

《网络编程技术》

机械工程及自动化学院

陈少强

bhidwww@163.com

口令: bhidwww2013



第二章 HTML简介

2.1 HTML、XML、XHTML、HTML5

2.1.1 HTML标准

HTML是构造网页文件和建设网站的基础，为了解决不兼容性问题，就必须为其制定标准，并由一个中立机构来控制。

目前，控制HTML标准的机构是万维网联盟（World Wide Web Consortium，即W3C）。该联盟于1994年10月在麻省理工学院计算机科学实验室成立，联盟的建立者是万维网的发明者蒂姆·伯纳特·李。

W3C每隔一段时间就会公布一套新的HTML标准，并加上修订版本编号来区分，例如1999年12月公布的HTML4.01。该联盟还制订了包括XML和CSS等众多影响深远的标准规范。



2.1 HTML、XML、XHTML、HTML5（续）

2.1.2 XML

XML是eXtensible Markup Language的缩写，即“可扩展标记语言”。在XML中，开发者可以根据自己的需要定义自己的标记，只要这种标记满足XML的命名规则。换句话说，XML是一种能够创造其它语言的语言，是一种元标记语言。

XML更重要的作用在于它可以表示结构化的数据，从而有利于数据交换。由于可以自定义标记，开发者可以使用XML来确定数据的格式和数据之间的关系，而这些是与应用程序、操作系统等平台无关的。所以，目前XML最广泛的应用是在不同应用程序间，不同数据库系统间交换数据。



2.1 HTML、XML、XHTML、HTML5（续）

2.1.3 XHTML

XHTML的英文全称是eXtensible Hypertext Markup Language，即“可扩展超文本标记语言”。

XHTML是一种基于XML的标记语言，是使用XML对HTML进行重新定义的语言。它看起来与HTML有些相似，但比HTML更加严格。

2000年1月，W3C将其认定为推荐标准，将其视为HTML的最新版本。它向下兼容HTML4.01（但有更严格要求），计划逐步取代HTML。



2.1 HTML、XML、XHTML、HTML5（续）

2.1.4 HTML5

然而事情的发展并未如W3C所愿。由于XHTML语法严格，并且不向下兼容，受到了Opera、Apple等浏览器厂商的抵制。它们脱离W3C成立了WHATWG（Web Hypertext Applications Technology Working Group，Web超文本应用技术工作组）开始研究HTML的下一版本。

在此情况下，W3C于2007年成立了HTML5工作组，在WHATWG的工作成果基础上继续HTML的发展工作。

HTML5更加简洁，兼容性好。

功能有了本质的提高，一些有趣的新特性包括：

- 用于绘画的 **canvas** 元素
- 用于媒介回放的 **video** 和 **audio** 元素
- 对本地离线存储的更好的支持
- 新的特殊内容元素，比如 **article**、**footer**、**header**、**nav**、**section**
- 新的表单控件，比如 **email**、**url**、**date**、**number**、**range**、**color**等



1.2 HTML标记与属性

1.2.1 一个简单的HTML5例子：1-1.html

```
<!DOCTYPE html >
```

```
<html lang="zh-cn">
```

```
  <head>
```

```
    <meta charset="UTF-8">
```

```
    <title>网页标题</title>
```

```
  </head>
```

```
  <body>
```

```
    这是我的第一个主页。<b>加黑文本</b>
```

```
  </body>
```

```
</html>
```



1.2 HTML标记与属性（续）

说明：

一个HTML文件的组成可分为以下几个部分：

1) DOCTYPE声明

THML5只剩这么一句： `<!DOCTYPE html>`

2) 顶层标记<HTML>

THML5中可以没有，也可以很简单：

`<html >...</html>`

或：

`<html lang="zh-cn">...</html>`



1.2 HTML标记与属性（续）

3) 网页表头

网页表头的范围以<head>标记开始，以</head>标记终止。在<head>、</head>间的范围用来加入网页的相关信息。例如网页标题由标记<title>～</title>设置，其中的内容将出现在浏览器窗口标题栏。

4) 网页主体

网页主体的范围以<body>标记开始，以</body>标记终止。在<body>、</body>之间的范围用来放置网页的主体内容，例如图片、文字、表格等。



1.2 HTML标记与属性（续）

1.2.2 HTML标记

标记包含在尖括号中，其中的文本是为HTML解释器提供的指令。对HTML的学习实际上主要是对HTML中的各种标记的认识、掌握和熟练应用。HTML的标记不区分大小写，可以分为两类：

- 双标记：

双标记成对出现，其语法格式为：<标记>文档内容</标记>

起始标记告诉浏览器从此处开始执行该标记所表示的功能；结尾标记告诉浏览器在这里结束此功能。例如：

加黑文本

HTML标记大部分是双标记。

- 单标记：

单标记的语法格式是<标记>，它只需要单独使用就能表达完整的意思。单标记的种类不多，例如：
，它表示换行。



1.2 HTML标记与属性（续）

1.2.3 HTML注释

HTML和XML的注释具有相同的形式，其语法格式为：

```
<!--注释内容-->
```

1.2.4 HTML标记的属性

不论是双标记还是单标记，都表示一个HTML元素。HTML元素可以具有属性，属性可以扩展HTML元素的能力。比如可以使用一个**bgcolor**属性，使得页面的背景称为黄色：

```
<body bgcolor="yellow">
```

属性设置通常是附加给HTML的起始标志，由属性名和属性值共同构成，其一般形式为：**name="value"**。不同的属性设置间以空格分隔（而不是逗号）。



1.2 HTML标记与属性（续）

1.2.5 HTML中的超链接

HTML的一个很重要的特色是能够创建“超链接”，通过点击超链接可以实现指定的跳转。下面是一个使用绝对URL的超链接：

```
<a href="http://www.buaa.edu.cn">北京航空航天大学</a>
```

超链接中也可以使用相对URL，例如：

```
<a href="../xxgk/index.htm">学校概况</a>
```

所谓相对指相对于所在文档。

另外，可以使用<a>标记的name属性来定义书签，例如：

```
<a name="书签名">提示文本</a>
```

这样其它的超链接就可以直接跳转到书签位置，例如：

```
<a href="#书签名">标记内容（文本或图像） </a>
```

注意：定义书签的**name**属性也可以用**id**属性代替，而且**THML5**中的<a>标记已经没有**name**属性了。



1.2 HTML标记与属性（续）

1.2.6 元数据标记<meta>

所谓元数据就是描述数据的数据。该标记是个单标记，过去内容很多，在HTML5中常用的是：

```
<meta charset="UTF-8">
```



1.2 HTML标记与属性（续）

1.2.7 其它比较重要的HTML元素

- 通用区块元素：对应的标记是<div>。使<div>、</div>之间的内容成为一个块状元素，用于布局。
- 通用线内元素：对应的标记是。使、之间的内容成为一个线内元素，用于布局。
- 图像：对应的标记是，用于在网页中引入图像。
- 表格：对应的标记是<table>，用于在网页中插入表格。其主要子元素有：表格标题<caption>、表头<thead>、表尾<tfoot>、表格主体<tbody>、表格行<tr>、表格单元<td>等。
- 表单：对应的标记是<form>，用于在网页中插入表单。表单用于设计交互界面，其主要子元素有：输入字段<input>、选择列表<select>、多行文本输入框<textarea>、按钮<button>等。
- 分级标题：对应的标记是<h1>～<h6>，用来显示各级标题。
- 列表：包括无序列表(unordered list)和有序列表(ordered list)，对应的标记分别是和。



1.3 HTML5新标记简介

1.3.1 画布标记

画布是一个矩形区域，原点在左上角。**canvas** 元素使用 **JavaScript** 在网页上绘图，拥有绘制二维图形、字符及图像的方法。

见例：1-2.html

1.3.2 音频和视频标记

audio 元素能够播放声音文件。

video 元素能够播放视频文件。

目前各浏览器支持的格式有限，并且不同。

见例：1-3.html、1-4.html



1.3 HTML5新标记简介

1.3.3 客户端存储

HTML5 提供了两种在客户端存储数据的新方法：

- **localStorage** - 没有时间限制的数据存储
- **sessionStorage** - 针对一个 **session** 的数据存储

对于不同的网站不同的浏览器，数据存储于不同的区域，并且一个网站只能访问其自身的数据。

过去，这些都是由 **cookie** 完成的。但是 **cookie** 不适合大量数据的存储，速度很慢而且效率不高。

现在，HTML5 使用 **JavaScript** 来完成这个任务。

参见例1-5.html和1-6.html



1.3 HTML5新标记简介

1.3.4 新的内容元素

这些标签使得网页的结构更附有实际意义。主要有：

- **article**: 适合包容一篇独立文章的自包含内容，例如：

```
<article>
```

```
  <h1>Internet Explorer 9</h1>
```

```
  <p>简称 IE9，于 2011 年 3 月 14 日发布.....</p>
```

```
</article>
```

- **header**: 表示页眉，例如：

```
<header>
```

```
  <h1>Welcome to my homepage</h1>
```

```
  <p>My name is Donald Duck</p>
```

```
</header>
```




1.3 HTML5新标记简介

- footer: 表示页脚。例如:

```
<footer> <p>Posted by: W3School</p> </footer>
```

- nav: 表示导航区域。例如:

```
<nav>
```

```
<a href="index.asp">Home</a>
```

```
<a href="html5_meter.asp">Previous</a>
```

```
<a href="html5_noscript.asp">Next</a>
```

```
</nav>
```

- section: 文档中的片段, 比如章节、页眉、页脚等部分。例如:

```
<section>
```

```
<h1>PRC</h1>
```

```
<p>The People's Republic of China was born in 1949...</p>
```

```
</section>
```



1.3 HTML5新标记简介

1.3.5 新的表单控件

即新的表单输入类型，它们提供了更好的输入体验和验证。包括 email、url、date、number、range、color等。

见例：1-7.html



1.3 HTML5新标记简介

1.3.6 地理定位(Geolocation)

HTML5 可以通过其JavaScript API获得用户的地理坐标（当然这需要用户的同意），这样就能实现移动客户端的基于地理位置的应用。

1) `getCurrentPosition()`方法可以得到用户的当前位置。

见例：1-9.html /* 只IE成功 */

2) `watchPosition()`方法则可以不断返回用户的位置，直到用
`clearWatch()`方法来停止。

见例：1-A.html /* 只IE成功 */

有了地理定位功能，就可以调用谷歌或百度的地图服务，在地图中显示当前位置。

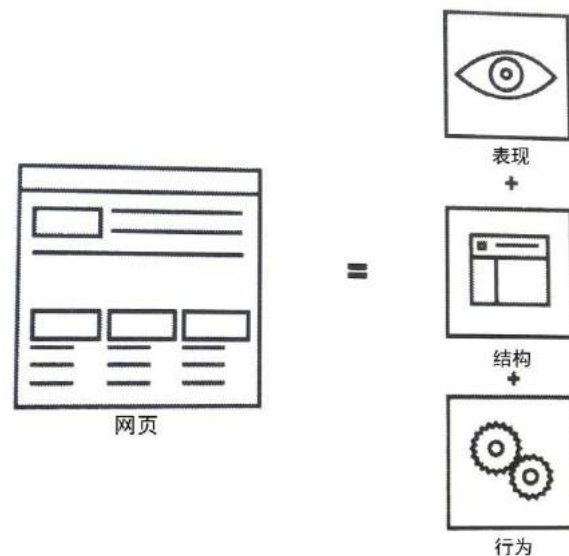
见例：1-B.html /*只IE、百度地图成功 */

第二章 CSS简介

2.1 网页组成

随着Web技术的发展，人们不再仅仅满足于在页面上浏览文本和图片并使用超链接，大家希望的是更加赏心悦目的界面并与之互动，而这些是HTML所不能满足的。目前认为，Web页面有三大部分内容：即：结构、表现和行为。

- 结构：网页的内容及层次关系。
- 表现：结构化内容的展示方式。
包括版式、颜色大小等。
- 行为：与用户的交互能力。



网页的三大组成部分



2.2 前台技术

要完成上述三部分内容，需要有相应的技术来实现。由于都是直接面对用户的技术，所以称为前台技术，或称表现层技术。具体包括：

- 结构技术：主要有HTML、XML、XHTML。
- 表现技术：主要是CSS
- 行为技术：主要是各种前台脚本语言，例如JavaScript。



2.3 层叠样式表（CSS）

网页设计最初是用**HTML**标记来定义网页的内容和简单效果，主要侧重于定义内容，并没有过多涉及排版和界面效果。当绚丽的页面变得越来越重要时，**HTML**就难以满足需求了。**CSS**的出现解决了这个问题。

CSS是**Cascading Style Sheets**（层叠样式表单）的简称，也叫样式表。它专门用来实现排版及网页元素（由**HTML**标记表示）的显示样式。

由于有了**CSS**，实现了内容与表现的分离，即：

- **HTML**（或**XHTML**）用来表示网页内容及结构。
- **CSS**用来决定网页内容的排版和显示方式。



2.4 CSS的使用方式

2.4.1 内嵌样式:

即将样式作为标记的**style**属性值直接写在标记内，样式之间以“；”分割。见下例， 2-1.html：

```
<!DOCTYPE...>
<html>
  <head>
    <title>网络编程技术</title>
  </head>
  <body>
    <h2 style="font-size:20pt">前台技术</h2>
    <p style="background-color:rgb(255,0,0)">HTML</p>
    <p style="background-color:#00ff00">CSS</p>
    <p style="background-color:#00f">JavaScript</p>
  </body>
</html>
```



2.4 CSS的使用方式

2.4.2 内部样式表:

内部样式表使用**style**标记，写在**head**标记之间，见下例，2-2.html：

```
<html>
<head>
  <title>网络编程技术</title>
  <style type="text/css">
    h2 {font-size:30pt; background-color:rgb(0,255,0)}
    p {background-color:rgb(255,0,0)}
  </style>
</head>
<body>
  <h2>前台技术</h2>
  <p>HTML</p>
  <p>CSS</p>
  <p>JavaScript</p>
</body>
</html>
```




2.4 CSS的使用方式

2.4.3 导入外部样式表:

将CSS文件导入到内部样式表, 见下例, 2-3.html:

```
<html>
<head>
  <title>网络编程技术</title>
  <style type="text/css">
    @import url(2-3.css); <!--注意分号一定要-->
  </style>
</head>
<body>
  <h2>前台技术</h2>
  <p>HTML</p>
  <p>CSS</p>
  <p>JavaScript</p>
</body>
</html>
```

```
/*2-3.css*/
h2{
  font-size:30pt;
  background-color:rgb(0,255,0)
}
p{background-color:rgb(255,0,0)}
```



2.4 CSS的使用方式

2.4.4 链接外部样式表:

使用link标记, 从<head>~</head>部分链接一个CSS文件, 见下例2-4.html:

```
<html>
<head>
  <title>网络编程技术</title>
  <link rel="stylesheet" href="2-3.css">
</head>
<body>
  <h2>前台技术</h2>
  <p>HTML</p>
  <p>CSS</p>
  <p>JavaScript</p>
</body>
</html>
```

注: 前面介绍的@import语句, 也可用在CSS文件中。



2.5 HTML与CSS的接口

为了能够准确地使用**CSS**控制由标记定义的**HTML**各元素的表现形式（也包括将来使用脚本控制其行为），**HTML**为所有标记定义了**id**和**class**属性，以便从不同级别定位元素。

id：元素标识符属性。用来唯一地标识一个元素。

class：类型标识符属性。用来表示元素所属的类型。多个元素可以属于一个类型，一个元素也可以属于多个类型（以空格隔开）。

例如：

```
<p id="id1" class="para cl1">HTML</p>
```

```
<p id="id2" class="para cl1">CSS</p>
```

```
<p id="id3" class="para cl2">JavaScript</p>
```



2.6 CSS词法结构

前面已经见过简单的CSS样式定义，例如：

```
p {background-color:rgb(255,0,0);}
```

CSS词法结构由3个部分组成：选择器(Selector)、属性(Property)和值(Value)。使用方法是：

```
Selector {Property: Value;}
```

其中：

选择器：指出需要设置样式的对象，有各种指定方式。

属性和值：指定具体的一个样式。

重要：CSS样式表中可以包含注释，形如： */*注释*/*



2.6 CSS词法结构

2.6.1 CSS选择器

1) *: 通用选择器举例:

```
p *{color:#ff0000}
```

p标记内的所有元素的文字都是红色。

2) 标记选择器举例:

```
p {color:#ff0000}
```

p标记内的文字为红色。

3) 群组选择器举例:

```
h1,h2,h3,p{font-size:12px; font-family:arial}
```

同时为多种选择器定义字体及大小

4) 包含选择器举例:

```
h2 p{font-weight:bold}
```

为h2标记内的p元素（可能相隔几层）设置字体的粗细



2.6.1 CSS选择器

5) 子元素选择器举例:

```
h2>p{font-weight:bold}
```

为以h2标记为父元素的p标记（只相差一级）设置字体的粗细

6) id选择器举例:

```
#content{font-size:14px} /*<div id="content">...</div>*/
```

为元素“content”设置字体大小。

7) class选择器举例:

```
.p1 {                                /*<div class="p1">...</div>*/  
margin:10px;                        /*<h2 class="p1">...</h2>*/  
background-color:blue               /*<p class="p1">...</p>*/  
}
```

为具有属性class="p1"的元素（可能不仅属于p1）设置外边距和背景色。



2.6.1 CSS选择器

8) 属性选择器

属性选择器可以根据元素的属性及属性值来选择元素。**id**和**class**也适用该选择器。其格式为：名称[表达式]。例如：

```
a[href][title^="def"] {color:red;}
```

将有**href**属性且**title**属性以**def**开头的超链接的文本设置为红色

9) 类型标记选择器举例：

```
h2.p1{font-size:12px; font-family:arial}
```

为**p1**类型的**h2**标记指定字体及大小

10) 组合选择器举例：

上述各种选择器均可进行组合。

```
h1 .p1{...}
```

```
#content h1{...}
```

```
h1 .p1, #content h1{...}
```

```
h1 #content h2{...}
```



2.6.2 伪类和伪元素

CSS自定义了伪类和伪元素来增强对页面的控制。

1) （主要）伪类：设置对象的动态样式

伪类	用途
:link	链接未被访问时的样式
:hover	链接在鼠标移上时的样式
:active	链接被点击与释放间的样式
:visited	链接已被访问后的样式
:focus	有输入焦点的对象样式
:first-child	作为第一个子对象的样式

例如：

```
a:hover{background-color:#333333}/*#303536#ffffff*/
```

设置当鼠标在链接上时，链接的背景色变灰



2.6.2 伪类和伪元素

2) (主要) 伪元素: 设置对象的部分内容

伪元素	用途
:first-line	第一行文本的样式
:first-letter	第一个字符的样式
:first-child	此元素是其父元素的首个子元素

例如:

```
p:first-letter{color:#0f0} /* p标记内的第一个字符为绿色 */  
p:first-child { background-color:yellow; }  
/* 选择属于首个子元素的每个 p标记 */
```



2.6.3 光标样式

样式表中的`cursor`属性，设置鼠标在元素上时的光标类型。

例如：

`<div style="cursor:auto">浏览器光标</div>`

`<div style="cursor:crosshair">十字光标</div>`

`<div style="cursor:default">默认光标（箭头）</div>`

`<div style="cursor:pointer">指针光标（一只手）</div>`

`<div style="cursor:move">可移动光标</div>`

`<div style="cursor:e-resize">东resize光标</div>`

`<div style="cursor:ne-resize">北/东resize光标</div>`

`<div style="cursor:text">文本光标</div>`

`<div style="cursor:wait">等待光标（表或沙漏）</div>`

`<div style="cursor:help">帮助光标（问号或气球）</div>`



2.7 “层叠”与“继承”

