）offset()方法

用于获取元素相对当前窗体的偏移量，其返回对象包括两个属性：top和left。方法如下：

$(selector).offset()

var offset= $("p").offset();

var left=offset.left;

var top=offset.top;

该示例用于获得元素p的偏移量。

**备注:**offset()只对可见元素有效。

（3）position()方法

用于获取元素祖先元素中最近的一个position样式属性设置为relative或者absolute的相对偏移量。方法如下：

$(selector).position();

参考代码：

var postion = $("p").positon();

var left=positon.left;

var top=positon.top;

该示例用于获得元素p的位置。

（4）scrollTop()和scrollLeft()方法

用于获取元素的滚动条距顶端的距离和距左侧的距离。方法如下:

$(selector).scrollTop();

$(selector).scrollLeft();

参考代码：

var scrollTop=$("p").scrollTop();

var scrollLeft=$("p").scrollLeft();

该示例用于获得元素的滚动条的位置。

也可以添加参数将元素滚动到指定的位置。

$("textarea").scrollTop(300);

$("textarea").scrollLeft(300);

### jQuery的事件

JavaScript是基于事件触发的编程语言，HTML元素发生某些变化或者操作的时候，都会触发相应的"事件"，事件触发后就会调用与之关联的"事件处理函数"，JavaScript提供了"事件"与"事件处理函数"之间的关联方式，但jQuery提供了更加优雅的事件处理方式，实现了结构与行为的分离，增强了事件处理能力。

在HTML中，要实现事件和事件处理函数之间的绑定，需要在标签上增加相应的事件属性，并把事件处理函数设置为事件属性的属性值，例如：

<div onclick="showInfo()">点击我</div>

这种方式把标记的事件和事件处理函数通过一种固定的方式关联在一起，未实现结构与行为的分离，不灵活。

jQuery中提供了更加灵活的使用方式。

#### Ready事件

这个事件纯粹是对window.load事件的替代方法。通过使用这个方法，可以在DOM载入完毕后立即调用你所编写的函数，而90%的JavaScript函数都需要在那一刻执行。

要编写在DOM加载后运行的代码，可以这样写：

$(document).ready(function(){

//文档加载后需要执行的代码。

});

简写方式：

$(function($) {

//文档加载完后需要执行的代码。

});

除了Ready事件之外，jQuery中可以使用的事件和原生JavaScript中能够使用的事件是一样的，只是绑定方式不同。

绑定的事件处理函数默认采用匿名函数的方式定义，也可以采用先定义事件处理函数，后进行绑定的方式，例如：

//定义事件处理函数

function eventHandle(){

//事件处理代码

}

//事件绑定

$(document).ready(eventHandle());

#### 绑定方式1

jQuery中提供了一个Bind方法，当文档加载完毕之后，能够为DOM元素进行事件和事件处理函数的绑定，语法如下：

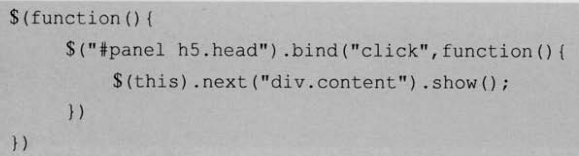
$("选择符").bind("事件名称"，function(event){

事件处理代码；

});

该代码作用是为"选择符"选中的元素的click事件绑定事件处理函数。

例如：



通过Bind方式绑定的事件处理函数，jQuery中提供了事件处理函数解绑方法，通过

$("选择符").unbind("事件");

可以将以前绑定的某个事件处理函数移除。

#### 绑定方式2

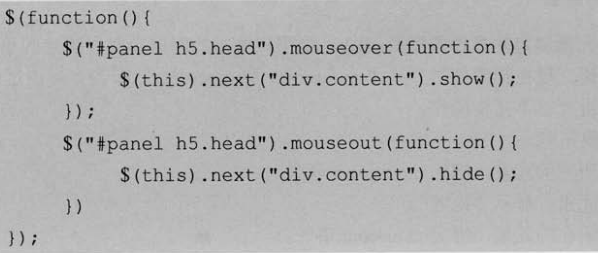
事件绑定方式1撰写的时候不是很方便，需要提供事件名称，如果拼写错误，则会出现绑定不上的问题。jQuery中还提供了一种方便的方式，如下：

$("选择符").事件(function(event){

事件处理代码;

});

例如：



注意：事件绑定代码一定要放在文档加载完毕之后执行。

#### 绑定方式3

绑定方式1和绑定方式2，要求被绑定的元素一定要加载完毕才可以进行事件与事件处理函数绑定。

如果要为新增的DOM元素进行事件和事件处理函数的绑定，方式1和方式2都会失效。

例如：如果在文档加载完毕的时候，通过

$("div").click(function(){ });

为页面上所有的div的click事件绑定了事件处理函数，如果后期通过DOM操作新增了一个div，那么新增的div的click事件并未绑定事件处理函数。

为了解决这个问题，jQuery中提供了另外一种事件绑定方式。

$("选择符"). on(events,[selector],fn) v1.7 之后提供的方式。

参数说明：

**events**:一个或多个用空格分隔的事件类型，如"click"。

**selector**:一个选择器字符串用于过滤器的触发事件的选择器元素的后代，如果启用该选择符，则代表后代元素来触发对应的事件。

**fn**:该事件被触发时执行的函数。

该方法提供了两种使用方法：

方式一：不使用[selector]参数。

$("div").on("click",function(){

事件处理代码；

});

该代码的意思是为页面上的div的click事件绑定事件处理函数，该方式和Bind函数具有相同的功能。但是对新增的元素，也无法进行事件的绑定。

方式二：使用[selector]参数。

$("ul").on("click","li",function(){

事件处理代码；

});

$("选择符")是要绑定元素所在的父元素的选择符，第一个参数是事件名，第二个参数是要绑定的元素选择符，第三个参数是事件处理函数。

该代码的意思为ul内所有li的click事件绑定事件处理函数，当点击li的时候，会触发事件处理函数的执行，如果通过DOM操作向body标记内新增li元素，新增的li标记的click事件也会被绑定上事件处理函数。

通过on方式进行的绑定，需要通过off方法进行解绑。

例如：

$("p").off();

解绑"p"选择符上绑定的所有事件。

$("body").on("click", "p", foo);

解绑body内所有p标记的click事件所绑定的foo事件处理函数。

#### 单次事件绑定

为每一个匹配元素的某事件绑定一个一次性的事件处理函数。

在每个对象上，这个事件处理函数只会被执行一次。其他规则与bind()函数相同。这个事件处理函数会接收到一个事件对象，可以通过它来阻止（浏览器）默认的行为。如果既想取消默认的行为，又想阻止事件起泡，这个事件处理函数必须返回false。

one(type,[data],fn)

#### 触发事件

$("选择符").trigger("事件名",[data]);

在每一个匹配的元素上触发某个事件。

这个函数也会导致浏览器同名的默认行为的执行。比如，如果用trigger()触发一个'submit'，则同样会导致浏览器提交表单。如果要阻止这种默认行为，应返回false。

你也可以触发由bind()注册的自定义事件而不限于浏览器默认事件。

例1:

$("form:first").trigger("submit")

例2：

//绑定事件

$("p").click( function (event, a, b) {

// 一个普通的点击事件时，a和b是undefined类型

// 如果用下面的语句触发，那么a指向"foo",而b指向"bar"

});

//触发事件

$("p").trigger("click", ["foo", "bar"]);

jQuery还提供了另外一个方法。

$("选择符").triggerHandle("事件名",[data]);