样式表一个最重要的优点是一些简单的声明就可以改变大量文本的外观，这是样式继承带来的好处。继承是将样式定义从父元素传递给子元素的原则。

大多数样式会被继承（有些样式不会被继承，例如margin、padding等）。例如，为了设置默认的文本样式，可以给body元素定义一个样式规则如下：

```
body{color:#0f0; font-family:arial}
```

这样页面中所有的文本元素都将继承上面的定义。

事实上，所有在元素中嵌套的元素都会继承外层元素指定的样式属性值，就是会把很多层嵌套的样式“叠加”在一起。当样式表发生冲突时，总是以最后定义的样式为准。这也正是“层叠样式表”名称的由来，层叠就是继承。



2.7 CSS样式优先权

由于存在4种不同的样式使用方式，以及定义样式时的各种选择器，因此在CSS的样式定义中，难免存在重复定义。

最终到底是哪个样式定义起作用？就是所谓的样式优先权问题。下面的原则与前页的“继承”是一致的：

- **!important**：加!important关键字的样式最优先。例如：

```
p{background-color:rgb(255,0,0) !important}
```

- 局部优先：除!important外，内嵌样式表最优先。

```
<p style="background-color:#000">JavaScript</p>
```

- 具体优先：选择器指定得越具体，样式越优先。

```
body{background-color:#ff0000}
```

```
body p{background-color:#00ff00}
```

- 后者优先：同一元素同一样式的定义，后者优先。

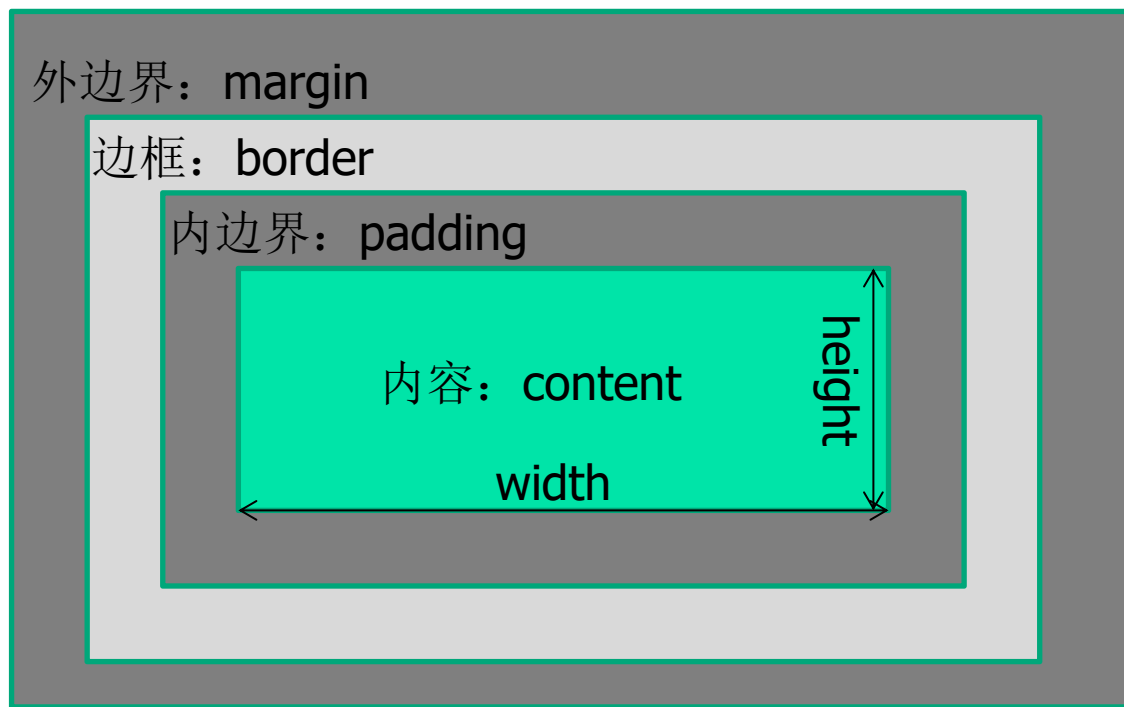
```
p{background-color:#ff0000}
```

```
p{background-color:#00ff00}
```

2.8 CSS页面布局与定位

2.8.1 CSS的元素模型

CSS将所有元素视为一个盒状物，称为盒(box)模型。由内容(content)、内边界(padding)、边框(border)和外边界(margin)构成。如下图：



其中：外边界是透明不可见的，用于控制元素边框之间的距离；内边界用于控制元素边框与内容之间的距离，其中有背景；边框可以设置的内容较多，包括宽度、颜色和线型等。三者都是上下左右可以分别设置的。



2.8 CSS页面布局与定位

2.8.2 块状元素与线内元素

HTML中的标记元素可以分为两类：

block块状元素：默认状态下独占一整行。例如：<p>,<h2>等。

in-line线内元素：默认状态下不独占一行。例如：<a>,等。

2.8.3 通用块状元素和通用线内元素

div：通用块状元素。将其中的子元素整体组合为一个块状元素。

span：通用线内元素。通过它可以对其间的文本（元素）进行设置。

通用块状元素**div**在**CSS**布局中起着关键作用。



3.8 CSS页面布局与定位

2.8.4 使用div进行页面布局

一般认为，实现页面的总体布局有两类方法：

table布局法：这是一种传统的布局方式，现在仍在大量使用。它使用**table**标记，设置边框宽度为0（即不显示边框），通过表格嵌套的方法，使得表格单元格的划分符合排版要求，然后将网页中的元素按排版要求分别放入表格的各个单元格，可以实现复杂的排版效果。这种方式的缺点是代码结构复杂，可读性低，维护成本高。

div布局法：使用**div**标记进行布局，利用**CSS**强大的样式定义功能，可以更简洁、更自由的控制页面的版式及样式。

2.8 CSS页面布局与定位

2.8.5 文档流

对于一个HTML页面，body元素下的所有元素，根据其先后顺序，组成一个元素流，这便是文档流。浏览器根据这些元素的顺序去显示它们在页面中的位置。文档流是浏览器的默认显示规则。

<!--例2-5.html： 一列固定宽度居中-->

<head>

<title>网络编程技术</title>

<link rel="stylesheet" type="text/css" href="2-5.css">

</head>

<body>

<div id="layout">

<h2>文档流</h2>

对于一个HTML页面，body元素下的所有元素，根据其先后顺序，组成一个元素流，这便是文档流。浏览器根据这些元素的顺序去显示它们在页面中的位置。

<p>文档流是浏览器的默认显示规则。 </p>

</div>

</body>

/*2-5.CSS*/

#layout{

background-color:#ccc;

border:2px solid #333;

width:300px;

height:300px;

margin:0px auto;

}



2.8 CSS页面布局与定位

2.8.6 浮动布局

CSS有两种网页元素布局方式：浮动布局和定位布局。

浮动（**float**）可以打破默认的文档流显示规则，通过浮动实现元素布局。浮动的块状元素（如果是线内元素则需要指定宽度）就好像一个个在水里的木箱尽量向上浮，而水平方向则根据**float**的两个值**left**、**right**，尽量向左浮动或向右浮动。现举两例：



2.8 CSS页面布局与定位

1) 两列自适应宽度

```
<!DOCTYPE ...>
<!--例： 2-6.html-->
<html>
<head>
  <title>网络编程技术</title>
  <link rel="stylesheet" type="text/css"
    href="2-6.css">
</head>
<body>
  <div id="left">左列</div>
  <div id="right">右列</div>
</body>
</html>
```

```
/*例： 2-6.CSS*/
#left{
  background-color:#ccc;
  border:2px solid #333;
  width:30%;
  height:300px;
  float:left;
}
#right{
  background-color:#ccc;
  border:2px solid #333;
  width:60%;
  height:300px;
  float:left;
}
```



2.8 CSS页面布局与定位

2) 两列固定宽度居中

```
<!DOCTYPE ...>
<!--例： 2-7.html-->
<html>
<head>
  <title>网络编程技术</title>
  <link rel="stylesheet" type="text/css"
    href="2-7.css">
</head>
<body>
<div id="layout">
  <div id="left">左列</div>
  <div id="right">右列</div>
</div>
</body>
</html>
```

```
/*例： 2-7.CSS*/
#left{
background-color:#ccc;
border:2px solid #333;
width:200px;
height:300px;
float:left;
}
#right{
background-color:#ccc;
border:2px solid #333;
width:200px;
height:300px;
float:left;
}
#layout {
width:408px;
margin:0 auto;
}
```



2.8 CSS页面布局与定位

3) 浮动的清理 (Clear)

指清除前面浮动元素的影响。既可以写在本元素的样式中（见右例），也可以专门添加一个div元素用于清理：

```
<div style="clear:both"></div>
```

```
#left{  
background-color:#ccc;  
border:2px solid #333;  
width:200px;  
height:300px;  
float:left;  
}  
#right{  
background-color:#ccc;  
border:2px solid #333;  
width:300px;  
height:300px;  
Clear:left;  
float:left;  
}
```



2.8 CSS页面布局与定位

2.8.7 定位布局

定位布局分为三种，由属性`position`的值确定。分别是：

- 相对定位： `position:relative`。
- 绝对定位： `position:absolute`。
- 固定定位： `position:fixed`。

相对定位指元素依据自己的原有位置进行定位，并保持自己原有站位的影响。具体位置由属性`top`、`right`、`bottom`、`left`的值确定（`top`优先于`bottom`；`left`优先于`right`，其实只需要两个）。

绝对定位元素被从文档流中抽出，失去对文档流的影响。若其父元素经过`position`定位，则其定位是相对于父元素的，否则是相对于页面的。具体位置由样式`top`、`right`、`bottom`、`left`的值确定（只需要两个）。

固定定位则是依据窗口的位置进行定位，元素不随滚动条滚动。具体位置同样由样式`top`、`right`、`bottom`、`left`的值确定（只需要两个）。



2.8 CSS页面布局与定位

2.8.7 定位布局

(1) 相对定位举例

<!-- 例2-8.html -->

```
<link rel="stylesheet" type="text/css"
href="2-8.css">
```

...

```
<body>
```

```
<div id="a">a</div>
```

```
<div id="b">b
```

```
    <div id="c">c</div>
```

```
    <div id="d">d</div>
```

```
</div>
```

```
</body>
```

```
/*例： 2-8.CSS*/
#a,#b,#c,#d{
background-color:#ccc;
border:2px solid #333;
width:100px;
height:100px;
float:left;
}
#b{
width:300px;
height:200px;
}
#c{
position:relative;
top:50px;
left:50px;
}
```



2.8 CSS页面布局与定位

2.8.7 定位布局

(2) 绝对定位举例

<!-- 例2-9.html -->

```
<link rel="stylesheet" type="text/css"
href="2-9.css">
```

...

```
<body>
```

```
<div id="a">a</div>
```

```
<div id="b">b
```

```
    <div id="c">c</div>
```

```
    <div id="d">d</div>
```

```
</div>
```

```
</body>
```

```
/*例： 2-9.CSS*/
#a,#b,#c,#d{
background-color:#ccc;
border:2px solid #333;
width:100px;
height:100px;
float:left;
}
#b{
width:3000px;
height:2000px;
position:relative;
}
#c{
position:absolute;
top:50px;
left:50px;
}
```



2.8 CSS页面布局与定位

2.8.7 定位布局

(3) 固定定位举例

<!-- 例2-10.html -->

```
<link rel="stylesheet" type="text/css"
href="3-10.css">
```

...

```
<body>
```

```
<div id="a">a</div>
```

```
<div id="b">b
```

```
    <div id="c">c</div>
```

```
    <div id="d">d</div>
```

```
</div>
```

```
</body>
```

```
/*例： 2-10.CSS*/
#a,#b,#c,#d{
background-color:#ccc;
border:2px solid #333;
width:100px;
height:100px;
float:left;
}
#b{
width:3000px;
height:2000px;
}
#c{
position:absolute;
top:50px;
left:50px;
}
```




2.8 CSS页面布局与定位

2.8.7 定位布局

(4) 设置层叠顺序

由于定位的出现，使得元素重叠的情况成为可能。这就出现谁遮挡谁的问题，即层叠顺序问题。

在默认规则下，使用`position`属性的元素要遮挡没有该属性的元素；在使用`position`属性的元素间，后面的元素遮挡前面的元素。

使用`z-index`属性，可以改变使用`position`属性元素间的层叠顺序。其值为一整数，值越大越具有显示优先权，即层叠顺序越高。但只有具有相同父元素的元素之间才具有可比性。

见例2-11.html。注意字母**b**原来被**float**排挤，现在已不受**float**影响。

```
/*例： 2-11.CSS*/
#a,#b,#c,#d{
background-color:#ccc;
border:2px solid #333;
width:100px;
height:100px;
float:left;
}
#b{
width:300px;
height:200px;
position:relative;
}
#c{
position:absolute;
top:50px;
left:50px;
z-index:10;
}
#d{
position:absolute;
z-index:1;
}
```



2.9 控制元素的显示

2.9.1 display属性

此属性最重要的属性值有3个：

- **block**: 该元素按块状元素显示。
- **inline**: 该元素按行内元素显示。
- **none**: 该元素不显示，而且被取消，即不占位。

例2-12.html: 下拉菜单的一种实现

<!-- 例2-12.html-->

<ul id="nav">

文章

CSS教程

DOM教程

XML教程

Flash教程

参考

XHTML

XML

CSS

Blog

全部

网页技术

UI技术

FLASH技术


```
/* 例2-12.css */
```

```
ul{ width:360px;
    padding:0;
    margin:0 auto;
    list-style:none;}
li{ float:left;
    width:120px;
    background-color:#eee;}
ul li a{
    display:block;
    font-family:"微软雅黑","宋体";
    font-size:12px;
    border:1px solid #000;
    padding:3px;
    text-decoration:none;
    text-align:center;
    color:#777;}
```

```
ul li a:hover{
    background-color:#fff;
    color:#000;}
li ul{
    width:120px;
    display:none;}
li:hover ul{
    display:block;}
ul li ul li a{
    border:1px solid #fff;
    color:#fff;
    background-color:#e12e6b;}
ul li ul li a:hover{
    background-color:#fff;
    color:#e12e6b;}
```



2.9.1 display属性

例2-12的说明：

- 1) 用无序列表（及其嵌套）生成菜单，通过**list-style:none**取消列表项符号。
- 2) 通过对列表项使用**float:left**样式，实现菜单项横排。
- 3) 对超链接<a>进行了详细设置：
 - 伪类:**hover**，利用超链接元素<a>实现按钮功能。伪类:**hover**是专门为超链接设计的，通过它可以改变鼠标在上时<a>元素的显示状态，实现按钮及菜单功能。
 - **display:block**，将超链接从一段文本转变为块状元素。这里的超链接是一段普通文本，转换为块状元素后就可以进一步设置其布局与显示，实现按钮功能。
 - **display:none**，它使元素不显示且消失（不占位），从而实现下拉菜单的功能。与其相关的是**visibility**属性（本课程未讲），但该属性只是控制元素的显示与否，其占位仍然有效。



2.9.2 对子元素的剪切控制

Overflow属性用来决定对超出元素边框范围的子元素的处理方式，可能的选择包括：

- 1) **visible**: 显示全部内容。这是缺省选项。
- 2) **hidden**: 隐藏超出显示范围的内容，也就是进行剪切。
- 3) **scroll**: 元素出现水平和垂直两个滚动条。
- 4) **auto**: 两个滚动条会根据内容是否超出范围分别自动出现。

例2-13.html: 剪切效果



2.9.2 对子元素的剪切控制

```
<!-- 例2-13.html-->
```

```
<div id="clipper">
```

```
<div id="box">
```

```
<div id="a">a</div>
```

```
<div id="b">b</div>
```

```
<div id="c">c</div>
```

```
<div id="d">d</div>
```

```
</div>
```

```
</div>
```

```
/* 例2-13.css */
```

```
#a,#b,#c,#d{
```

```
background-color:#ccc;
```

```
border:2px solid #333;
```

```
width:100px;
```

```
height:100px;
```

```
float:left;}
```

```
#box{
```

```
width:416px;
```

```
height:104px;
```

```
border:2px solid #333;
```

```
position:relative;
```

```
left:-50px;}
```

```
#clipper{
```

```
width:300px;
```

```
height:108px;
```

```
padding: 8px;
```

```
margin:0 auto;
```

```
border:2px solid #f00;
```

```
overflow:hidden;}
```



2.9.3 字体及段落样式控制

1) 字体样式:

- **color**: 指定文字颜色。例如: #fff、rgb(255,255,255)等。
- **font-family**: 指定字体族名或类名。有如下说明:
 - ① 族名 (**family-name**): 就是通常所说的“字体名”。包括“Arial”、“Times New Roman”、“宋体”、“黑体”等等。
 - ② 类名 (**generic family**): 一组具有统一外观的字体族的通用名。包括: **serif** (有衬线)、**sans-serif** (无衬线)、**cursive** (草书)、**monospace** (等宽)、**fantasy** (装饰)。
 - ③ 可指定多个属性值, 其间用逗号分隔, 依次递补候选。类名应当写在最后。
 - ④ 若名称中包含空格或中文, 则要以单引号或双引号括起来。
- **font-size**: 指定字体的大小。例如: 12px、30pt、2.3em、120%等
- **font-style**: 指定斜体样式。包括: **normal**、**italic**、**oblique**。
- **font-weight**: 指定字体的粗细。指定数值 (100、200、300、400、500、600、700、800、900) 或**normal**、**lighter**、**bold**、**bolder**。

注: **font-***共同指定了一种字体。



2.9.3 字体及段落样式控制

2) 段落样式（针对段落内所有元素）：

- **line-height**: 指定行高。例如：15px、120%。
- **letter-spacing**: 指定字符间距。例如：2em、15px。不能百分比。
- **text-indent**: 首行缩进。例如：30px、2em。
- **vertical-align**: 线内元素的垂直对齐方式。例如：top、bottom、middle。
- **text-align**: 水平对齐方式。包括：left、right、center、justify。
- **text-decoration**: 文字装饰。例如：underline、line-through、blink。
- **white-space**: 空格、Tab、换行的处理方式。例如：pre、nowrap。

2.9.3 字体及段落样式控制

3) 举例:

<!-- 例2-14.html-->

<div id="box">

<p id="p1">第一段, ..., 第一段。 </p>

<p id="p2">第二段, ..., 第二段。 </p>

<p id="p3">第三段, ..., 第三段。 </p>

<p id="p4">第四段,
第四段,

第四段。 </p>

<p id="p5">棕榈树:

</p>

</div>

/*例: 2-14.CSS*/

```
#box{ width:256px;  
background-color:#f6f6f6;  
padding:10px;  
margin:0 auto;}
```

```
P{ color: #666;  
font-family: "宋体";}
```

```
#p2{ color: #666;  
font-family: "微软雅黑","宋体";  
font-size: 12px;  
font-style: oblique;  
font-weight: bold;}
```

```
#p3{ text-indent: 2em; /*字单位*/  
line-height:200%;}
```

```
#p4{ letter-spacing:10px;  
white-space:pre; /*显示空格*/  
text-decoration: underline;}
```

```
#p5{ text-align:right;}
```

```
img{ vertical-align: middle;}
```



2.10 CSS3的新样式

CSS3增加了很多新的特性，这里只介绍其中的几个。

2.10.1 属性选择器

CSS2原有4个属性选择器：

选择器	例子	说明
[attribute]	[target]	选择具有 target 属性的元素。
[attribute=value]	[target=_blank]	选择 target="_blank" 的元素。
[attribute~value]	[title~flower]	选择title属性包含单词(空格分隔)"flower"的元素。
[attribute =value]	[lang =en]	选择lang属性值以 "en" 开头(连字符分隔)的元素。