这个特别的方法将会触发指定的事件类型上所有绑定的处理函数。但不会执行浏览器默认动作，也不会产生事件冒泡。

这个方法的行为表现与trigger类似，但有以下三个主要区别：

第一，他不会触发浏览器默认事件。

第二，只触发jQuery对象集合中第一个元素的事件处理函数。

#### 复合事件

jQuery中还提供了一些方法，能够一次性的绑定多个事件处理函数，这些方法称为复合事件。

1、hover([over],[out])

当要为某个元素绑定mouseover和mouseout两个事件的事件处理函数的时候，无需通过两次绑定，只需要通过hover方法就可以一次性的完成事件绑定。

$("td").hover(

function () {

$(this).addClass("hover");

},

function () {

$(this).removeClass("hover");

}

);

第一个事件处理函数绑定到了mouseover，第二个事件处理函数则绑定到了mouseout。

2、toggle([fn1],[fn2],[fn3]…)

通过toggle绑定的事件处理函数能够实现如下的效果：

第一次点击元素的时候会执行第一个事件处理函数，

第二次点击元素的时候会执行第二个事件处理函数，

第三次点击元素的时候会执行第三个事件处理函数，

依次类推…

如果共绑定了三个事件处理函数。

第四次点击元素的时候会执行第一个事件处理函数，

第五次点击元素的时候会执行第二个事件处理函数，

依次类推…

该功能从Jquery 1.9之后被作废了，如果需要继续使用该效果，需要添加如下的Jquery代码：

$.fn.toggle = function( fn, fn2 ) {

var args = arguments,guid = fn.guid || $.guid++,i=0,

toggle = function( event ) {

var lastToggle = ( $.\_data( this, "lastToggle" + fn.guid ) || 0 ) % i;

$.\_data( this, "lastToggle" + fn.guid, lastToggle + 1 );

event.preventDefault();

return args[ lastToggle ].apply( this, arguments ) || false;

};

toggle.guid = guid;

while ( i < args.length ) {

args[ i++ ].guid = guid;

}

return this.click( toggle );

};

例如：

$("td"). toggle (

function () {

$(this).addClass("hover");

},

function () {

$(this).removeClass("hover");

}

);

Text,html,val

#### 事件冒泡

DOM元素是相互嵌套的，当触发某元素的某事件时，其父元素相应的事件也会被执行，这种现象称为事件冒泡，为了阻止事件冒泡现象的发生，需要在事件处理函数内通过如下的方法来实现：

event.stopPropagation()

event是事件处理函数的参数名称。

#### 事件参数

事件处理函数允许携带多个参数，第一个参数默认是事件参数，参数内携带了一些与事件有关的信息，如果是鼠标事件，则会携带鼠标位置，鼠标按键等相关信息，如果是键盘事件，则会携带键盘有关的信息等，通过这些信息，我们可以完成一些相关的功能。

事件参数携带的属性和方法有：

1. type属性：获取事件的类型
2. preventDefault()方法： 阻止默认事件行为的触发，例如a标签点击的时候，不想出发超链接跳转就需要通过这个方法，或者返回false。
3. stopPropagation()方法：阻止冒泡。
4. pageX属性：获取鼠标相对于文档左边缘的位置。
5. pageY属性：获取鼠标相对于文档右边缘的位置。
6. currentTarget属性：获取触发该事件的元素，也可以通过$(this)。
7. which属性：针对键盘和鼠标事件，这个属性能确定你到底按的是哪个键或按钮。

### jQuery的动画

动画效果是jQuery库吸引人的地方，通过jQuery的动画方法，能够轻松的为网页添加非常精彩的视觉效果，给用户一种全新的体验。

为了演示相关的方法效果，我们定义如下的HTML元素。

<div class="header" style="width:200px;height:25px;border:1px solid red;">

新闻标题

</div>

<div class="content" style="width:200px;height:200px;border:1px solid red;">

新闻内容

</div>

#### show()和hide()方法

show()方法和hide()方法是jQuery中最基本的动画方法，能够控制元素的隐藏和显示。

hide()方法就是把元素的display属性设置为none。

这两个方法都可以携带参数，参数就是完成显示或隐藏所需要的时间（单位：毫秒），

$("div").hide(1000);