CSS3增加了3个新的更方便的属性选择器：

选择器	例子	说明
[attribute^=value]	a[src^="https"]	选择src属性以"https"开头的<a>元素。
[attribute\$=value]	a[src\$=".pdf"]	选择src属性以".pdf"结尾的<a>元素。
[attribute*=value]	a[src*="abc"]	选择src属性中包含"abc"子串的<a>元素。



2.10 CSS3的新样式

2.10.2 圆角和阴影样式

1、圆角

过去，**border**属性可以指定边框的宽度、颜色、线型。现在，使用**border-radius**属性可以使边框具有圆角。见例2-15：

```
div
{
    border:2px solid;
    border-radius:25px;
}
```

2、阴影

box-shadow属性向框添加一个或多个阴影。见例2-16：

```
div
{
    box-shadow: 10px 10px 5px #888888;
}
```

其参数的具体意义请查手册。

这些功能在**CSS3**之前都需要使用图片技巧来实现。



2.10 CSS3的新样式

2.10.3 变换

1、2D变换

使用变换`translate()`、`rotate()`、`scale()`、`skew()`、`matrix()`，可以对元素进行平移，旋转、比例、倾斜和指定矩阵的二维变换。

见例2-17：

```
div {  
    transform: rotate(30deg);  
    -ms-transform: rotate(30deg); /* IE 9 */  
    -webkit-transform: rotate(30deg); /* Safari and Chrome */  
    -o-transform: rotate(30deg); /* Opera */  
    -moz-transform: rotate(30deg); /* Firefox */  
}
```



2.10 CSS3的新样式

2.10.3 变换（续）

其中的`matrix()`变换需要六个参数：

`transform: matrix(a,b,c,d,e,f);`

其实是给出了二维齐次变换矩阵：

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

它把各种二维变换组合在一起。例如：

- `transform: translate(dx+"px", dy+"px");`
等价于`transform: matrix(1, 0, 0, 1, dx, dy);`
- `transform: rotate(θ+"deg");`
等价于`transform: matrix(cosθ, sinθ, -sinθ, cosθ, 0, 0);`
- `transform: scale(sx, sy);`
等价于`transform: matrix(sx, 0, 0, sy, 0, 0);`
- `transform: skew(θx+"deg", θy+"deg");`
等价于`transform: matrix(1, tan(θy), tan(θx), 1, 0, 0);`



2.10 CSS3的新样式

2.10.3 变换（续）

2、3D变换

三维变换的种类有很多，但最常用的是旋转。二维旋转是绕Z轴的旋转，三维旋转可以绕X、Y或任意指定轴进行。见例2-18：

```
div {  
    transform: rotateY(130deg);  
    -webkit-transform: rotateY(130deg); /* Safari 和 Chrome */  
    -moz-transform: rotateY(130deg); /* Firefox */  
}
```

目前，三维变换并未被浏览器广泛支持。例如rotateX就没有多少浏览器支持。



2.10 CSS3的新样式

2.10.4 过渡

在不使用flash和Javascript的情况下，过渡可以实现在两种样式之间的渐变。见例2-19：

```
div {  
    transition: width 2s; /* 宽度方向上的渐变为两秒 */  
    -moz-transition: width 2s; /* Firefox 4 */  
    -webkit-transition: width 2s; /* Safari 和 Chrome */  
    -o-transition: width 2s; /* Opera */  
}  
div:hover { width:300px; }
```

过渡有不少参数可以设置，请参阅CSS3手册。如果忽略时长，默认值是0。



2.10 CSS3的新样式

2.10.5 动画

动画是在关键帧之间的过渡。动画开始的关键帧表示为**0%**或**from**，动画结束的关键帧表示为**100%**或**to**，其它关键帧位置只能用百分比表示。**@keyframes**规则用于定义动画的各个关键帧，**animation**属性用于定义动画的用时和方式等。见例2-20：

```
@keyframes myfirst
{
  from {background: red;}
    to {background: yellow;}
}
@-moz-keyframes myfirst /* Firefox */
{
  0% {background: red;}
  100% {background: yellow;}
}
...
```

这里的**from**、**to**和百分比也属于选择器。



2.10 CSS3的新样式

2.10.5 动画（续）

```
div
{
  animation: myfirst 5s;
  -moz-animation: myfirst 5s;
  ...
}
```

定义`animation`属性时至少需要指定动画名称和时长。如果忽略时长，默认值是0。



2.10 CSS3的新样式

2.10.6 媒体类型和媒体查询

@media规则可以使样式应用于特定的设备或媒介。

1、在CSS2中，@media可用于指定不同的媒体类型。例如：

➤ `<link href="css/reset.css" rel="stylesheet" media="screen">`

➤ `@media print { p.test {font-family:times,serif; font-size:10px} }`

其它的媒体类型还包括：all、handheld、tv等。

2、CSS3引入了媒体查询功能，它扩展了媒体类型功能。例如：

```
<link href="css/reset.css" rel="stylesheet"
      media="screen and (max-width: 600px)" >
```

```
@media screen and (min-width: 900px) {
  .class { background: #666; }
}
```

其它的媒体查询还包括：

(min-width:900px)

(orientation:portrait)等



第三章 JavaScript简介

3.1 概述

JavaScript是一种无类型的、解释型的程序设计语言。

客户端JavaScript是一种JavaScript的变体，它使用JavaScript语法，设计了针对浏览器和网页的变量和类型，可以在浏览器中运行。

客户端JavaScript使得网页不再是静态的HTML，而是能够与用户进行交互、对浏览器进行控制以及动态创建HTML的内容。

JavaScript是W3C(World Wide Web Consortium)制定的Web标准的组成部分。对HTML5来说，JavaScript显得比以往更加重要。



3.2 JavaScript语言的基本内容

3.2.1 词法结构

1、对大小写敏感

程序中标识符（包括：关键字、变量、函数名等）的大小写是有区别的。例如：**while**不能写为**While**。

2、语句末尾的分号

JavaScript语句以分号结束。尽管行末语句的分号可以省略（系统会自动补上），但这不是个好习惯，有时会引起误解，例如：

```
return
```

```
true
```

会被解释为：

```
return;
```

```
true;
```



3.2.1 词法结构

3、注释

JavaScript使用c语言的注释方法，包括：

区间注释：可跨行，即：`/*...*/`

行末注释：以`//`开始至行末，即：`//...`

4、常量

即在程序中直接写出的数据。例如：

1.2 `//数字1.2`

`"Hello world"` `//字符串`

true `//布尔值`

null `//空值，是对象类型的一个特殊值`

undefined `//未定义值`

`{x:1,y:2}` `//用于初始化对象`

`[1,2,3,4,5]` `//用于初始化数组`



3.2.1 词法结构

3、标识符

标识符就是名字。在JavaScript中，标识符用来命名变量和函数。还可用标签，来作标识语句位置。

标识符的第一个字符必须是字母、下划线或美元符号。其它字符可以是字母、数字、下划线或美元符号。下面列出的都是合法标识符：

`i,x1,y_1,_1,$m`

注意：标识符不能与JavaScript保留字（保留字：在JavaScript中有特殊意义，而被保留的名字）、JavaScript全局变量或全局函数同名。



3.2.2 数据类型和值

JavaScript的变量不分类型，并不意味着JavaScript的数据不分类型，而是指JavaScript的变量可以存储各种数据类型。

JavaScript数据分为两大类：基本数据类型和复合数据类型。

基本数据类型分为三种：数字、字符串和布尔值。（此外还支持两个小的数据类型：**null**(空)和**undefined**(未定义)，他们各自只定义了一个值）。

复合数据类型则包括：对象、数组和函数。

JavaScript已经预先定义了一些类，例如：表示日期的**Date**类可以创建不同的日期对象。

JavaScript也提供了创建自定义类的方法，但由于时间所限，本课程不予介绍，只使用JavaScript的预定义类。



3.2.2 数据类型和值

1、数字

与某些常用的语言（例如C语言）不同，JavaScript并不区分整形数值和浮点型数值。在内部，JavaScript采用IEEE 754标准定义的64位浮点格式（就是C语言中的double格式）表示数字。

它能表示的最大值是 $\pm 1.7976931348623157 \times 10^{303}$ ，最小值为 $\pm 5 \times 10^{-324}$ 。能精确表达的整数范围在-9007199254740992（ -2^{53} ）到9007199254740992（ 2^{53} ）之间。

数字的表示形式包括：

- 十进制整数。例如：0, 3, 10000
- 十六进制整数。例如：0xff, 0Xabc123, 0X123DEF
- 十进制浮点数。例如：3.14, .33333
- 科学计数法。例如：6.0e23, 1.54E-32, 5E+32



3.2.2 数据类型和值

JavaScript还定义了一些特殊的数值常量:

Infinity, -Infinity: 正负无穷

NaN: 非数字。例如: 0/0

Number.MAX_VALUE: 最大数字

Number.MIN_VALUE: 最小数字

Number.NaN: 非数字。同上NaN

Number.POSITIVE_INFINITY: 正无穷。同上Infinity

Number.NEGATIVE_INFINITY: 负无穷。同上- Infinity



3.2.2 数据类型和值

2、字符串

即字符序列。在JavaScript中没有字符类型，字符就是一个字符的字符串。

(1) 单引号和双引号

字符串常量是用单引号或双引号括起来的字符序列：由单引号定界的字符串常量中可以包含双引号；由双引号定界的字符串常量中可以包含单引号。*而且字符串常量必须写在一行中。*

在客户端程序设计中，JavaScript代码中常包含HTML代码，HTML代码中也常常包含JavaScript代码。和JavaScript一样，HTML也使用单引号或双引号来定界字符串。因此在同时使用JavaScript和HTML时，最好对JavaScript采用一种引用方式，对HTML采用另一种引用方式，例如：

```
<a href="" onclick="alert('Thank you')">Click me</a>
```

它在JavaScript代码中使用单引号；在HTML代码中使用双引号。



3.2.2 数据类型和值

(2) 转义字符

当字符串中需要无法直接出现的字符时（例如：字符串必须在一行中写完，因此不能直接出现换行符），就需要使用所谓的转义字符（转义：意思转变了）。转义字符以反斜线“\”开始，常用的转义字符包括：

转义字符	意义	转义字符	意义
\0	空字符	\r	回车符
\b	退格符	\"	双引号
\t	水平制表符	\'	单引号
\n	换行符	\\	反斜线
\v	垂直制表符	\xXX	两位16进制数XX表示的Latin-1字符
\f	换页符	\uXXXX	四位16进制数表示的Unicode字符

注意：如果“\”后面的字符不匹配，那就构不成转义字符，反斜线将被忽略。例如：\#等价于#。



3.2.2 数据类型和值

(3) 字符串的使用（举例）

- 连接字符串。例如：`s="Hello,"+"world";`
- 字符串的长度。例如：`len=s.length;`
- 获得一个字符。例如：`first=s.charAt(0);last=s.charAt(s.length-1);`
- 获得一个子串。例如：`sub=s.substring(1,4);`
- 查找字符位置。例如：`i=s.indexOf('e');`

上面的**s**为字符串变量，字符串变量的属性和方法还有很多。

字符串是一个很特殊的JavaScript数据类型。它的使用方式与对象十分相似。



3.2.2 数据类型和值

3、布尔值

布尔值表示真假（或对错）。它只有两个值“true”和“false”。

例如：

```
if( a==4 )
```

```
    b=b+1;
```

```
else
```

```
    a=a+1;
```

很多语言（包括JavaScript、c）把非零（特别是1）当做true，把0当做false。



3.2.2 数据类型和值

4、函数

一个函数就是一个可执行的JavaScript代码段。JavaScript将函数看做一种特殊类型的对象，这种类型叫Function。

因此，和简单类型一样，在JavaScript中，函数有常量、变量、可以作为参数传递、也可以作为对象的属性（成员）。

1) 函数的三种定义方式

① 标准定义（语句）

例如：`function square(x) {return x*x};`

这种定义方式由关键字function开始，后面跟随的是：

- 函数名
- 一个用圆括号括起来的参数表，参数间用逗号隔开。无参数可省。
- 用大括号括起来的函数体，其中是JavaScript语句。

注意：上面的函数中用return语句返回了一个值，如果没有return语句，则函数返回undefined。



3.2.2 数据类型和值

② 函数常量定义

例如：`var square = function(x) {return x*x;};`
`var tensquared = (function(x) {return x*x;})(10);`

“=”号右边为函数常量，它的形式和标准函数定义一样，只是没有函数名。因此又被称为匿名函数或**lambda**函数。它与标准定义的区别是，函数常量可以直接出现在表达式中，而标准定义本身就是语句。

③ 构造函数定义

例如：`var square = new Function("x", "y", "return x*y;");`

Function()构造函数可以有任意多个字符串参数，最后一个字符串是函数的主体，其它参数构成函数的参数表。

构造函数创建的函数也没有函数名，可以像函数常量一样出现在表达式中。

使用构造函数的好处是：通过改变字符串的内容实现，可以动态创建函数的内容。这点有时非常灵活。



3.2.2 数据类型和值

2) 函数的调用

函数被定义后就可以反复使用，而且JavaScript已经提供了许多标准的预定义函数（例如：**Math.sin()**）。

```
y = square(x);
```

```
z = Math.sin(x);
```

3) 函数名是变量

在标准函数定义中指定的函数名是用来保存函数的变量。



3.2.2 数据类型和值

5、对象

对象(object)是已命名的数据的集合，这些已命名的数据通常作为对象的属性来引用。要引用一个对象的属性，就必须引用这个对象，在其后加“.”号（运算符）和属性名。例如：

```
image.width
```

```
image.height
```

```
document.myform.button
```

属性可以是任何数据类型，包括函数。当函数作为某个属性的值时，这个属性称为方法，函数名称为方法名。例如：

```
document.write("This is a test!");
```

另外，JavaScript中的对象还可以作为关联数组（关联数组：用字符串做下标的数组）使用。例如：

```
image["width"]
```

```
image["height"]
```



3.2.2 数据类型和值

1) 对象的两种创建方式:

① 使用构造函数。例如:

```
var o = new Object();
```

```
var now = new Date();
```

② 使用对象常量初始化。例如:

```
var point = {x:2.3, y:-1.2}; // “=” 右边为对象常量
```

```
var rectangle = {upperLeft:{x:2,y:2},  
    lowerRight:{x:4,y:4}}; //嵌套
```

```
var square = {upperLeft:{x:point.x, y:point.y},  
    lowerRight:{x:(point.x+size),y:(point.y+size)}}; //使用表达式
```

2) 对象的使用

对象创建后，就可以设计并使用它的属性了。例如:

```
point.z = 3.6; //添加属性
```

```
x=square.upperLeft.x; //使用属性
```



3.2.2 数据类型和值

6、数组

数组(**array**)和对象一样也是数值的集合。所不同的是，对象中的每个数值都有一个名字，而数组中的每个数值则对应一个下标。其实数组是一种特殊类型的对象。

数组（非关联数组）的下标是从**0**开始的非负整数，所以**a[2]**表示数组**a**中的第三个元素。

数组中可以存放任意类型的数据，而且由于**JavaScript**是一种无类型语言，因此数组元素不必具有相同的类型。

JavaScript不支持多维数组，但由于数组元素还可以是数组，所以可以实现多维数组的效果。例如：

```
var mat=[[1,2,3],[4,5,6],[7,8,9]];
document.write("mat[1][2]="+mat[1][2]); //mat[1][2]=6
```



3.2.2 数据类型和值

1) 数组的两种创建方式:

① 使用构造函数。例如:

```
var a = new Array();
```

```
var b = new Array(1.2, "JavaScript", true, {x:1, y:2});
```

```
var c = new Array(10); //只有一个参数时, 参数指定数组长度
```

② 使用数组常量初始化数组。例如:

```
var b = [1.2, "JavaScript", true, {x:1, y:2}]; //右边为数组常量
```

```
var matrix = [[1,2,3],[4,5,6],[7,8,9]]; //常量嵌套
```

```
var table = [x, x+1, x+2, x+3]; //使用表达式
```

```
var sparse = [1,,,,5]; //5个元素, 3个未定义
```

2) 数组的使用

一旦创建数组后, 就可以方便地使用。例如:

```
x=b[0];
```

```
y=matrix[2][2]; //多维数组效果
```



3.2.2 数据类型和值

7、关于函数、对象和数组的说明

在JavaScript中，函数和数组都是对象。它们都有大量的方法和属性，而且非常有用（例如对数组可以方便地进行排序）。由于课时所限，本课程没有介绍。



3.2.3 关于变量

变量是一个和数值**相关**的名字。我们说变量“存储”了那个值。

1、JavaScript变量不分类型

与c语言完全不同，JavaScript变量中可以存储任何类型的数据：

```
i = 10; i="ten";
```

2、变量的声明

在JavaScript程序中，使用变量前，应当用关键字**var**进行显式声明。例如：

```
var i=0, sum=0, message="hello";
```

如果没有显式声明就使用变量名，则：

- 尝试读取一个未声明变量的值将导致一个错误。
- 若给一个未声明的变量赋值，JavaScript将隐式声明一个全局变量。

在同一范围反复声明一个变量是合法的，而且不会改变变量的值，如果重复的声明有一个初始值，则仅相当于一个赋值语句。



3.2.3 关于变量

3、变量的作用域

变量的作用域就是变量的有效范围。

在函数中声明的变量（包括函数的参数）称为局部变量；在函数外声明的变量称为全局变量。隐式声明不管是否发生在函数中，其声明的变量都是全局变量。

全局变量的作用域是全局的，即从定义点处开始，在代码中处处有效；局部变量的作用域是局部性的，即只在函数内部有效。

局部变量的优先级高于全局变量：如果一个局部变量的名字与某个全局变量的名字相同，局部变量就隐藏了全局变量。

局部变量的定义应该放在函数的最前面。



3.2.3 关于变量

5、变量存储的内容

1) 基本类型与引用类型

可以将数据类型分为两组：基本类型和引用类型。

① 基本类型：包括数值、布尔值、**null**和**undefined**。

它们的特点是在内存中具有固定的大小。

② 引用类型：就是对象（包括函数、数组等）。

它们的特点是在内存中的大小不固定。

2) 变量的存储内容

① 对基本类型，变量存储的是值。

② 对引用类型，变量存储的是引用，也就是通常说的指针。

3) 字符串的类型

字符串的类型是个特例：它既不是基本类型，因为不同字符串没有固定的大小；也不是引用类型，因为不可以通过变量名改变字符串的内容和大小。可以把它看做是不可变引用类型。



3.2.4 表达式和运算符

1、表达式

表达式（**expression**）是JavaScript的一个“短语”，它可以生成一个值。最简单的表达式是一个常量或一个变量。利用运算符可以将简单的表达式合并为复杂的表达式。

2、运算符

JavaScript有同c语言类似的运算符。运算符有符号型的，例如：**+**、**-**、*****、**/**等；也有关键字型的，例如：**new**、**delete**、**typeof**、**instanceof**等，但它们都是运算符。

涉及运算符的概念包括：

1) 运算数的个数

根据运算符需要的运算数的个数，可以将运算符分为：

- 一元运算符：例如：“**-**”（负号运算符）
- 二元运算符：最多的运算符，例如：“**+**”、“**-**”、“*****”、“**/**”
- 三元运算符：例如：“**? :** ”（条件运算符）



3.2.4 表达式和运算符

2) 运算数的类型

不同的运算符需要不同类型的运算数，计算结果的数据类型也不尽相同。在有可能的情况下，JavaScript会将运算数转换为适当的类型。例如：

```
"a"*"b"; //非法
```

```
"3"*"5"; //合法，结果为数字15
```

3) 运算符的优先级

优先级控制执行操作的顺序。考虑表达式：

```
w = x+y*z;
```

由于乘法运算符“*”的优先级高于加法运算符“+”的优先级，所以先执行乘法，再执行加法。赋值运算符“=”的优先级最低，所以在右边的操作都结束后才会被执行。

使用括号可以提高运算符的优先级。例如：

```
w = (x+y)*z;
```

运算符的优先级定义符合常规，必要时可查阅手册。



3.2.4 表达式和运算符

4) 运算符的结合性

运算符的结合性有两种：左结合和右结合。

结合性用来决定当优先级相等时，执行操作的顺序。

左结合表示操作由左向右执行。例如：

$w = x + y + z;$

右结合表示操作由右向左执行。例如：

$w = x = y = z;$



3.2.4 表达式和运算符

4、一些特殊的运算符

1) in

用法：字符串 in 对象

用途：检查对象中是否存在以字符串为名字的属性（包括方法）。

举例：`var point={x:1,y:2}; var b="x" in point; //返回true`

2) void

用法：`void` 任意

用途：舍弃运算数的值，返回undefined。

举例：`var a=0; void a; //返回undefined，但并没有改变a的值。`

3) typeof

用法：`typeof` 任意

用途：返回运算数类型的字符串。

举例：`var i=1; var s = typeof i; //返回 “number”`



3.2.4 表达式和运算符

4) instanceof

用法：对象 instanceof 对象类名

用途：判断左边对象是否是右边类的一个实例。

举例：var a = [1,2,3];

var b = a instanceof Array; //值为true

var b = a instanceof Object; //值为true，数组都是对象

5) new

用法：new 构造函数(参数表)

用途：创建一个指定类型的新对象。

举例：var d = new Date();



3.2.5 语句

语句是完整的句子或命令，JavaScript程序就是一些语句的集合。语句的形式多种多样：

1、表达式语句

在JavaScript中，最简单的语句莫过于表达式了。例如：

```
var a=0; a++;
```

2、复合语句

将一些语句用花括号括起来，就形成了一个复合语句，这样它就可以出现在需要使用单个语句的地方。例如：

```
{x = Math.PI; cx = Math.cos(x);}
```

注意：花括号的后面不需要有分号，而且这不是不良习惯。



3.2.5 语句

3、if语句

该语句用来实现有条件地选择执行语句，与c语言中的if语句用法基本相同。例如：

```
if(n==1) {  
    //执行代码块#1  
}  
else if(n==2) {  
    //执行代码块#2  
}  
else {  
    //执行代码块#3  
}
```




3.2.5 语句

4、switch语句

用来实现简单的多分支结构，该语句与c语言中的switch语句用法基本相同。例如：

```
switch(n) {  
    case 1:  
        //执行代码块#1  
        break;  
    case 2:  
        //执行代码块#2  
        break;  
    default:  
        //执行代码块#3  
}
```

注意：switch后的表达式与case后的常量表达式的“匹配”指“===”。



3.2.5 语句

5、while语句

循环语句，该语句与c语言中的while语句用法基本相同。例如：

```
var count = 0;
while(count < 10) {
    document.write(count+"<br>");
    count++;
}
```

6、do/while语句

至少执行一次的循环语句，与c语言中的do/while语句用法基本相同。

```
var count = 0;
do {
    document.write(count+"<br>");
    count++;
} while(count < 10) ;
```



3.2.5 语句

7、for语句

最常用的循环语句，与c语言中的for语句用法基本相同。例如：

```
for(var count=0; count < 10, count++)  
    document.write(count+"<br>");
```

8、for/in语句

针对对象（包括数组）的循环语句，用于枚举对象所有属性。C和C++语言中都没有响应的语句。其语法为：

```
for(变量 in 对象)
```

```
    循环体; // 或称子语句
```

当循环体执行之前，对象的一个属性名被作为字符串赋值给变量。在循环体内部可以使用这个变量和“[]”运算符来查询对象的属性值。例如，下面的for/in循环将输出一个对象和一个数组的所有属性名及它的值：



3.2.5 语句

```
var obj = {x:1, y:2, z:3};  
var ary = [1, 2, 3];  
for(var prop in obj)  
    document.write("name:"+prop+ "; value:"+obj[prop]+"<br>");  
for(var prop in ary)  
    document.write("index:"+prop+ "; value:"+ary[prop]+"<br>");
```

9、标签语句

任何语句都可以通过在它前面加上标识符和冒号来标记，成为带有标签的语句。这样在程序的任何地方都可以通过这个标签来引用它。由于标签名不同于变量名和函数名，所以标签名可以和变量或函数同名，且互不影响。标签语句的语法为：

标签：语句

例如：

```
p1: for(i=0; i<10;i++){...};
```



3.2.5 语句

10、break语句

break语句用于中断**switch**语句或循环语句的执行。缺省情况下，其中断的是最内层**switch**语句或循环语句；当**break**后有语句标签时，其中断的是标签所指定的语句。例如：

```
for(i=0; i<a.length; i++){  
    if(a[i] == target) break;}
```

11、continue语句

continue语句用于中断循环语句的当前循环（至当前循环体的末端），并开始下次循环。缺省情况下，其中断的是最内层循环语句的循环体；当**continue**后有语句标签时，其中断的是标签所指定语句的循环体。例如：

```
for(i=0; i<data.length; i++){  
    if(data[i] == null) continue;  
    total += data[i];  
}
```



3.2.5 语句

12、return语句

用在函数体中，用于返回函数的值或中断函数的执行。用法与c语言中的**return**语句基本相同。没有参数的**return**语句返回**undefined**。



3.3 客户端JavaScript

前面介绍的是作为通用语言的JavaScript核心，客户端JavaScript是其在浏览器环境下的一个具体实现。

3.3.1 Web浏览器中的JavaScript

1、客户端JavaScript特点

1) 顶层Window对象

在客户端JavaScript中，表示HTML文档的是Document对象，Window对象表示显示该文档的窗口（或框架）。

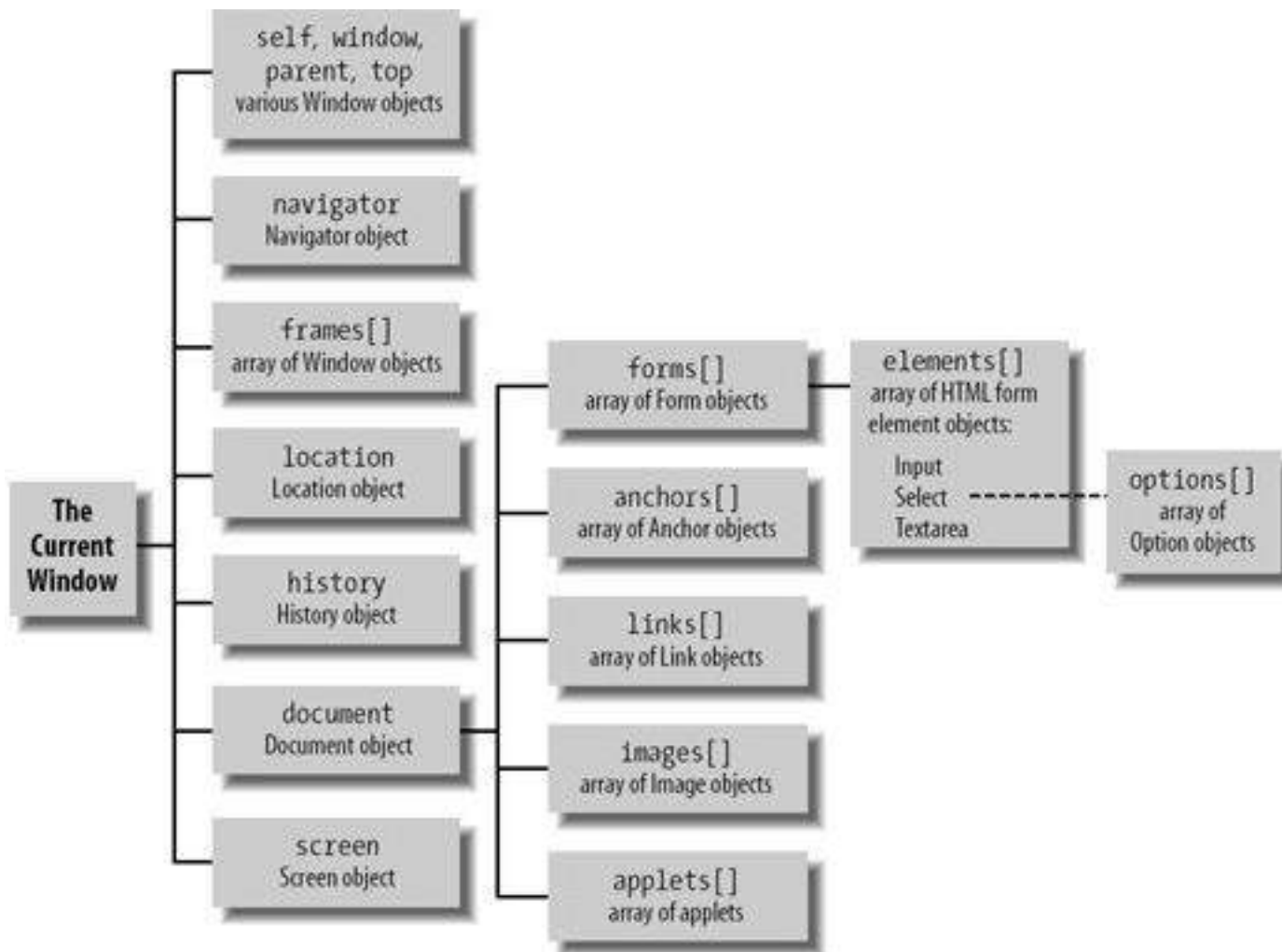
Window对象定义了一些属性和方法用于控制浏览器窗口，也定义了一些属性用来表示一些重要的对象（例如Document对象）。而且，Window对象定义了两个表示自己的属性：`window`和`self`。

Window对象是顶层对象（顶层`this`），所有定义的全局变量都是其属性。所以下面两行代码的功能是相同的：

```
var answer = 42;  
window.answer = 42;
```

3.3.1 Web浏览器中的JavaScript

2) 客户端JavaScript对象层次结构及（0级）文档对象模型（DOM）





3.3.1 Web浏览器中的JavaScript

上图中的许多对象都继承自**Document**对象，这个较大的客户端对象子树被称为文档对象模型（**DOM**）。它构成了文档功能的基础层次，被称为**0级文档对象模型**。由于已经被大多数浏览器实现，已经成为事实上的标准，程序员们可以非常信赖地在所有的浏览器上使用它。

3) 事件驱动的程序设计模型

在客户端**JavaScript**中，**Web**浏览器使用事件（**event**）来通知程序用户有输入。事件的类型有很多，例如键盘事件，鼠标事件等等。当一个事件发生时，**Web**浏览器会先尝试调用一个适合的事件处理程序来响应这个事件。

因此，要编写一个动态的、交互性的客户端**JavaScript**程序，必须先定义一些适当的事件处理程序，并将它们注册到系统中，这样浏览器才能在适当的时候调用它们。

这是典型的事件驱动程序设计模型。在大部分时间中，程序根本就不运行，而是等待用户输入，由系统调用它的某一个事件处理程序。



3.3.1 Web浏览器中的JavaScript

2、在HTML中嵌入JavaScript

在HTML中嵌入JavaScript代码有很多方法，例如：

- ① 放置在标记对<script>和</script>之间。
- ② 放置在由标记<script>的src属性指定的外部文件中。
- ③ 作为HTML标签的属性（例如onclick、onmouseover等）的值，放置在事件处理程序中。
- ④ 写在超链接中

1) <script>标记

客户端JavaScript脚本是HTML文件的一部分，放置在标记对<script>和</script>之间。 <script>标记既可以出现在HTML文档的<head>部分，也可以出现在<body>部分。但不可嵌套和交叉。

```
<script>
```

```
// Your JavaScript code goes here
```

```
</script>
```



3.3.1 Web浏览器中的JavaScript

一个HTML文档可以包含任意多个不重叠的<script>、</script>标记对，这些独立脚本的执行顺序就是它们在文档中出现的顺序。尽管在装载和解析一个HTML文档的过程中，各个脚本在不同时刻执行，但它们是同一个JavaScript程序的组成部分。在一个脚本块中定义的函数和变量，适用于随后出现的同一个HTML文件中的所有脚本块。关键的环境是这个HTML文档，而不是脚本块。

例如，可以将下面的一行脚本放到HTML文档某个地方：

```
<script>var x=1;</script>
```

然后，你可以引用变量x，即使这个引用出现在另一个脚本块中：

```
<script>document.write(x);</script>
```

document.write()是个重要且常用的方法，它将把自己的输出插入到脚本所在的文档位置。当脚本执行完毕时，HTML解析器将继续解析文档，并开始解析document.write()方法生成的文本。

下例是一个HTML文档，它包含一个简单的JavaScript程序：



3.3.1 Web浏览器中的JavaScript

```
<html><!--3-1.html-->
<head>
  <title>Today's Date</title>
  <script language="JavaScript"> // 定义一个函数供以后使用
    function print_todays_date( ) {
      var d = new Date( ); // 得到当前时间和日期
      document.write(d.toLocaleString( )); // 插入到文档
    }
  </script>
</head>
<body>
  The date and time are:<br>
  <script language="JavaScript"> // 调用上面定义的函数
    print_todays_date( );
  </script>
</body>
</html>
```



3.3.1 Web浏览器中的JavaScript

`<script>`标记的`language`和`type`属性:

上例中出现了`<script>`标记的`language`属性。该属性是可选的，用于指定书写脚本的语言。过去曾经有`<script language="VBScript">`，但现在已经基本不用了。

新规范主张使用`type`属性来区分，例如：

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

但实际上什么也不用写，现在已经没有第二种选择了！



3.3.1 Web浏览器中的JavaScript

2) 包含JavaScript文件

`<script>`标记支持`src`属性，其值用来指定一个包含JavaScript代码文件的URL。它的用法如下：

```
<script src="../../scripts/util.js"></script>
```

例如：

```
<html><!--3-2.html-->
```

```
<head>
```

```
  <title>Today's Date</title>
```

```
  <script src="3-2.js"></script>
```

```
</head>
```

```
<body>
```

```
  The date and time are:<br>
```

```
  <script language="JavaScript">
```

```
    print_todays_date( );
```

```
  </script>
```

```
</body>
```

```
</html>
```

```
// 3-2.js
```

```
function print_todays_date( ) {
```

```
  var d = new Date( );
```

```
  document.write(d.toLocaleString( ));
```

```
}
```



3.3.1 Web浏览器中的JavaScript

具有src属性的<script>标记的行为就像指定的JavaScript文件内容直接出现在标记<script>和</script>之间一样，浏览器会忽略标记之间的代码，但结束标记</script>不可省略。

使用<script>标记的好处有很多：

- 可以把大型的JavaScript代码块移出HTML文件，从而简化了HTML文档。
- 将公共代码放置在共享文件中，可以减少磁盘占用，并提高维护效率。
- 多个页面共享JavaScript文件时，该文件将被缓存，读取效率会更高。
- 通过设置URL，一个Web服务器可以使用另一个Web服务器输出的代码。



3.3.1 Web浏览器中的JavaScript

3) 事件处理程序

由于客户端JavaScript的事件是由HTML元素引发的，因此事件处理程序被定义为这些元素的属性值。例如：

```
<html><!--3-3.html-->
<head>
  <title>Welcome to Sina!</title>
</head>
<body>
  <a href="http://www.sina.com"
    onclick="alert('You are cheated!'); return false;">
    Welcome to Sina!
  </a>
</body>
</html>
```




3.3.1 Web浏览器中的JavaScript

这里的HTML元素是超链接，其**onclick**属性的值是一个字符串，其中可以包含多个JavaScript语句。当指定的事件（这里是点击）发生时，字符串中的JavaScript代码就会被执行。

为了减少JavaScript与HTML的混合，如果事件处理程序的语句多于一条或两条，通常将事件处理程序定义为标记**<script>**和**</script>**之间的一个函数，然后在事件处理程序中调用这个函数。

最常见的代表事件的属性包括：

onclick,onmousedown,onmouseup,onmouseover,onmouseout

onchange,onsubmit,onreset

事件处理程序的返回值非常重要，返回**true**意味着去执行与元素相关的默认动作（对标记**<a>**，**onclick**默认将执行超链接），返回**false**则不执行。当然元素可以没有默认动作。

关于事件处理程序，需要记住的重要一点是：在事件处理程序的代码中，关键字**this**引用触发该事件的文档元素。



3.3.1 Web浏览器中的JavaScript

4) 写在URL处的JavaScript

伪协议说明符`javascript:`加后面的JavaScript代码，可以用作URL。这种URL被看做是单行代码，通常用于代替事件处理程序，因此语句后一定要加“`;`”号，而且需要用`/*...*/`注释代替`//`注释。例如：

```
<a href='javascript:function()'
```

可以和`onclick`能起到同样的效果。

但W3C标准不推荐在`href`里面执行`javascript`语句，在某些浏览器中可能会引发其他不必要的事件，造成非预期效果。



3.3.2 窗口和框架

窗口和框架都由Window对象表示，它定义了许多方法和属性：

1、方法alert() , confirm()和prompt()

这三个Window方法用来显示简单的对话框。 **alert()** 用于向用户提示信息； **confirm()**要求用户点击**OK**或**Cancel**按钮来确认或取消某个操作； **prompt()**要求用户输入一个字符串。

2、 setTimeout()、clearTimeout()与setInterval()和clearInterval()

这两组函数都是用来控制代码的定期执行。

- 1) **setTimeout()**有两个参数，第一个参数是代码字符串，第二个参数是时间的毫秒数。它指定第一个参数的代码，在第二个参数的时间后运行一次。该方法返回一个id，用于由**clearTimeout()** 取消其设置。当然，如果代码已经执行，这个id就无用了，也用不着取消。
- 2) **setInterval()**的参数与**setTimeout()**相同，代码将按指定的时间间隔反复执行。它也返回一个id，用于由**clearInterval()**取消代码的执行。



3.3.2 窗口和框架

3、navigator属性

navigator属性引用一个**Navigator**对象，它包含Web浏览器的大量信息。主要包括：

- ① **appName**: Web浏览器的简单名称。
- ② **appVersion**: 浏览器的版本号和（或）其它版本信息。
- ③ **userAgent**: 浏览器在它发送的HTTP头中包含的USER-AGENT 字符串。通常包含**appName**和**appVersion**的所有信息。
- ④ **appName**: 浏览器的代码名。 Netscape和IE都是Mozilla
- ⑤ **Platform**: 运行浏览器的系统平台。例如：Win32



3.3.2 窗口和框架

4、screen属性

screen属性引用一个**Screen**对象，它提供有关用户显示器大小和可用颜色数量的信息。例如：

- ① **width**、**height**：显示器的宽和高，单位是像素。
- ② **availWidth**、**availHeight**：可用宽度和高度，排除了任务栏等。
- ③ **colorDepth**：可用位面数。



3.3.2 窗口和框架

5、对窗口的控制方法

- ① `open()`: 打开并返回一个新窗口，显示指定URL的内容。
- ② `close()`: 关闭窗口。
- ③ `moveTo()`、`moveBy()`: 移动窗口。
- ④ `resizeTo()`、`resizeBy()`: 改变窗口大小。
- ⑤ `focus()`、`blur()`: 获取或放弃键盘交点
- ⑥ `scrollTo()`、`scrollBy()`: 滚动窗口或框架中的文档。

6、`opener`属性

Window对象的 `open()`方法返回新创建的窗口的Window对象，而每个Window对象都有一个`opener`属性，引用打开它的原始窗口。由此，两个窗口就可以相互引用，彼此都可以读取对方的属性或调用对方的方法。如果窗口是由用户，而不是有JavaScript代码创建的，那么它的`opener`属性就是`null`。