就是让div慢慢的隐藏掉，大家可以验证一下运行的效果，会感到很神奇，jQuery提供的动画都提供了事件参数，后边的方法就再提及。

如果点击新闻标题隐藏新闻内容，再次点击，则显示新闻内容，间隔事件为2秒钟，代码如下：

$(function(){

$("div.header").toggle(function(){

$("div.content").hide(2000);

},function(){

$("div.content").show(2000);

});

});

#### fadeIn()和fadeOut()方法

与show()和hide()方法不相同的是，fadeIn()和fadeOut()方法只改变元素的不透明度。fadeOut()方法会在指定的一段事件内降低元素的不透明度，直到元素完全消失，fadeIn()方法则相反。

如果点击新闻标题淡出新闻内容，再次点击，则淡入新闻内容，间隔时间为2秒钟，代码如下：

$(function(){

$("div.header").toggle(function(){

$("div.content").fadeOut(2000);

},function(){

$("div.content").fadeIn(2000);

});

});

#### slideUp()和slideDown()方法

slideUp()和slideDown()方法用来模拟升降窗效果，slideUp()用来实现元素的慢慢向上缩短，直到隐藏。

如果点击新闻标题收起新闻内容，再次点击，则展开新闻内容，间隔时间为2秒钟，代码如下：

$(function(){

$("div.header").toggle(function(){

$("div.content").slideUp(2000);

},function(){

$("div.content").slideDown(2000);

});

});

#### animate()方法

通过如上的方法，无法满足用户的各种需求，如果要对动画进行更多的控制，需要采用一些高级的自定义动画来解决这些问题。

jQuery提供了一个animate()方法来实现特殊的效果。

语法结构如下：

animate(params,[speed],[callback])

参数说明：

1）params:是动画结束后元素的样式，例如：{width:300px;Height:500px;color:red;}，代表元素要从当前的样式效果变更为参数所指定的样式效果。

2）speed:变化的时间间隔，可选参数。

3）回调函数，当变化完成后所调用的函数，如果变化完毕后，要继续变化到另一个状态，可以采用回调函数继续变化，可选参数。

如果新闻标题点击后，要把新闻内容放大一倍，并且字体变成红色，变更完毕之后，再变化回来，每次变化的间隔事件都是2秒，代码如下：

$(function(){

$("div.header").click(function(){

$("div.content").animate({ width: "400px", height: "400px", color: "red" }, 2000, function () {

$("div.content").animate({ width: "200px", height: "200px", color: "black" }, 2000);

});

});

});

#### 停止动画

很多时候需要停止匹配元素正在进行的动画，要在某处停止动画，需要使用stop()方法。

语法结构如下：

stop([clearQueue],[goToEnd]);

两个参数都是布尔类型

1. clearQueue代表是否清空未执行的动画队列。
2. goToEnd代表是否直接将正在执行的动画跳转到未状态。

如果直接使用stop()方法，不携带参数，则会立即停止当前正在执行的动画，如果接下来还有动画等待继续执行，则以当前状态开始接下来的动画。

经常会遇到这样的情况，为一个元素绑定了hover复合事件，移入和移除会执行相应的动画，如果快速的移入和移除，就会出现上一次移入和移除的动画未执行完，无法快速的相应移入和移除的变化，这个时候就可以采用stop()方法，当移除后，如果移入的动画未执行完，则终止，立刻执行移除的动画，移入的时候也可以采用stop()方法，让移入的动画立刻执行。

如果为某个元素绑定了组合连续动画，则需要利用两个参数来完成相关的控制。

#### 判断动画是否在执行

当使用animate()方法的时候，要避免动画积累而导致的动画与用户的行为不一致，当用户快速在某个锇元素上执行animate()动画时，就会出现动画积累。解决方法是判断元素是否正处于动画状态，如果元素不处于动画状态，则为元素添加新的动画，否则不添加。