3.3.2 窗口和框架

7、location属性

location属性引用一个Location对象，它代表该窗口中当前显示文档的URL。它的各个属性代表了URL的不同部分，例如：

- ① href: 完整的URL字符串。例如：`http://www.oreilly.com:1234/catalog/search.html?q=JavaScript&m=10#results`
- ② protocol: 协议。例如：`"http:"`。
- ③ host: 主机名和端口号。例如：`"www.oreilly.com:1234"`
- ④ hostname : 主机名。例如：`"www.oreilly.com"`
- ⑤ port: 端口号。例如：`"1234"`。
- ⑥ pathname: 路径部分。例如：`"/catalog/search.html"`。
- ⑦ search: ?号之后的那部分URL，即URL的参数部分。例如：`"?q=JavaScript&m=10"`。
- ⑧ hash: URL的锚部分，用于定位在文档的特定位置。例如：`"#result"`。



3.3.2 窗口和框架

改变Location对象的任何属性或直接将URL字符串赋值给对象，都将导致新的页面被装入。

另外，Location对象还定义了两个方法：

- ① **reload()**: 重新装入页面。
- ② **replace()**: 用新的页面代替当前页面，包括在历史记录中的位置。



3.3.2 窗口和框架

8、history属性

history属性引用一个**History**对象，这个对象保存着该窗口访问过的页面历史列表。但这个历史只能用**History**对象的三个方法去访问：

- ① **back()**: 跳转到前一页面的文档，就像按了浏览器的**back**按钮。
- ② **forward()**: 向下一页面跳转，就像按了浏览器的**forward**按钮。
- ③ **go()**: 向前或向后跳转参数指定个页面。



3.3.2 窗口和框架

9、frames[]、parent和 top属性

- ① **frames[]**: 该属性是一个Window对象的数组，其中的每个元素代表这个窗口中包含的一个子框架窗口。如果这个窗口中没有框架，则**frames[]**数组为空，**frames.length**的值为0。例如：**frames[0]**表示第一个子框架窗口，**frames[1]**表示第二个子框架窗口。
- ② **parent**: 子框架窗口的父窗口（与**opener**意义不同）。如果不是子框架窗口，这个属性就是窗口本身，即**parent==self**。
- ③ **top**: 子框架窗口的顶级父窗口。如果不是子框架窗口，则**top==self**；如果是第一层子框架窗口，则**top==parent**；如果是第二层子框架窗口，则**top== parent .parent**。

下面是一些典型的相互引用：

`frames[1].frames[2]` //第二个框架的第三个子框架

`parent.frames[1]` //兄弟框架



3.3.2 窗口和框架

10、name属性

每个窗口都可以有一个名字。例如：`open()`方法的第二个参数就是新开窗口的名称；HTML标记`<frame>`的name属性值也是框架的名称。

窗口或框架名称的重要作用是作为标记`<a>`、`<map>`和`<form>`的属性`target`的值。这就告诉浏览器将激活链接、点击图像或提交表单的结果显示在该窗口中。如果没有名为name值的窗口存在，就新开一个名为name值的窗口显示目标网页。

如果给框架窗口命了名，则这个名称将成为其父窗口对象的一个属性，可以直接用这个属性引用该框架窗口。因此，可以使用下面的HTML代码创建一个框架窗口，然后在兄弟窗口中引用它：

```
<frame name="table_of_contents" src="toc.html">
```

```
parent.table_of_contents //兄弟窗口中的代码，易读易理解
```

Window对象的name属性值就是这个窗口的名称。浏览器启动时，初始窗口并没有名称，因此`target`属性也就无法使用它。如果用JavaScript设置了该窗口的name属性，就可以在`target`中使用它了。



3.3.3 Document对象

在客户端JavaScript中，Document对象代表在窗口中显示的HTML文档，它给了客户端JavaScript访问文档内容的能力，所以Document是客户端JavaScript中一个最常用的对象。

1、Document对象概览

1) Document对象与标准

Document对象和它呈现给JavaScript的HTML元素集合构成了文档对象模型（DOM）。W3C标准化了DOM，并发布了该标准的两个版本，分别为1级DOM和2级DOM。

本章介绍的DOM早于W3C标准，但几乎所有的浏览器都支持它，因此它是事实上的工业标准，被称为0级DOM。



3.3.3 Document对象

2) 命名文档中的对象

HTML文档中的每个元素在客户端JavaScript中都是一个对象。每个HTML文档中<form>元素都是Document对象的forms[]数组中的一个（对象）元素。同样，每个HTML文档中元素也是Document对象的images[]数组中的一个元素；每个HTML文档中<a>元素也是Document对象的links[]数组中的一个元素；如此等等。

上述这些HTML元素都可以具有name属性。如果定义了这个属性，则其值将成为Document对象的属性，可以用来引用这个对象元素。

例如，如果在HTML文档中有如下表单：

```
<form name="f1">  
    <input type="button" value="Push Me">  
</form>
```

假定它是文档中的第一个表单元素，则下面的表达式等价：

document.forms[0] //由表单在文档中的位置确定

document.f1 //由名字确定



3.3.3 Document对象

事实上，设置form元素的name属性，也将使name值成为数组forms[]的属性，因此也可以用下属方式引用它：

```
document.forms.f1 //属性
```

```
document.forms["f1"] //关联数组
```

3) 文档对象与事件处理程序

为文档中的对象注册事件处理函数有两种方法：

① 在HTML中，把JavaScript代码串直接赋予一个事件处理程序属性：

```
<form name="myform" onsubmit="return  
    validateform( );">...</form>
```

② 在JavaScript中，将函数（名）赋予一个事件处理程序属性：

```
document.myform.onsubmit = validateform;
```

注意：HTML不区分大小写，因此它的事件处理程序属性可以大写、小写或混写。但JavaScript的所有事件处理程序属性都必须小写。



3.3.3 Document对象

2、文档的动态生成

在HTML文档解析过程（而不是在事件处理程序）中，可以使用write()方法向其输出HTML代码。这是最常用的方法。例如：

```
<script>
```

```
var today = new Date( );
```

```
document.write("<p>Document accessed on: "+today.toString());
```

```
</script>
```



3.3.3 Document对象

3、Document对象的属性

body: 提供对 **<body>** 元素的直接访问。对于定义了框架集的文档，该属性引用最外层的 **<frameset>**。

cookie: 设置或返回与当前文档有关的所有 **cookie**。

domain: 返回当前文档的域名。

lastModified: 文档的最后修改日期

title: 位于文档标记**<title>**和**</title>**之间的文本

URL: 文档的**URL**。它应该与**Window**对象的属性**location.href**相同。

referrer: 将浏览器带到当前文档的前一个文档的**URL**。

4、文档中的表单

HTML中的表单元素由**<form>**标签定义。**Document**对象的**forms[]**数组中存放着对它们的引用。**Web**应用程序的交互界面通常由表单构成，所以它非常重要，将在后面单独介绍。



3.3.3 Document对象

5、图像

HTML中的图像元素由标签定义。Document对象的images[]数组中存放着对它们的引用。

1) 使用src属性进行图像置换

Image对象的src属性既可读又可写。通过改变src属性，浏览器可以装载一个新图像并显示在同一个地方。例如，下例可以实现图像的翻转效果：

```
<a href="help.html"
onmouseover="document.helpimage.src='images/help_rollover.gif';"
onmouseout="document.helpimage.src='images/help.gif';">
  
</a>
```

3.3.3 Document对象

2) 图像的缓存

为了达到图像翻转、动画等的即时相应，需要将用到的图像事先调入浏览器，即对它们进行缓存。要迫使图像被缓存，需要首先使用**Image**构造函数创建一个离屏图像，然后把它的**src**属性设置成想要的**URL**。之后，当需要使用同一个**URL**时，就可以迅速地从浏览器的缓存中装载进来。

上一段代码在执行时，第一次翻转会有延迟，可以进行如下修正：

```
<script>
(new Image(80,20)).src = "images/help_rollover.gif"; //未保存引用
</script>
<a href="help.html"
onmouseover="document.helpimage.src='images/help_rollover.gif';"
onmouseout="document.helpimage.src='images/help.gif';">

</a>
```



3.3.3 Document对象

3) Image对象的事件处理程序属性

- ① **onload**: 图像装载完成后会调用该属性的代码（函数）。
- ② **onerror**: 图像装载发生错误时调用。
- ③ **onabort**: 放弃装载（例如点击了浏览器的停止按钮）时调用。

与上述事件处理程序有关的还有一个**complete**属性。当图像处于装载状态时，其值为**false**；当图像装载完成或终止时，其值为**true**。如果上述三个事件处理程序中的一个被调用了，属性**complete**的值就是**true**。

4) Image对象的其它属性

width、height: 图像的宽度和高度

border: 图像（外加）边界的宽度

hspace、vspace: 图像与水平或垂直边界的距离（空隙）。

lowsrc: 当原始图片下载较慢或下载出错时用来代替的图片URL。



3.3.3 Document对象

6、链接

Document对象的links[]数组中存放着文档中的超文本链接的Link对象。HTML中的带href属性的<a>标记和影像地图map中的<area>标记都可以创建超链接。

Link对象代表超文本链接的URL，它拥有Location对象的所有属性。通过重新设置这些属性，可以对超链接进行重定向。

Link对象支持大量有趣的事件处理程序，如： onclick、onmouseover和onmouseout。

7、锚（书签）

Document对象的数组anchors[]包含了代表HTML中已命名位置的Anchor对象，这些位置由标记<a>和它的name属性标识。

Anchor对象是一种比较简单的对象，它所定义的唯一标准属性是name，即标记<a> 的属性name的值。



3.3.4 表单和表单元素

1、Form对象

JavaScript的Form对象代表了一个HTML表单。它们按照自己在文档中出现的顺序存放在Document对象的forms[]数组中。例如：

```
document.forms[document.forms.length-1]; // 最后一个表单
```

表单对象最重要的属性是elements[]数组，其中包含着各种类型的表单输入元素的JavaScript对象。它们也是按照其在文档中出现的顺序存放的。

还有其它的表单属性，例如：

- **action**: 对应<form>的属性action，提交表单时，向该URL发送表单数据。
- **method**: 对应<form>的属性method，指定表单数据传送至HTTP服务器的方法：**get/post**。**get**: 以URL的形式发送数据；**post**: 浏览器与服务器建立联系后，再将表单数据发往服务器。
- **target**: 对应<form>的属性target，指定目标窗口或框架的名称。

它们都用于控制如何将表单数据提交给服务器以及在哪里显示结果。它们的值都是可读可写的字符串，可以动态地进行设置来改变提交表单的方式。



3.3.4 表单和表单元素

JavaScript的Form对象支持两个方法：**submit()**和**reset()**。调用它们相当于点击了表单的Submit按钮和Reset按钮，用于提交和重置表单。

为了配合**submit()**和**reset()**方法，Form对象提供了事件处理程序**onsubmit**和**onreset**，前者用于探测表单的提交，后者用于探测表单的重置。**onsubmit**在表单提交之前调用，如果它返回**false**，则取消表单提交。这给JavaScript提供了检查用户输入是否存在错误的机会，以免向Web服务器提供不完整或无效的数据。

注意：只用真正点击**Submit**按钮才会触发**onsubmit**事件处理程序，调用表单的**submit()**方法则不会触发它。**reset()**方法也于此类似。

如果表单又长又复杂，就应该使用类似下面的重置确认：

```
<form...
```

```
    onreset="return confirm('Really erase ALL data and start over?')"
```

```
>
```



3.3.4 表单和表单元素

2、表单元素

JavaScript的表单元素对象用于创建图形用户界面。每个Form对象都有elements[]数组，用来存放表示表单元素的对象。每个元素都有type属性，用于区别元素的不同类型，见下页表。

下页表中的大部分元素都是由HTML标签创建的，实际上都是input对象。表中第一列诸如Button、Checkbox这样的名称在客户端JavaScript中可能并不对应真实的对象类型，DOM标准也没有定义使用这些名称的接口。给它们不同的名称是因为每种类型的表单元素都具有不同的外观和行为，把它们看做不同的类型来处理比较方便。



3.3.4 表单和表单元素

对象	HTML标记	type属性	描述和事件
Button	<input type="button"> 或 <button type="button">	"button"	按钮; onclick.
Checkbox	<input type="checkbox">	"checkbox"	多选切换按钮; onclick.
FileUpload	<input type="file">	"file"	带有浏览按钮的文件名输入框; onchange.
Hidden	<input type="hidden">	"hidden"	随表单提交的不可见数据;无事件处理程序
Option	<option>	none	Select对象的一个选项; 事件处理程序属于Select对象.
Password	<input type="password">	"password"	口令输入域, 字符不可见; onchange.
Radio	<input type="radio">	"radio"	单选切换按钮; onclick.
Reset	<input type="reset"> 或 <button type="reset">	"reset"	重置表单按钮; onclick.
Select	<select>	"select-one"	单选列表或下拉菜单(由size属性决定); onchange. (参考Option)
Select	<select multiple>	"select-multiple"	多选列表或下拉菜单(由size属性决定); onchange. (参考Option)
Submit	<input type="submit"> 或 <button type="submit">	"submit"	表单提交按钮; onclick.
Text	<input type="text">	"text"	单行文本输入域; onchange.
Textarea	<textarea>	"textarea"	多行文本输入域; onchange.



3.3.4 表单和表单元素

3、为表单元素书写脚本

(1) 表单和表单元素的命名

每个表单元素都有**name**属性，如果要將表单提交给服务器程序，必须在相应的**HTML**标记中设置这一属性。

表单本身也有一个**name**属性，这个属性与表单的提交没有任何关系。它的存在是为了方便**JavaScript**程序设计者。

这样，就可以使用下面的表达式引用**address**表单中的**zipcode**元素：

document.address.zipcode

为了使表单中的一组**Radio**元素称为单选，它们必须具有相同的**name**属性。对于不需要单选的**Checkbox**按钮，虽然不是必须的，但给它们定义相同的**name**属性也很常见。

当表单中的多个元素具有相同的**name**属性时，**JavaScript**就将这些元素存放在一个数组中，这个数组的名字就是**name**属性的值，数组元素的顺序就是它们在文档中出现的顺序。



3.3.4 表单和表单元素

(2) 表单元素的属性

所有（或绝大多数）表单元素都具有下列性质：

- **type**: 一个只读的字符串，参见前面表格的**type**列。
- **form**: 只读属性，是对包含该元素的**Form**对象的引用。
- **name**: 只读字符串属性，见前页。
- **value**: 可读可写的字符串。与**HTML**表单元素的**name**属性对应。在提交表单时，它将作为**name**值变量的值发送到**Web**服务器。



3.3.4 表单和表单元素

(3) 表单元素的事件处理程序

大多数表单元素都支持如下的事件处理程序：

- **onclick**：在元素上点击鼠标时触发。
- **onchange**：当用户改变了元素表示的值（输入文本且焦点改变，或选择了新的选项）时触发。
- **onfocus**：元素收到输入焦点时触发。
- **onblur**：元素失去输入焦点时触发。

表单元素还支持（几乎）所有HTML元素支持的各种事件处理程序（如 **onmousedown**）。



3.3.4 表单和表单元素

(4) 按钮

Button按钮为用户触发脚本的动作提供了一种清晰可见的方法。它没有自己的默认行为，除非它具有**onclick**（或其它）事件处理程序，否则它在表单中没有任何用处。

需要注意的是，超链接提供了与按钮作用一样的**onclick**事件处理程序，任何按钮对象都可以用一个超链接对象替换，只要该链接在被点击时进行与按钮被点击时一样的操作。

Submit按钮和**Reset**按钮与**Button**按钮类似，只是它们有相关的默认动作，所以即使没有**onclick**事件处理程序，它们也有用。如果**onclick**事件处理程序返回**false**，这两种按钮的默认动作就不会被执行。可以使用**Submit**按钮的**onclick**事件处理程序执行表单验证，但使用**Form**对象自身的**onsubmit**事件处理程序进行表单验证更加常用。

可以用**<button>**标记代替**<input>**标记创建**Button**、**Submit**和**Reset**按钮。**<button>**标记更灵活一些，因为它不显示由**value**属性指定的纯文本，而是显示**<button>**和**</button>**之间的**HTML**内容（格式化的文本或图像），但它们的**type**属性具有相同的值。



3.3.4 表单和表单元素

(5) 切换按钮

Radio元素和**Checkbox**元素都是切换按钮。一组**Radio**元素总是具有相同的**name**属性值，以便具有互斥功能；通常也将**Checkbox**元素进行分组，每组元素有相同的**name**属性值。这些同名元素被放在一个元素数组中，数组名即是**name**属性的值。

Radio元素和**Checkbox**元素都定义了布尔值属性**checked**。属性**defaultChecked**是只读的布尔值，指示装载页面时该元素是否被选中。

Radio元素和**Checkbox**元素自身不显示任何文本，而是显示相邻的HTML文本（可与一个<label>标记配合使用）。这意味着设置**Radio**元素和**Checkbox**元素的**value**属性不会改变它们的外观（像<button>元素一样），只能改变提交表单时发送给Web服务器的字符串。

用户点击切换按钮时，**Radio**元素或**Checkbox**元素将触发它的**onclick**事件处理程序，新的浏览器还会触发**onchange**事件处理程序。



3.3.4 表单和表单元素

(6) 文本框

Text元素允许用户输入单行文本。它的可读写属性值**value**即表示其中的文本。当用户输入新文本或编辑已有文本，然后把焦点移开时，将触发**onchange**事件处理程序。

Textarea元素和**Text**元素相似，只是它允许输入多行文本。**Textarea**元素由**<textarea>**标记创建，而不是创建**Text**元素的**<input>**标记。可以像使用**Text**元素的**value**属性和**onchange**事件处理程序一样使用**Textarea**元素的这些属性。但其实两种元素的类之间没有继承关系。

Password元素是修改过的**Text**元素，在用户输入时它显示星号。注意：**Password**元素可以防止用户输入的密码被窥视，但当表单被提交时，其值并没有被加密（除非通过安全的**HTTPS**链接提交它），所以当它通过网络被提交时，便可能被看到。

File元素允许用户输入要上传到服务器的文件名。它实质上是一个与内建按钮组合在一起的**Text**元素，这个内建按钮可以弹出一个文件选择对话框。像**Text**元素一样，**File**元素也有**onchange**事件处理程序。但与**Text**元素不同的是，**File**元素的**value**值是只读的。



3.3.4 表单和表单元素

W3C还没有定义标准的键盘事件及其处理程序。然而，目前的浏览器都定义了onkeypress, onkeydown和onkeyup 事件处理程序。

任何Document对象都可以设置这些事件处理程序，但真正与键盘输入有关的元素（例如Text）更倾向于使用这些事件。

从onkeypress或onkeydown事件处理程序返回false，可以防止用户的键盘输入被记录下来。例如：假如你想强迫用户只输入数字字符时，这是非常有用的。



3.3.4 表单和表单元素

(7) Select元素和Option元素

Select元素表示用户可以选择的选项集合（由**Option**元素表示）。浏览器用下拉菜单或列表框表示**Select**元素（由**size**属性决定）。**Select**元素可以用两种完全不同的方式操作，其**type**属性值由其设置决定。如果**<select>**标记有**multiple**属性，就允许用户进行多选，其**Select**对象的**type**属性为“**select-multiple**”。否则，如果没有**multiple**属性，那么用户只能进行单元，其**Select**对象的**type**属性为“**select-one**”。

Select元素没有**value**属性，但**Option**元素有。

当用户选中或取消一个选项时，**Select**元素将触发它的**onchange**事件处理程序（**Option**元素没有事件处理程序）。对于“单选”型的**Select**元素，可读可写的属性**selectedIndex**用数字指示了当前被选中的选项。对于“多选”型的**Select**元素，要确定选中了哪些选项，必须遍历**options[]**数组中的所有元素，检查每个**Option**对象的**selected**属性的值。

除了**selected**属性外，**Option**元素还有**text**属性，用于读取或设置元素所呈现的文本。而**value**属性也是可读可写的字符串，通常表示的是表单提交时发送给**Web**服务器的值，也可以用来存储一个与**text**不同的值。



3.3.4 表单和表单元素

除了设置Option对象的text属性外，还有其它方法可以动态改变Select元素中显示的选项。可以通过把options.length设置为想要的数目来截取Option元素的数组（把options.length设置为0可以删除所有Option对象）；把options[]数组中的某个元素设置为null，就可以删除一个Option对象（其后的对象自动前移）。最后，利用构造函数Option()，可以动态创建新的Option元素，并把它附加在options[]数组的末尾给Select元素增加新的选项。例如：

```
var zaire = new Option("Zaire", // text 属性
                        "zaire", // value 属性
                        false, // defaultSelected 属性
                        false); // selected 属性
```

```
var countries = document.address.country; // 获取Select对象
countries.options[countries.options.length] = zaire;
```

可以用<optgroup>标记对Option元素进行分组(缩格)。<optgroup>标记具有label属性，是其呈现的文本。<optgroup>标记可见但不可选，也不会出现在Select元素的options[]数组中。



3.3.4 表单和表单元素

(8) Hidden元素

Hidden元素在表单中不可见。它的作用是在提交表单时把任意的文本发送给服务器。由于**Hidden**元素没有可视化的外观，所以它不能生成事件，没有事件处理程序。**value**属性允许你读写与**Hidden**元素相关的文本，它将被传送到服务器端处理程序。



3.3.5 文档对象模型

文档对象模型（Document Object Model, DOM）是表示文档、访问和操作构成文档的各种元素的应用程序接口（API）。

本章讨论W3C DOM，它是由World Wide Web委员会定义的标准文档对象模型，简称DOM。DOM标准是传统Web浏览器DOM（0级DOM）的全特性超集。

3.3.6 文档对象模型

1、DOM概述

(1) 把文档表示为树

HTML文档有一个由嵌套标记构成的层次结构，这个结构在DOM中被表示为一棵对象树。这个HTML文档的树形表示由节点构成，主要包含表示元素（即标记）的节点和表示文本串的节点。也可以包含表示HTML注释的节点。例如：

<!--简单的HTML文档和其对应的DOM树形表示-->

<html>

<head>

<title>Sample Document</title>

</head>

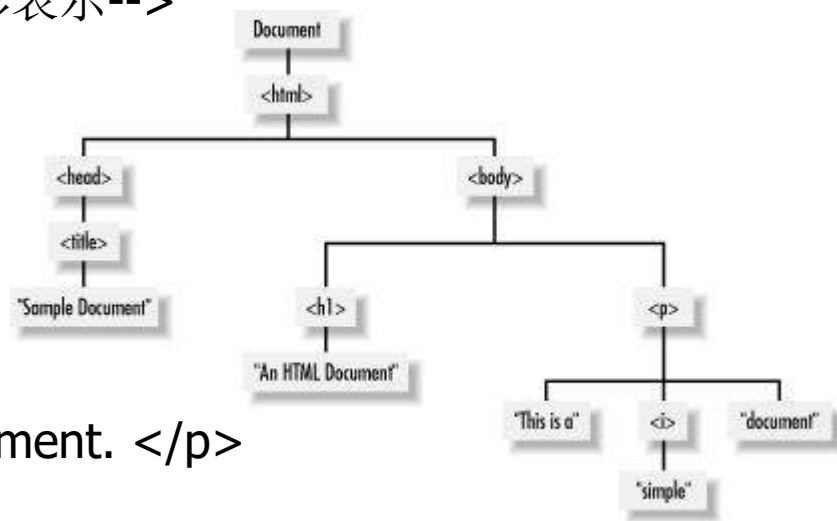
<body>

<h1>An HTML Document</h1>

<p>This is a <i>simple</i> document. </p>

</body>

</html>





3.3.6 文档对象模型

(2) 树中的节点

DOM树形结构由各种类型的节点对象构成。所有类型的节点对象都具有：`childNodes`属性，其中包含所有的子节点；`firstChild`、`lastChild`、`nextSibling`、`previousSibling`和`parentNode`属性，提供了遍历树的途径；`appendChild()`、`removeChild()`、`replaceChild()`和`insertBefore()`方法使你能够为文档树添加节点或从文档树中删除节点。

`Document`节点类型定义了W3C标准化之前（0级DOM）的浏览器支持的各种`Document`对象的属性和方法。例如：`location`属性、`forms[]`数组和`write()`方法的。

`Element`节点类型定义了`id`、`style`、`title`、`lang`、`dir`和`className`属性。用这些属性访问HTML标记的`id`、`style`、`title`、`lang`、`dir`和`class`属性非常方便，这些属性是所有HTML标记共有的。

各种HTML元素类型（它们属于`Element`的子类）都定义了标记专用的属性。你可以安全地假定：表示特定HTML标记的节点类型具有该标记的所有标准属性所对应的属性。

`Element`节点的属性可以由`Element`类型的方法`getAttribute()`、`setAttribute()`和`removeAttribute()` 进行查询、设置和删除。



3.3.6 文档对象模型

(3) 节点属性的命名规则

属性名应该以小写字母开头。如果属性名由多个单词构成，第二个单词开始的每个单词的首字母都要大写。因此，`<input>`标记的`maxlength`属性将命名为`maxLength`。

当HTML属性名与JavaScript关键字发生冲突时，应该在属性名前加前缀“`html`”。因此，`<label>`标记的`for`属性被命名`htmlFor`。这个规则的一个例外是`class`性质，它被命名为`className`属性。

记住：JavaScript的所有事件处理程序属性都必须小写。



3.3.6 文档对象模型

2、使用DOM

(1) 遍历文档

① 使用childNodes属性:

```
<head> <!-- 3-4.html -->
```

```
<script>
```

```
function countTags(n) { // n是一个Node
```

```
    var numtags = 0
```

```
    if (n.nodeType == 1 /*Node.ELEMENT_NODE*/) numtags++;
```

```
    var children = n.childNodes;
```

```
    for(var i=0; i<children.length; i++) numtags+=countTags(children[i]);
```

```
    return numtags;}
```

```
</script>
```

```
</head>
```

```
<body onload="alert('It has ' + countTags(document) + ' tags')">
```

```
    This is a <i>sample</i> document.
```

```
</body>
```



3.3.6 文档对象模型

② 使用firstChild和nextSibling属性:

```
<head> <!-- 3-5.html -->
```

```
<script>
```

```
function countCharacters(n) { // n是一个Node
    if (n.nodeType == 3 /*Node.TEXT_NODE*/) return n.length;
    var numchars = 0;
    for(var m = n.firstChild; m != null; m = m.nextSibling) {
        numchars += countCharacters(m);
    }
    return numchars;
}
```

```
</script>
```

```
</head>
```

```
<body onload="alert('Length: ' + countCharacters(document.body))">
```

```
    This is a sample document.How long is it?
```

```
</body>
```




3.3.6 文档对象模型

(2) 确定文档中的特定元素

① Document对象有几个获取元素对象（元素）的方法：

➤ `getElementsByTagName()`：返回元素的数组。例如：

```
var body = document.getElementsByTagName("body")[0];
```

➤ `getElementsByName()`：返回元素的数组。例如：

```
// 找出所有<input type="radio" name="shippingMethod">单选按钮  
var choices = document.getElementsByName("shippingMethod");
```

➤ `getElementById()`：返回具有特定标识符元素。例如：

```
//<p id="specialParagraph">  
var myParagraph = document.getElementById("specialParagraph");
```

② Element对象也定义了`getElementsByTagName()`方法，它返回此对象的子元素数组。例如：

```
// 找到文档中特定的Table元素，统计它的行数  
var tableOfContents = document.getElementById("TOC");  
var rows = tableOfContents.getElementsByTagName("tr");  
var numRows = rows.length;
```



3.3.6 文档对象模型

(3) 修改文档

① 可以用插入、删除、重排节点和重定父节点的方式修改文档，例如：

```
<script> <!-- 3-6.html -->
function embolden(node) {
    var bold = document.createElement('b');
    var parent = node.parentNode;
    parent.replaceChild(bold, node);
    bold.appendChild(node);
}
</script>
<p id="p1">This <i>is</i> paragraph #1.</p>
<p id="p2">This <i>is</i> paragraph #2.</p>
<button onclick="embolden(document.getElementById('p1'));">
    Embolden
</button>
```



3.3.6 文档对象模型

② 也通过设置文档元素的性质改变文档的显示。例如：

```
var headline = document.getElementById("headline");  
headline.setAttribute("align", "center");
```

表示HTML元素的DOM元素，为HTML元素的每个标准属性定义了JavaScript属性，所以下面的代码可以得到相同的效果：

```
var headline = document.getElementById("headline");  
headline.align = "center"; // Set alignment attribute.
```

注意：IE对setAttribute()的支持有限。



3.3.6 文档对象模型

(4) 给文档添加内容

`Document.createElement()`和`Document.createTextNode()`方法，可以创建新的元素和文本节点。`Node.appendChild()`、`Node.insertBefore()`和`Node.replaceChild()`方法可以将它们加入文档中。使用这些方法，你可以建立包含任意文档内容的DOM文档树。例如：

```
function debug(msg) { //msg为字符串
    if (!debug.box) { //第一次调用创建
        debug.box = document.createElement("div");
        document.body.appendChild(debug.box); //body是Document的属性
        debug.box.innerHTML =
            "<h1 style='text-align:center'>Debugging Output</h1>";
    }
    var p = document.createElement("p");
    p.appendChild(document.createTextNode(msg));
    debug.box.appendChild(p);
}
```



3.3.6 文档对象模型

上例中的`debug()`函数可以将调试信息附加在文档的后面。可用于下面的HTML文档：

```
<!-- 3-7.html -->
<body>
  <script>var numtimes=0;</script>
  <button onclick="debug('clicked: ' + numtimes++);">
    press me
  </button>
</body>
```

这个例子中示范了一个给文档添加内容的新方法。即：使用了`<div>`元素的`innerHTML`属性。把任何元素的`innerHTML`属性设置为一个HTML文本串，都会使那段HTML被解析为元素的内容。虽然这个属性不属于DOM API，但它非常简便，而且所有的浏览器都支持它。



3.3.7 层叠样式单和动态HTML

层叠样式单（Cascading Style Sheets, CSS）指定了HTML文档的视觉表现。使用CSS和JavaScript，可以创造出各种视觉效果，这些视觉效果统称为动态HTML（DHTML）。

1、使用脚本控制样式

文档对象模型允许将应用到每个文档元素上的样式进行脚本化，利用脚本化CSS样式的能力，可以动态地改变颜色、字体等。更重要的是，可以用它设置和改变元素的位置，甚至隐藏或显示元素。这意味着可以用DHTML技术创造出动画效果。



3.3.7 层叠样式单和动态HTML

一旦获取了想应用样式的元素的引用，就可以使用元素的**style**属性获取那个元素的**CSSProperties**对象。这个**JavaScript**对象拥有与**CSS**的每个样式对应的**JavaScript**属性。设置这些属性与设置**HTML**元素的**style**属性中的相应样式的效果一样。读取这些属性将返回**HTML**元素的**style**性质设置的**CSS**属性（如果存在）。即设置和读取的都是**HTML**元素的内联样式，而不是应用到该元素的样式表的样式信息。设置对象的这个属性后，就定义了元素的内联样式，它能有效地覆盖样式表样式。例如：

```
var imgs = document.getElementsByTagName("img");
for(var i = 0; i < imgs.length; i++) {
    var img=imgs[i];
    if (img.width == 468 && img.height == 60) //标题广告尺寸
        img.style.visibility = "hidden"; //占位但不可见
}
```



3.3.7 层叠样式单和动态HTML

(1) CSS属性在JavaScript中的命名惯例

许多CSS样式名中都包含连字符，例如：`font-faminy`。在JavaScript中连字符被解释为减号，因此，`CSSProperties`对象的属性名和真正的CSS性质名有点不同：

- 如果一个CSS属性名包含一个或多个连字符，那么`CSS2Properties`属性名删除了连字符，且原来紧接在连字符后的字母改为大写。例如，可以用下列代码访问`font-family`属性：

```
element.style.fontFamily = "sans-serif";
```

- `float`是JavaScript的保留关键字。在JavaScript中，CSS的`float`样式名对应`CSS2Properties`的`cssFloat`属性。



3.3.7 层叠样式单和动态HTML

(2) 使用样式属性

在使用**CSSProperties**对象的属性时，要记住，所有值必须是字符串，而且必须是有单位的。不论是设置还是读取。

```
e.style.marginLeft = "30px";
```

```
var totalMarginWidth = parseInt(e.style.marginLeft) +  
                        parseInt(e.style.marginRight); //相同单位时
```

有些**CSS**属性是其它属性的简化形式，**CSSProperties**对象也有相应与这些简化属性的属性。例如：

```
e.style.margin = topMargin + "px " + rightMargin + "px " +  
                 bottomMargin + "px " + leftMargin + "px";
```

但使用简化属性通常并不方便。

再次强调：**CSSProperties**对象只能设置单个元素的内联属性，也只能读取单个元素的内联属性（包括由**CSSProperties**对象设置的）。它没有提供查询**CSS**级联样式的方法，也没有提供判断应用到给定元素的完整样式集合的方法。



3.3.7 层叠样式单和动态HTML

(3) DHTML动画

用JavaScript和CSS能够实现的最强大的DHTML功能是动画。DHTML动画没有什么特殊之处，所需要做的只是周期性地改变元素的一个或多个样式属性。例如：

```
<div id="urgent">
    <h1>Red Alert!</h1>The Web server is under attack!
</div>
<script>
var e = document.getElementById("urgent");
var colors = ["white", "yellow", "orange", "red"]; //特殊色
var nextColor = 0;
setInterval("e.style.backgroundColor=colors[nextColor++ % colors.length];", 500);
</script>
```

所有DHTML动画都要是用window对象的setInterval()或setTimeout()方法。



3.3.7 层叠样式单和动态HTML

(4) DHTML动画举例

由于代码较长，请参看：3-8.html。

此例演示了下列功能：

- ① 具有动画功能的HTML标签<marquee>。它不属于W3C标准，可酌情使用。
- ② 使用css进行剪切
- ③ 使用javascript和css实现动画效果。
- ④ 使用javascript的事件处理实现动画控制。



3.3.7 层叠样式单和动态HTML

2、关于样式和样式表的其它DOM API

对于想创建DHTML的脚本作者，使用CSSProperties对象的属性控制HTML元素的内联样式通常就足够了，但这毕竟只是CSS的DOM API的一部分。其它DOM API则提供了诸如确定元素的完整样式集合、处理或创建CSS样式表单，甚至覆盖内联样式的接口。使用时可以查询相关文档。



第四章 jQuery简介

4.1 jQuery能做什么

4.1.1 概述

JavaScript功能强大，可以实现具有交互功能的动态页面。但直接使用JavaScript过于繁琐，特别是当面对不同的浏览器或不同版本的浏览器时更是如此。

jQuery是一个优秀的JavaScript库，也是当前最常用的JavaScript库。它使我们能够站在巨人的肩膀上实现常见任务的自动化和复杂任务的简单化。

它的大部分概念都是从HTML和CSS中借用来的，因此对HTML和CSS相对熟悉但编程经验并不丰富的设计人员可以快速地掌握它。



4.1.2 jQuery的优点

1. 功能强大，易扩展，代码量小，免费使用
2. 充分利用CSS的概念和语法
使得熟悉CSS的设计人员有顺畅的切入点。
3. 屏蔽浏览器的不一致性
4. 总是面向集合
查询结果和操作对象都是集合。这样就可以通过所谓的隐式迭代避免臃肿的循环结构，大幅减少代码。
5. 将多重操作集于一行
这被称为连缀（chaining）：对一个jQuery对象的操作结果还是一个jQuery对象。这样可以避免使用临时变量和重复代码。
所有这些得益于jQuery优秀的设计理念。



4.1.3 第一个jQuery驱动的面

1. 下载jQuery到本地

2. 在HTML中包含jQuery

...

```
<head>
```

```
  <meta charset="utf-8">
```

```
  <title>Through the Looking-Glass</title>
```

```
  <link rel="stylesheet" href="4-1-1.css" type="text/css" />
```

```
  <script src="jquery.js"></script>
```

```
  <script src="4-1-1.js"></script>
```

```
</head>
```

...

详细代码见例4-1-1.html



4.1.3 第一个jQuery驱动的面

3.编写jQuery代码(4-1-1.js)

```
$(document).ready(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```

说明:

- ① 在jQuery中，总是使用`$(document).ready()`方法。通常，它比`onload`事件处理程序更有优势，不必等待页面所有内容加载完毕，只要DOM已经就绪。
- ② 尽量使用匿名函数（即函数常量或称lambda函数）。

4.查看效果



4.1.3 第一个jQuery驱动页面

5.与纯JavaScript比较

```
window.onload = function() {  
    var divs = document.getElementsByTagName('div');  
    for (var i = 0; i < divs.length; i++) {  
        if (hasClass(divs[i], 'poem-stanza')  
            && !hasClass(divs[i], 'highlight')) {  
            divs[i].className += ' highlight';  
        }  
    }  
}  
  
function hasClass( elem, cls ) {  
    var reClass = new RegExp(' ' + cls + ' ');  
    return reClass.test(' ' + elem.className + ' ');  
}  
};
```



4.1.3 第一个jQuery驱动的面

6. 开发工具

推荐适用**firefox**浏览器及其**firebug**工具



4.2 选择元素

这里所说的元素即DOM元素。jQuery不仅利用了CSS选择元素的方式，而且还自定义了它独有的选择符。

4.2.1 DOM

文档对象模型是JavaScript和网页之间的接口，它以对象网络(树)而非文本的形式来表现HTML的源代码。例如：

DOM一例

<html>

<head>

<title>the title</title>

</head>

<body>

<div>

<p>This is a paragraph.</p>

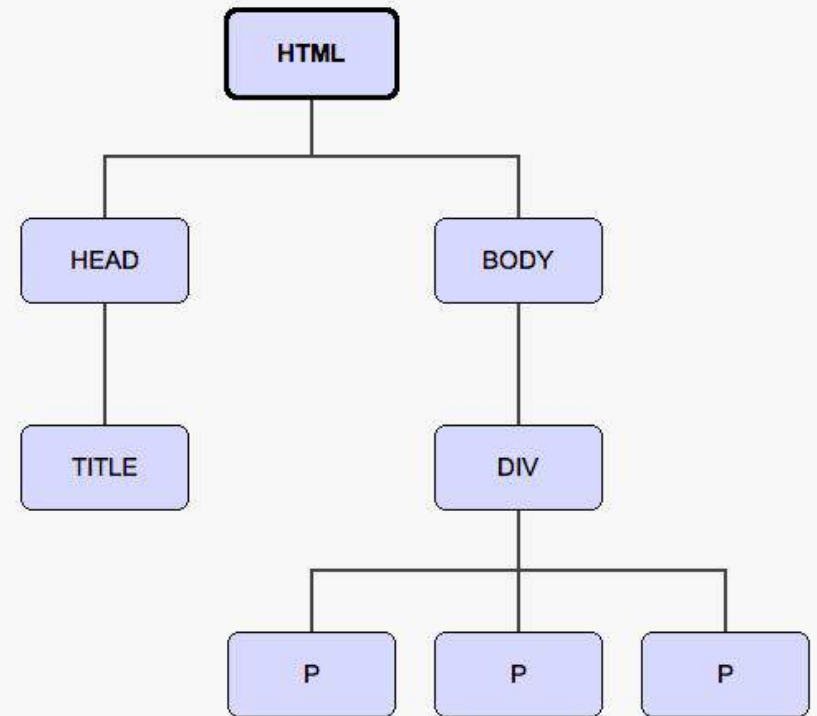
<p>This is another paragraph.</p>

<p>This is yet another paragraph.</p>

</div>

</body>

</html>





4.2.2 选择函数\$()

在jQuery中，所有选择由\$()函数完成。它的参数通常是CSS选择符，返回结果是一个jQuery对象（一个集合对象）。

需要说明的是，\$是JavaScript的合法标识符，而且很多类似jQuery的JavaScript库都使用了这个标识符。当同时使用多个JavaScript函数库时，是会引起冲突的。

其实在jQuery中，\$标识符不过就是jQuery的“别名”。在引起冲突的情况下，需要在自定义的jQuery文件中，将\$标识符全部替换为jQuery。



4.2.3 CSS选择符

jQuery支持几乎所有CSS选择符。有三种基本的CSS选择符：标签名、ID和类。它们可以单独使用，也可以与其它选择符组合使用。

基本的选择符

选择符	CSS	jQuery	说明
标签名	p{}	\$('p')	取得文档中的所有段落
ID	#some-id{}	\$('#some-id')	取得文档中ID为some-id的一个元素
类	.some-class{}	\$('.some-class')	取得文档中类为some-class的所有元素

jQuery支持从CSS1到CSS3的几乎所有选择符，只要启用JavaScript，你不必关心浏览器支持什么版本。

通过在jQuery中使用CSS选择符，4-2-1.html利用一个无序列表实现了一个常用的导航菜单。



4.2.3 CSS选择符

4-2-1.html: 一个无序列表

Selected Shakespeare Plays

- Comedies
 - [As You Like It](#)
 - All's Well That Ends Well
 - A Midsummer Night's Dream
 - Twelfth Night
- Tragedies
 - [Hamlet](#)
 - Macbeth
 - Romeo and Juliet
- Histories
 - Henry IV ([email](#))
 - Part I
 - Part II
 - [Henry V](#)
 - Richard II



4.2.3 CSS选择符

1. 基于列表项的级别添加样式

- 1) 使顶级项水平排列。
- 2) 为非顶级项添加样式