判断的方法如下：

$("选择符").is(":animated")

返回布尔类型

#### 其它动画方法

1、toggle(speed,[callback])

用来切换元素的可见状态，如果元素是可见的，切换为隐藏，如果是隐藏则，则切换为可见，交替执行show()和hide()方法。

例如：

$(function(){

$("div.header").click(function(){

$("div.content").toggle(2000);

});

});

类似于：

$(function(){

$("div.header").toggle(function(){

$("div.content").hide(2000);

},function(){

$("div.content").show(2000);

});

});

2、slideToggle(speed,[callback])

用来切换元素的高度，类似于交替执行slideUp()和slideDown()方法。

例如：

$(function(){

$("div.header").click(function(){

$("div.content"). slideToggle (2000);

});

});

类似于：

$(function(){

$("div.header").toggle(function(){

$("div.content").slideUp(2000);

},function(){

$("div.content").slideDown(2000);

});

});

3、fadeTo(speed,opacity,[callback])

这个方法可以把元素的不透明度以渐进方式调整到指定的值。

例如：

$(function(){

$("div.header").click(function(){

$("div.content").fadeTo(2000);

});

});

### jQuery的AJAX

AJAX全程为"Asynchronous JavaScript and XML"（异步JavaScript和XML），它并不是指一种单一的技术，而是有机地利用了一些列交互式网页相关的技术所形成的结合体，它的出现，揭开了无刷新更新网页的新时代，并有代替传递的Web方式和通过隐藏的框架来进行异步提交的趋势，是Web开发应用的一个里程碑。

AJAX具有如下的优势：

1、不需要插件支持

2、优秀的用户体验

3、提高Web程序的性能

4、减轻服务器和带宽的负担

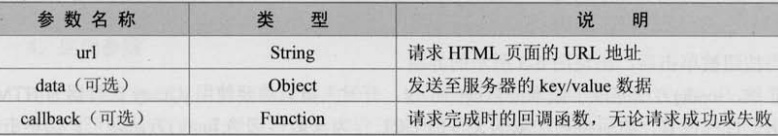
要进行AJAX的学习，需要搭建服务器运行环境，AJAX的请求是发送给服务器的，通过服务器的来回应AJAX请求的内容。

jQuery中把常用的AJAX请求都封装成一个个的方法，通过调用对应的方法，就可以实现AJAX请求效果。

#### load()方法

$("选择符").load()方法用来异步加载网页内容，把载入的HTML内容插入选择符中，语法结构如下：

load(url,[data],[callback])



例如：

在页面上添加如下的HTML

<div class="loadBaidu">载入百度</div>

在页面上增加如下的jQuery代码

$(function(){

$(". loadBaidu").click(function(){

$(".loadBaidu").load("http://www.baidu.com",null,function(){

alert("载入完毕");

});

});

});

#### $.getScript()方法

Web2.0时代，一个页面中所需要的JS代码越来越多，在页面初次加载的时候不需要全部载入，在后期需要的时候，再载入到页面中，这个时候就需要使用到$.getScript()方法。

语法格式：

$.getScript("jsUrl",[callback]);



#### $.ajax()方法

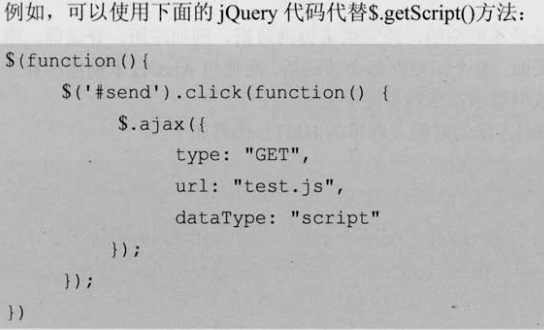
$.ajax()方法是jQuery最底层的AJAX实现，通过该方法能够实现所有的AJAX请求需求。