```
/* 4-2-1.css */
.horizontal {
    float: left;
    list-style: none;
    margin: 10px;
}
.sub-level {
    background: #ccc;
}
/* 4-2-1.js */
$(document).ready(function() {
    $('#selected-plays > li').addClass('horizontal');
    $('#selected-plays li:not(.horizontal)').addClass('sub-level');
});
```

注：这里使用了新的否定式伪类选择符：**not(.horizontal)**



4.2.3 CSS选择符

2. 属性选择符

属性选择符是**CSS**选择符中特别有用的一类选择符，**HTML**标记的属性放在方括号[]中，用于选择具有该属性的标记（元素）。

在属性选择符中可以使用通配符：

^：值在字符串的开始

\$：值在字符串的结尾

*：值可以在字符串的任意位置

!：对值取反

下面将为4-2-1.html中的链接添加样式。



4.2.3 CSS选择符

```
/* 4-2-1.css */
a { color: #00c;}
a.mailto {
    background: url(email.png) no-repeat 100% 2px;
    padding-right: 18px;}
a.pdflink {
    background: url(pdf.png) no-repeat 100% 0;
    padding-right: 18px;}
a.henrylink {
    background-color: #fff;
    padding: 2px;
    border: 1px solid #000;}
/* 4-2-1.js */
$(document).ready(function() {
    $('a[href^="mailto:"]').addClass('mailto');
    $('a[href$=".pdf"]').addClass('pdflink');
    $('a[href^="http"][href*="henry"]').addClass('henrylink');
});
```

注：这里使用了组合的属性选择符，它们是‘与’的关系。



4.2.4 jQuery自定义选择符

jQuery自定义选择符的语法与CSS中的伪类选择符语法相同，即选择符以冒号（:）开头。它通常跟在一个CSS选择符后面，基于已经选择的元素集的位置来查找元素。例如：

`$('#div.horizontal:eq(1)')`

用来确定原选择集中的第二个元素（JavaScript中的编号从0开始）。

需要注意的是，由于CSS中的编号则是从1开始的，所以：

`$('#div:nth-child(1)')`

获取的是作为其父元素的第一个子元素的所有div元素，也可以写为：

`$('#div.first-child')`。

下面介绍jQuery中的三个十分有用的自定义选择符:odd、:even和contains()。



4.2.4 jQuery自定义选择符

4-2-2.html: 两个表格

Shakespeare's Plays

As You Like It	Comedy	
All's Well that Ends Well	Comedy	1601
Hamlet	Tragedy	1604
Macbeth	Tragedy	1606
Romeo and Juliet	Tragedy	1595
Henry IV, Part I	History	1596
Henry V	History	1599

Shakespeare's Sonnets

The Fair Youth	1-126
The Dark Lady	127-152
The Rival Poet	78-86



4.2.4 jQuery自定义选择符

```
/* 4-2-2.css */  
.alt {  
    background-color: #ccc;  
}  
/* 4-2-2.js */  
$(document).ready(function() {  
    //$('tr:even').addClass('alt');  
    $('tr:nth-child(odd)').addClass('alt');  
    $('td:contains(Henry)').addClass('highlight');  
});
```

注意：

- ① 这里使用的:**nth-child()**是**jQuery**中唯一从**1**开始计数的选择符。它的参数可以是数值、**odd**或**even**。
- ② **contains()**选择符区分大小写。



4.2.5 jQuery中用于遍历DOM的方法

1. filter()方法的作用

该方法可以完成简单选择符表达式的功能，例如：

```
$('tr').filter(':even').addClass('alt');
```

可以用来代替：`$('#tr:even').addClass('alt');`

而且由于该方法可以接受函数作参数，执行复杂的测试，因此功能十分强大。

下面的代码为所有外部链接添加类名：**external**。

```
/* 添加在4-2-1.css中 */
```

```
a.external {  
    background: #fff url(external.png) no-repeat 100% 2px;  
    padding-right: 16px;  
}
```

```
/* 添加在4-2-1.js中 */
```

```
$('#a').filter(function() {  
    return this.hostname && this.hostname != location.hostname;  
}).addClass('external');
```

注：更准确地说，**.filter()**方法会迭代所有匹配的元素，对每个元素(**this**)都调用的函数并测试返回值，如果函数返回**false**，则从匹配集合中删除相应元素；如果返回**true**，则保留相应元素。



4.2.5 jQuery中用于遍历DOM的方法

2. 主要遍历方法

```
$('td:contains(Henry)').next().addClass('highlight'); //next()
$('td:contains(Henry)').nextAll().addClass('highlight'); //nextAll()
$('td:contains(Henry)').nextAll().andSelf().addClass('highlight'); //andSelf()
$('td:contains(Henry)').parent().children().addClass('highlight');//parent()  
和children()
```

说明：连缀可以写在多行上以增加可读性。例如：

```
$('td:contains(Henry)) // Find every cell containing "Henry"
.parent() // Select its parent
.find('td:eq(1)') // Find the 2nd descendant cell
.addClass('highlight') // Add the "highlight" class
.end() // Return to the parent of the cell containing "Henry"
.find('td:eq(2)') // Find the 3rd descendant cell
.addClass('highlight'); // Add the "highlight" class
```



4.2.5 jQuery中用于遍历DOM的方法

3. 获取元素

所有选择符表达式和大多数jQuery方法都返回一个jQuery对象，它是一个集合对象，其中包含代表HTML标记的DOM元素。如果需要访问其中的元素，可使用jQuery提供的.get()方法。这样就可以直接访问DOM元素，例如标签名：

```
var myTag = $('#my-element').get(0).tagName
```

或简写为：

```
var myTag = $('#my-element')[0].tagName;
```




4.3 事件

jQuery增强并扩展了基本的JavaScript事件处理机制，它不仅提供了更加优雅的事件处理语法，而且也极大地增强了事件处理机制。

4.3.1 `$(document).ready()`说明

1. 对页面加载的响应

`$(document).ready()`与`window.onload`在对页面加载的响应时机上存在着微妙的差异。一般来说，`$(document).ready()`要优于使用`onload`事件处理程序，但由于支持文件可能还没有加载完毕，所以类似图像的宽度和高度这样的属性此时不一定会有效。如果需要访问这些属性，可能就得选择实现一个`onload`事件处理程序。这两种机制是可以和平共处的。

2. 反复调用的效果

每次调用`$(document).ready()`都会向内部的行为队列中添加一个新函数，当页面加载完毕后，所有函数都会被执行，而且是按照它们的添加顺序执行。这点比常规JavaScript事件处理机制灵活。



4.3.1 \$(document).ready()说明

3. \$(document).ready()的简写方式

```
$(document).ready(function() {  
    // Our code here...  
});
```

可以被简写为:

```
$(function() {  
    // Our code here...  
});
```

就是说: 当为\$()传递一个函数作为参数时, jQuery会隐式调用.ready()。但不提倡使用这种方法。

4. 在有库冲突时使用\$()

在某些情况下, 可能有必要在同一个页面中使用多个JavaScript库。由于很多库都使用了\$标示符, 这将引起冲突。

为此, jQuery提供了一个jQuery.noConflict()方法, 调用该方法可以把对\$标示符的控制权转让给其它库。例如:



4.3.1 \$(document).ready()说明

```
<script src="prototype.js"></script>
<script src="jquery.js"></script>
<script> jQuery.noConflict(); </script>
<script src="myscript.js"></script>
```

上面的带代码将对\$的控制权交给了prototype.js库。这时，在自己的脚本(myscript.js)中，可以使用两个库。当然，在使用jQuery方法时，需要用jQuery代替\$。

在这种情况下，仍然有一种在.ready()方法中使用\$的技巧。在传递给ready的回调函数中可以接收一个参数——jQuery对象本身。利用这个参数，可以重新命名jQuery为\$，而不必担心冲突：

```
jQuery(document).ready(function($) {
    // 在这里，可以正常使用$
```

```
});
```

或者简写为：

```
jQuery(function($) {
    // Code that uses $.
});
```

4.3.2 简单事件

jQuery通过.bind()方法为DOM元素添加事件响应函数。

1. 一个简单的样式转换器

例：4-3-1.html

Style Switcher

Default

Narrow Column

Large Print

A Christmas Carol

In Prose, Being a Ghost Story of Christmas

by Charles Dickens

Preface

I HAVE endeavoured in this Ghostly little book, to raise the Ghost of an Idea, which shall not put my readers out of humour with themselves, with each other, with the season, or with me. May it haunt their houses pleasantly, and no one wish to lay it.

Their faithful Friend and Servant,

C. D.

December, 1843.

Stave I: Marley's Ghost

MARLEY was dead: to begin with. There is no doubt whatever about that. The register of his burial



4.3.2 简单的事件

该例的按钮部分有如下代码：

```
/* 在4-3-1.html中 */
```

```
<div id="switcher" class="switcher">  
  <h3>Style Switcher</h3>  
  <button id="switcher-default">Default</button>  
  <button id="switcher-narrow">Narrow Column</button>  
  <button id="switcher-large">Large Print</button>  
</div>
```

```
/* 在4-3-1.css中 */
```

```
body.large .chapter { font-size: 1.5em;}  
body.narrow .chapter { width: 250px;}
```



4.3.2 简单的事件

```
/* 4-3-1.js */
$(document).ready(function() {
    $('#switcher-default').bind('click', function() {
        $('body').removeClass('narrow');
        $('body').removeClass('large');
    });
    $('#switcher-narrow').bind('click', function() {
        $('body').addClass('narrow');
        $('body').removeClass('large');
    });
    $('#switcher-large').bind('click', function() {
        $('body').removeClass('narrow');
        $('body').addClass('large');
    });
});
```



4.3.2 简单事件

2. 使用this关键字

在事件处理程序中，**this**关键字表示引发事件的DOM元素本身。由于`$()`函数可以将DOM元素作为参数，通过在事件处理程序中使用`$(this)`，为相应的元素创建jQuery对象，就可以用jQuery的方式处理它。

下面使用**this**关键字为4-3-1.html增加当前按钮提示功能。

```
/* 在4-3-2.css中 */  
.selected {  
    font-weight: bold;  
}
```



4.3.2 简单事件

```
/* 4-3-2.js */
$(document).ready(function() {
    $('#switcher-default').addClass('selected');
    $('#switcher button').bind('click', function() {
        $('body').removeClass();
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
    $('#switcher-narrow').bind('click', function() {
        $('body').addClass('narrow');
    });
    $('#switcher-large').bind('click', function() {
        $('body').addClass('large');
    });
});
```




4.3.2 简单事件

2. 事件的简写

由于为某个事件绑定事件处理函数极为常见，jQuery为此提供了简写方式。可以不写出`.bind()`，而是以事件名作为方法名，直接将事件处理程序绑定到同名事件上。

```
/* 4-3-3.js */
$(document).ready(function() {
    $('#switcher-default').addClass('selected');
    $('#switcher button'). click(function() {
        var bodyClass = this.id.split('-')[1];
        $('body').removeClass().addClass(bodyClass);
        $('#switcher button').removeClass('selected');
        $(this).addClass('selected');
    });
});
```



4.3.2 简单事件

jQuery为标准的DOM事件都提供了相应的事件简写方法:

- change
- click
- dblclick
- error
- focus
- keydown
- keypress
- keyup
- load
- mousedown
- mousemove
- mouseout
- mouseover
- mouseup
- resize
- scroll
- select
- submit
- unload

4. jQuery事件方法的来历

jQuery中的大多数事件方法都直接对应标准的DOM事件，也有少数出于跨浏览器优化和方便性考虑而添加的自定义处理方法，例如`.ready()`方法。另外还有经过标准化的IE同名事件的处理方法，例如：`.mouseenter()`，`.mouseleave()`，`.focusin()`，和`.focusout()`。



4.3.3 复合事件

jQuery自定义了两个事件处理方法`.toggle()`和`.hover()`。由于它们截取组合的用户操作，并且以多个函数作为响应，因此被称为复合事件处理程序。

1. `.toggle()`

`.toggle()`方法接受两个参数，这两个参数都是函数。第一次在元素上单击会调用第一个函数，第二次单击会调用第二个函数。这一过程反复循环。下面通过该方法实现4-3-4.html中样式转换器的折叠。

```
/* 在4-3-4.css中 */  
.hidden {  
    display: none;  
}
```



4.3.3 复合事件

```
/* 在4-3-4.js中 */  
$(document).ready(function() {  
    $('#switcher h3').toggle(  
        function() {$('#switcher button').addClass('hidden');},  
        function() {$('#switcher button').removeClass('hidden');}  
    );  
});
```

在本例这种特殊的情况下，可以使用.toggleClass()方法。例如：

```
$(document).ready(function() {  
    $('#switcher h3').click(function() {  
        $('#switcher button').toggleClass('hidden');  
    });  
});
```

该方法在自动检查该类是否存在，从而决定是为元素应用还是移除该类名。



4.3.3 复合事件

2. .hover()

虽然CSS具有: hover伪类选择符，但它通常只用在链接元素上。jQuery的.hover()方法同.toggle()方法一样，也接受两个函数参数。第一个函数会在鼠标进入被选择元素时执行，而第二个函数会在鼠标离开该元素时执行。通过使用该方法，可以实现4-3-5.html中样式转换器标题的反转效果。

```
/* 在4-3-5.css中 */
```

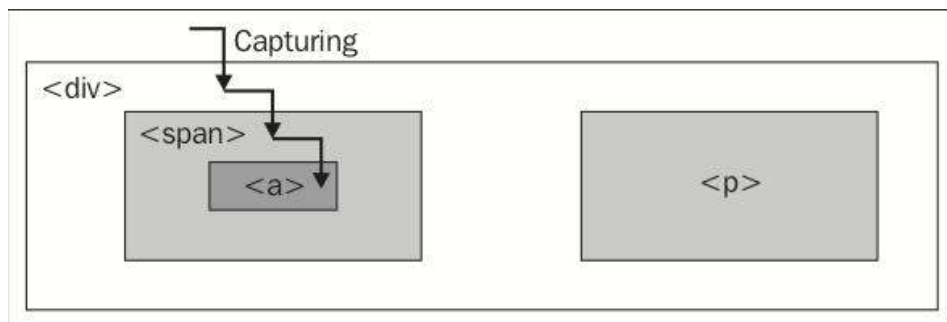
```
.hover {  
    cursor: pointer;  
    background-color: #afa; }
```

```
/* 在4-3-5.js中 */
```

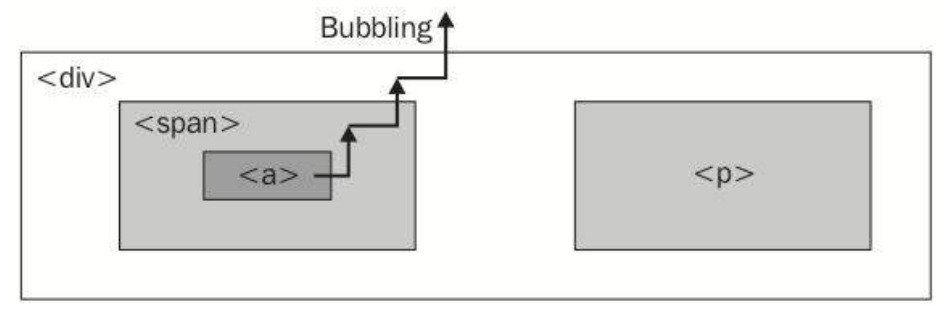
```
$(document).ready(function() {  
    $('#switcher h3').hover(  
        function() {$(this).addClass('hover');},  
        function() {$(this).removeClass('hover');}  
    );  
});
```

4.3.4 事件的旅程

当页面上发生一个事件时，每个层次上的元素都有机会处理这个事件。事件的处理策略有两种：**Netscape**的事件捕获和微软的事件冒泡。在事件捕获的过程中，事件首先会交给最外层的元素，接着再交给更具体的元素。见下图：



相反，在事件冒泡过程中，事件首先发送给最具体的元素，在这个元素获得响应机会之后，事件会向上冒泡到更一般的元素。见下图：





4.3.4 事件的旅程

DOM标准同时使用这两种策略：首先，事件从一般元素到具体元素逐层捕获，然后，事件再通过冒泡返回DOM树的顶层。而事件处理程序可以注册到这个过程中的任何一个阶段。

为了提供跨浏览器的一致性，jQuery始终会在模型的冒泡阶段注册事件处理程序。



4.3.5 改变事件的传播过程

DOM提供了事件对象，其中包含与事件有关的信息，也提供了可以用来影响事件在DOM中传播过程的一些方法。为了使用事件对象，需要为事件处理函数添加一个参数。例如：

```
$(document).ready(function() {  
    $('#switcher').click(function(event) {  
        $('#switcher button').toggleClass('hidden');  
    });  
});
```

1、事件目标

事件对象中包含着目标元素的信息，即属性.target。

```
$(document).ready(function() {  
    $('#switcher').click(function(event) {  
        if (event.target == this) {  
            $('#switcher button').toggleClass('hidden');  
        }  
    });  
});
```




4.3.5 改变事件的传播过程

2、停止事件传播

事件对象提供了一个`.stopPropagation()`方法，可以阻止事件冒泡。这样就可以不使用`event.target == this`，仍然得到正确的结果。

/* 在4-3-6.css中 */