语法结构：

$.ajax(options)

参数以key/value的形式存在，有的参数是可选的。

|  |  |  |
| --- | --- | --- |
| **参数名** | **参数类型** | **参数作用** |
| url | string | 请求的网址 |
| type | string | 请求的类型;get/post |
| data | object | 携带的数据，格式{参数:值,参数:值} |
| dataType | string | 服务器返回的数据类型  支持的类型[xml,html,json,script,text] |
| beforeSend | function | 发送前执行的代码 |
| complete | function | 请求结束执行的代码 |
| success | function | 请求成功执行的代码 |
| error | function | 请求出错执行的代码 |
| global | boolean | 默认true，是否触发ajaxStart和ajaxStop事件 |



#### 元素序列化方法

当采用AJAX进行请求的时候，需要把表单控件的值以参数的方式传递给后台，可以采用$("表单选择符").val()的方式获取表单的值，当表单数量少的时候这种方式还凑合能用，当表单数量很多的时候，就不合适了。

jQuery中提供了一些数据序列化相关的方法。

**1、serialize()方法**

用来把选择符选中的表单或选择符内所有的表单序列化字符串格式，格式形式如下：

表单标识=表单值&表单标识=表单值…

备注：表单标识是表单的name属性值。

例如：

$("form1").serialize()

把form1容器内所有的表单序列化为字符串。

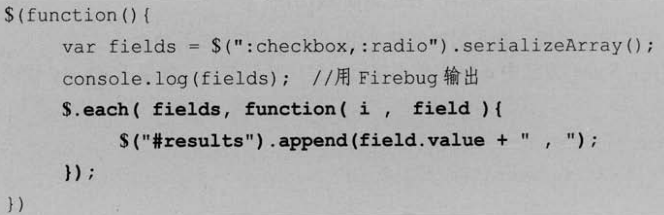
$(":checkbox,:radio").serialize()

把页面内所有复选框和单选框的序列化为字符串，只会讲选中的值序列化。

**2、serializeArray()方法**

该方法不是返回字符串，而是把表单元素序列化后，返回JSON格式的数据。

如果返回的是一个集合对象，就可以对其进行遍历，例如：



#### AJAX全局事件

jQuery提供了一些自定义全局函数，能够为各种与AJAX相关的之间注册回调函数，例如当AJAX请求开始的时候，会触发ajaxStart()方法的回调函数；当AJAX请求结束时，会触发ajaxStop()方法的回调函数，这些方法都是全局的方法，因此无论创建他们的代码位于何处，只要有AJAX请求发生时，就会触发他们。

当AJAX请求需要的时间很长是，我们可以通过这些全局方法来实现相关的提示功能，提高用户的体验。

例如：当AJAX请求未回应的会后，把屏幕罩住，并显示加载动画。

1）HTML中提供展示加载效果的元素

<div id="loading">

<img src="/Content/images/loading.gif" />

</div>

2）CSS中定义层的遮罩效果和加载动画图片的位置

#loading {

background: #000;

filter: alpha(opacity=30); /\* IE的透明度 \*/

opacity: 0.3; /\* 透明度 \*/

display: none;

position: absolute;

top: 0px;

left: 0px;

width: 100%;

z-index: 100; /\* 此处的图层要大于页面 \*/

display: none;

}

#loading img

{

width: 124px;

height: 124px;

position: absolute;

top: 50%;

left: 50%;

margin-left: -62px;

margin-top: -62px;

}

3）jQuery代码中，控制遮罩层的展示，并根据屏幕大小设置遮罩层的高度。

$(function(){

//ajax请求开始显示loading

$(document).ajaxStart(function () {

$("#loading").css("height", $(document).height());

$("#loading").show();

});

//ajax请求结束隐藏loading

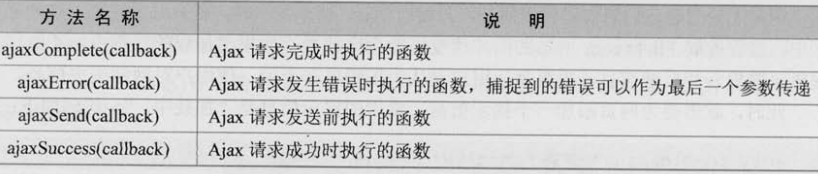
$(document).ajaxStop(function () {

$("#loading").hide();

});

});

AJAX还提供了其它的全局函数：



### jQuery的其它相关方法

1. **index方法**

$("选择符1").index(选择符2)，返回选择符2在选择符1中的索引位置。

1. **each方法**

$("选择付").each(function(){

});

遍历选择符选中的每个元素，对每个元素进行处理，each方法内的this指当前遍历的某个元素。

1. **data**

$("选择付").data("参数名","参数值") 在选择符元素上附加数据项。

$("选择付").data("参数名") 获取选择符上附加的数据项。

1. **$.trim("字符串")**

去除字符串两头的空格符号。

1. **元素筛选类型的**

见：http://jquery.cuishifeng.cn/

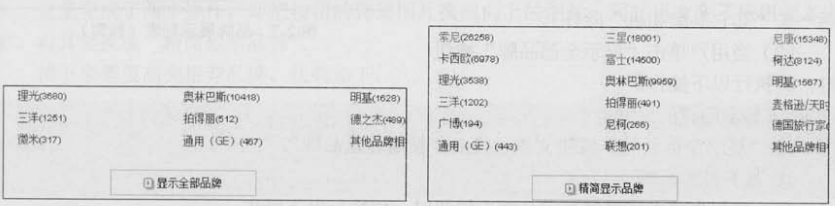
1. **位置和尺寸类的方法**

见：http://jquery.cuishifeng.cn/

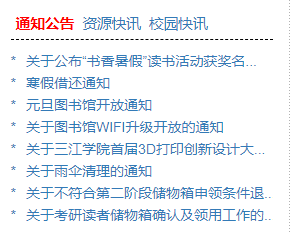
### 相关练习

1. **按照要求完成如下的练习。**

通过三列来展示相关的品牌，默认情况下只展示前三行，单击下边的"显示全部品牌"后，显示出所有的品牌，如右边图效果，并且修改下边的按钮为"精简显示品牌"，当点击"精简显示品牌"后，再显示左边图效果。



1. **完成三江图书馆首页的Tab切换效果。**



1. **完成三江图书馆首页导航栏目以及弹出功能。**







1. **完成商品放大图展示的效果。**



## JSON技术

[JSON](https://baike.baidu.com/item/JSON)([JavaScript](https://baike.baidu.com/item/JavaScript) Object Notation, JS 对象简谱) 是一种轻量级的数据交换格式。它基于 [ECMAScript](https://baike.baidu.com/item/ECMAScript) (欧洲计算机协会制定的js规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

JSON是一个对象，内部是key/value格式，Value支持多种形式：

1）字符串

2）数组

3）JSON对象

例如：

1）普通的JSON

{"firstName": "Brett", "lastName": "McLaughlin"}

2）包含数组的JSON，数据内还是JSON

{

"people": [{

"firstName": "Brett",

"lastName": "McLaughlin"

},

{

"firstName": "Jason",

"lastName": "Hunter"

}

]

}

3）复杂的JSON

{

"name": "中国",

"province": [{

"name": "黑龙江",

"cities": {

"city": ["哈尔滨", "大庆"]

}

}, {

"name": "广东",

"cities": {

"city": ["广州", "深圳", "珠海"]

}

}, {

"name": "台湾",

"cities": {

"city": ["台北", "高雄"]

}

}, {

"name": "新疆",

"cities": {

"city": ["乌鲁木齐"]

}

}]

}