```
$(document).ready(function() {  
    $('#switcher').click(function(event) {  
        $('#switcher button').toggleClass('hidden');  
    });  
});  
$(document).ready(function() {  
    $('#switcher-default').addClass('selected');  
    $('#switcher button').click(function(event) {  
        var bodyClass = this.id.split('-')[1];  
        $('body').removeClass().addClass(bodyClass);  
        $('#switcher button').removeClass('selected');  
        $(this).addClass('selected');  
        event.stopPropagation();  
    });  
});
```



4.3.5 改变事件的传播过程

3、默认操作

有些元素有默认操作，例如a元素的默认操作是加载一个新页面。这种默认操作是事件的`.stopPropagation()`方法所无法阻止的，因为默认操作不是在正常的事件处理传播中发生的。

事件的`.preventDefault()`方法则可以在触发默认操作之前终止它。

事件传播和默认操作是相互独立的两套机制，任何一个都可以在另一个发生时被停止。可以在事件处理程序最后返回**false**，这是在事件对象上同时调用`.stopPropagation()`和`.preventDefault()`的一种简写方法。



4.3.5 改变事件的传播过程

4、事件委托

事件冒泡并不是总带来问题。事件委托就是利用了冒泡的事件传播特性，借助一个父元素上的事件处理程序完成很多工作。例如：

```
/* 在4-3-7.js中 */
$(document).ready(function() {
    $('#switcher').click(function(event) {
        if ($(event.target).is('button')) {
            var bodyClass = event.target.id.split('-')[1];
            $('body').removeClass().addClass(bodyClass);
            $('#switcher button').removeClass('selected');
            $(event.target).addClass('selected');
            event.stopPropagation();
        }
    });
});
```

这里的`.is()`方法接受一个选择符表达式，然后用选择符来检测当前的jQuery对象。如果集合中有至少一个元素与选择符匹配，`.is()`返回`true`。它比`.hasClass()`使用范围更大。



4.3.5 改变事件的传播过程

5、事件委托的方法

jQuery专门提供了一组利用事件委托的方法，包括：`.live()`、`.die()`、`.delegate()`和`.undelegate()`。例如：

```
$('#switcher button').live('click', function() {  
    var bodyClass = event.target.id.split('-')[1];  
    $('body').removeClass().addClass(bodyClass);  
    $('#switcher button').removeClass('selected');  
    $(this).addClass('selected');  
});
```

在后台，jQuery会把单击处理程序绑定到`document`身上，并且检查`event.target`（或其父元素）是否与传入的选择符表达式（`#switcher button`）匹配。如果匹配，jQuery会把关键字`this`映射到匹配的元素上。这样就避免了把事件绑定到多个子元素，提高了代码运行效率。而且`.live()`方法可以将事件处理程序绑定到将来插入到DOM中的元素。



4.3.6 移除事件处理程序

移除事件处理程序可以使用`unbind()`方法。例如：

`/* 在4-3-8.css中 */`

```
$(document).ready(function() {  
    $('#switcher').click(function(event) {  
        if (!$ (event.target).is('button')) {  
            $('#switcher button').toggleClass('hidden');  
        }  
    });  
    $('#switcher-narrow, #switcher-large').click(function() {  
        $('#switcher').unbind('click');  
    });  
});
```

这里的`.unbind()`把`#switcher`上所有的`click`事件都取消了，在本例中也影响到了按钮上的`click`事件。



4.3.6 移除事件处理程序

1、事件的命名空间

为了使对`.unbind()`的调用更具有针对性，可以使用事件的命名空间。方法是为事件增加命名空间后缀。例如：

```
/* 在4-3-9.css中 */
$(document).ready(function() {
    $('#switcher').bind('click.collapse', function(event) {
        if (!$ (event.target).is('button')) {
            $('#switcher button').toggleClass('hidden');
        }
    });
    $('#switcher-narrow, #switcher-large').click(function() {
        $('#switcher').unbind('click.collapse');
    });
});
```

对事件处理系统而言，后缀`.collapse`是不可见的。就是说，这里仍然会像编写`.bind('click')`一样，让注册的函数响应单击事件。但在绑定和解除绑定时，这个后缀却有意义。



4.3.6 移除事件处理程序

2、重新绑定事件

已经移除的事件响应可以重新绑定。这时应当使用命名函数。例如：

/* 在4-3-10.css中 */

```
$(document).ready(function() {  
    var toggleSwitcher = function(event) {  
        if (!$ (event.target).is('button')) {  
            $('#switcher button').toggleClass('hidden');  
        }  
    };  
    $('#switcher').bind('click', toggleSwitcher);  
    $('#switcher button').click(function() {  
        $('#switcher').unbind('click', toggleSwitcher);  
        if (this.id == 'switcher-default') {  
            $('#switcher').bind('click', toggleSwitcher);  
        }  
    });  
});
```



4.3.6 移除事件处理程序

如上例所示，使用命名函数不仅可以重复利用，而且可以带来另外一个好处：不必再使用事件命名空间，因为`unbind`函数将一个函数作为第二个参数时，它只会移除对该处理函数的绑定。

对于只需触发一次，随后就要立即解除绑定的函数还有一种简写绑定方法：`.one()`，例如：

```
$('#switcher').one('click', toggleSwitcher);
```




4.3.7 模仿事件操作

可以通过程序模拟引发事件，以便调用相应的事件处理程序。这需要调用`.trigger()`方法。例如：

```
$(document).ready(function() {  
    $('#switcher').trigger('click');  
});
```

`.trigger()`方法具有与`.bind()`方法同样的简写方式。当使用不带参数的事件名函数时，结果将是触发操作而不是绑定操作。例如：

```
$(document).ready(function() {  
    $('#switcher').click();  
});
```

键盘事件：

键盘事件与鼠标事件稍有不同。键盘事件可以分为两类：直接对键盘按键给出的响应事件(**keyup**和**keydown**)和对文本输入给出的响应事件(**keypress**)。

如果只想知道用户按了几个键中的哪个键，应该侦听**keyup**和**keydown**事件；如果想知道用户输入的是什么字符，应当侦听**keypress**事件。



4.3.7 模仿事件操作

键盘事件对象的`keyCode`属性中包含着被按下的那个键的标识符。对字母键来说，这个标示符就是相应大写字母的**ASCII**值。

键盘事件的目标对象是当前具有输入焦点的元素，如果对此并不关心，可以通过冒泡，将事件绑定到**document**对象上，例如：

```
/* 在4-3-11.css中 */
$(document).ready(function() {
    var triggers = {
        D: 'default',
        N: 'narrow',
        L: 'large'
    };
    $(document).keyup(function(event) {
        var key = String.fromCharCode(event.keyCode);
        if (key in triggers) {
            $('#switcher-' + triggers[key]).click();
        }
    });
});
```



4.4 样式与动画

在前面几节中，为了修改文档的外观，都是先在单独的样式表中为类定义好样式，然后再通过jQuery来添加或者移除这些类。一般而言，这是为HTML应用CSS样式的首选方式。

但jQuery还提供了更加灵活的控制样式的方式。

4.4.1 修改内联CSS

jQuery提供了.css()方法，用于获取和设置jQuery对象的CSS属性：

```
.css('property'); //获取CSS样式
```

```
.css('property', 'value'); //设置CSS样式
```

其中的属性名既可以用连字符形式的CSS表示法(如：background-color)，也可以是JavaScript中使用的驼峰表示法(如：backgroundColor)，而且驼峰表示法的属性名可以省略引号。

设置属性时，也可以通过使用由属性-值对构成的对象常量，同时设置多个CSS属性，即：

```
.css({property1: 'value1', 'property2': 'value2'});
```

4.4.1 修改内联CSS

//在例4-4-1.html中:

```
<div id="switcher">
```

```
  <div class="label">Text Size</div>
```

```
    <button id="switcher-default">Default</button>
```

```
    <button id="switcher-large">Bigger</button>
```

```
    <button id="switcher-small">Smaller</button>
```

```
</div>
```

```
<div class="speech">
```

```
  <p>Fourscore and seven years ago our fathers brought  
    forth on this continent a new nation, conceived in  
    liberty, and dedicated to the proposition that all men  
    are created equal.
```

```
</p>
```

```
</div>
```

```
</div>
```

Abraham Lincoln's Gettysburg Address

Text Size

Default

Bigger

Smaller

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.



4.4.1 修改内联CSS

//在例4-4-1.js中:

```
$(document).ready(function() {  
    var $speech = $('div.speech');  
    var defaultSize = $speech.css('fontSize');  
    $('#switcher button').click(function() {  
        var num = parseFloat($speech.css('fontSize'));  
        switch (this.id) {  
            case 'switcher-large':    num *= 1.4;  
            break;  
            case 'switcher-small':    num /= 1.4;  
            break;  
            default: num = parseFloat(defaultSize);  
        }  
        $speech.css('fontSize', num + 'px');  
    });  
});
```

4.4.2 基本的隐藏和显示

可以使用`.css('display', 'string')`来控制元素的显示或隐藏，其中string可以是none、block或inline。但jQuery提供了更好的方法：`.hide()`和`.show()`。

这两个方法的好处是：`.hide()`在将内联属性设置为`display:none`之前，会记住原来的`display`属性值；而`.show()`会将匹配元素的`display`属性恢复为应用`display:none`之前的显示状态，而不是简单地设置为`display: block`。

在例4-4-2.html中

```
<a href="#" class="more">read more</a>
```

Abraham Lincoln's Gettysburg Address

Text Size

Default

Bigger

Smaller

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

[read more](#)

The brave men, living and dead, who struggled here have consecrated it, far above our poor power

4.4.2 基本的隐藏和显示

在例4-4-2.js中:

```
$('#p').eq(1).hide();  
$('#a.more').click(function() {  
    $('#p').eq(1).show();  
    $(this).hide();  
    return false;  
});
```

其中:

- `.eq()`方法与自定义伪类:`eq()`相似, 它返回包含一个元素的jQuery对象。
- `return false;`用来避免默认的连接动作

Abraham Lincoln's Gettysburg Address

Text Size

Default

Bigger

Smaller

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that the nation might live. It is altogether fitting and proper that we should do this. But, in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow, this ground.

The brave men, living and dead, who struggled here have consecrated it, far above our poor power



4.4.3 动画效果

1、指定显示速度

jQuery有3种预设的速度常量：'slow'、'normal'和'fast'。如果想指定更精确的速度，可以使用毫秒值。

可以为hide()和show()方法指定速度参数来产生动画效果。例如：.show('slow')会在0.6秒钟内完成显示，.show('normal')是0.4秒，.show('fast')则是0.2秒。另外还可以指定更精确的0.85秒：.show(850)。

注意：与字符串参数不同，数值参数可以不要引号。

在例4-4-3.js中：

```
$(document).ready(function() {  
    $('p').eq(1).hide();  
    $('a.more').click(function() {  
        $('p').eq(1).show('slow');  
        $(this).hide();  
        return false;  
    });  
});
```


4.4.3 动画效果

2、淡入淡出

带速度参数的`hide()`和`show()`方法是在同时改变元素的高度、宽度和不透明度，但这种效果有时显得过于花哨。

为此jQuery提供了`.fadeIn()`和`.fadeOut()`方法。

在例4-4-4.js中：

```
$(document).ready(function() {  
    $('p').eq(1).hide();  
    $('a.more').click(function() {  
        $('p').eq(1).fadeIn('slow');  
        $(this).hide();  
        return false;  
    });  
});
```

Abraham Lincoln's Gettysburg Address

Text Size

Default Bigger Smaller

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that the nation might live. It is altogether fitting and proper that we should do this. But, in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow, this ground.

The brave men, living and dead, who struggled here have consecrated it, far above our poor power

这两个方法与`hide()`和`show()`方法类似，但渐变的只有不透明度。

4.4.3 动画效果

3、滑上和滑下

使用`.fadeIn()`和`.fadeOut()`，有时会感到文档跳了一下。jQuery提供了`.slideDown()`和`.slideUp()`方法，它们仅渐进改变文档的高度，用于实现垂直滑入、滑出的效果。

在例4-4-5.js中：

```
$(document).ready(function() {  
    $('p').eq(1).hide();  
    $('a.more').click(function() {  
        $('p').eq(1).slideDown('slow');  
        $(this).hide();  
        return false;  
    });  
});
```

Abraham Lincoln's Gettysburg Address

Text Size

Default

Bigger

Smaller

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battlefield of that war. We have come to

The brave men, living and dead, who struggled here have consecrated it, far above our poor power



4.4.3 动画效果

4、复合效果

前面的例子经过修改可以实现显示与隐藏的切换。在例4-4-6.js中：

```
$(document).ready(function() {  
    var $firstPara = $('p').eq(1);  
    $firstPara.hide();  
    $('a.more').click(function() {  
        if ($firstPara.is(':hidden')) {  
            $firstPara.fadeIn('slow');  
            $(this).text('read less');  
        }  
        else {  
            $firstPara.fadeOut('slow');  
            $(this).text('read more');  
        }  
        return false;  
    });  
});
```



4.4.3 动画效果

其中的`.is()`方法前面已经讲过：它可接受一个选择符表达式，只要jQuery对象中有一个元素与选择符匹配，就返回`true`。这里`:hidden`是一个jQuery的自定义伪类选择符，用来检查匹配元素的可见性。

其实jQuery还提供了实现复合效果的方法。

例如，使用单参数的`.toggle()`方法就可以实现交替显示，效果与交替调用`.show()`和`.hide()`方法一样，而且也可以有时间参数。

另一个复合方法是`.slideToggle()`，该方法的效果与交替调用`.slideDown()`和`.slideUp()`方法一样，通过调节高度来显示或隐藏元素。例如下面的例子：



4.4.3 动画效果

//在例4-4-7.js中:

```
var $firstPara = $('p').eq(1);
$firstPara.hide();
$('a.more').click(function() {
    $firstPara.slideToggle('slow');
    var $link = $(this);
    if ($link.text() == 'read more') {
        $link.text('read less');
    } else {
        $link.text('read more');
    }
    return false;
});
```



4.4.4 创建自定义动画

除了预置的动画方法外，jQuery还提供了一个强大的`.animate()`方法，用于创建更加精细的自定义动画效果。它有两种形式。

第一种形式有4个参数：

- ① 一个目标样式表：用于指定目标样式。
- ② 可选的整体时间参数：指定完成动画总共需要的时间。
- ③ 可选的局部速率参数(**easing**)：指定不同时间段动画进行的快慢。
- ④ 可选的回调函数：将在动画执行之后作为本次动画内容被调用。

形如：

```
.animate({property1: 'value1', property2: 'value2'},  
        speed, easing,  
        function() {alert('The animation is finished.')}  
);
```

第二种形式只有2个参数：

- ① 一个目标样式表：用于指定目标样式。
- ② 一组可选项。

事实上，第二种形式是将第一种形式的后三个可选项放在了一起，并且增加了一些更复杂的选项。形如：



4.4.4 创建自定义动画

```
.animate({  
    property1: 'value1',  
    property2: 'value2'  
}, {  
    duration: 'value',  
    easing: 'value',  
    specialEasing: {  
        property1: 'easing1',  
        property2: 'easing2'  
    },  
    complete: function() {  
        alert('The animation is finished.');    },  
    queue: true,  
    step: callback  
});
```



4.4.4 创建自定义动画

1、使用.animate()创建预置动画效果

//例4-4-8.js模拟了.slideToggle()效果:

```
var $firstPara = $('p').eq(1);
$firstPara.hide();
$('a.more').click(function() {
    $firstPara.animate({height: 'toggle'}, 'slow');
    var $link = $(this);
    if ($link.text() == 'read more') {
        $link.text('read less');
    } else {
        $link.text('read more');
    }
    return false;
});
```

可以看出，.animate()方法对CSS属性提供了简写值（例如：toggle）使对预置动画方法的模拟更加容易。



4.4.4 创建自定义动画

2、为多个属性同时添加动画效果

//例4-4-9.js同时实现了滑动和淡入淡出效果:

```
var $firstPara = $('p').eq(1);
$firstPara.hide();
$('a.more').click(function() {
    $firstPara.animate({
        opacity: 'toggle',
        height: 'toggle'},
        'slow'
    );
    var $link = $(this);
    if ($link.text() == 'read more')
        $link.text('read less');
    else
        $link.text('read more');
    return false;
});
```



4.4.4 创建自定义动画

//例4-4-10.js以动画形式改变字体大小:

```
var $speech = $('div.speech');
var defaultSize = $speech.css('fontSize');
$('#switcher button').click(function() {
    var num = parseFloat($speech.css('fontSize'));
    switch (this.id) {
        case 'switcher-large': num *= 1.4;
            break;
        case 'switcher-small': num /= 1.4;
            break;
        default:
            num = parseFloat(defaultSize);
    }
    $speech.animate({fontSize: num + 'px'}, 'slow');
});
```

4.4.4 创建自定义动画

//例4-4-11.js同时实现了文本尺寸盒的移动、增加高度和边框宽度:

```
$('#div.label').click(function() {  
    var paraWidth = $('div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
    var switcherWidth = $switcher.outerWidth();  
    $switcher.css({  
        position: 'relative'  
    }).animate({  
        borderWidth: '5px',  
        left: paraWidth - switcherWidth,  
        height: '+=20px'  
    },  
    'slow');  
});
```

其中

.outerWidth()方法用来计算元素的外宽度。
height: '+=20px'用来增加元素的CSS高度。

Abraham Lincoln's Gettysburg Address

Text Size

Default

Bigger

Smaller

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.



4.4.5 排队效果

前面的实例同时改变多种CSS属性，体现的是并发效果。但有时希望效果一个一个地发生，即想要排队效果。

1、对一组元素进行排队

这只要通过连缀即可实现。

//例4-4-12.js排队实现了文本尺寸盒的移动、增加高度和边框宽度：

```
$('#div.label').click(function() {  
    var paraWidth = $('#div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
    var switcherWidth = $switcher.outerWidth();  
    $switcher  
        .css({position: 'relative'})  
        .animate({left: paraWidth - switcherWidth}, 'slow')  
        .animate({height: '+=20px'}, 'slow')  
        .animate({borderWidth: '5px'}, 'slow');  
});
```



4.4.5 排队效果

连缀可以对任意jQuery效果进行排队，而不仅限于.animate()方法。

//在例4-4-13.js中：

```
$('#div.label').click(function() {  
    var paraWidth = $('#div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
    var switcherWidth = $switcher.outerWidth();  
    $switcher  
        .css({position: 'relative'})  
        .fadeTo('fast', 0.5) //不透明度减至0.5  
        .animate({left: paraWidth - switcherWidth}, 'slow') //向右移动  
        .fadeTo('slow', 1.0) //恢复至完全不透明  
        .slideUp('slow') //隐藏  
        .slideDown('slow'); //显示  
});
```



4.4.5 排队效果

2、使用.animate()第二种形式

可以实现更复杂的效果。

//在例4-4-14.js中:

```
$('#div.label').click(function() {  
    var paraWidth = $('#div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
    var switcherWidth = $switcher.outerWidth();
```



4.4.5 排队效果

```
$switcher
.css({position: 'relative'})
.fadeTo('fast', 0.5)
.animate({
    left: paraWidth - switcherWidth
}, {
    duration: 'slow',
    queue: false
})
.fadeTo('slow', 1.0)
.slideUp('slow')
.slideDown('slow');
});
```

本例中，`.animate()`的第二个参数包含了**queue**选项，把该选项设置为**false**可将当前动画与前一个动画同时开始，而不是顺序开始。由于两个动画的速度不同，这是**.animate()**的第一种形式所做不到的。



4.4.5 排队效果

排队效果不能自动应用到其它的非动画方法上，例如`.css()`方法，即使把它们放置在连缀序列中的正确位置上，它们也会立即执行。

//在例4-4-15.js中：

```
.fadeTo('slow', 1.0)
.slideUp('slow')
.css({backgroundColor: '#f00'})
.slideDown('slow');
```




4.4.5 排队效果

把非动画方法添加到队列中的一种方式，就是使用`.queue()`方法。

//在例4-4-16.js中：

```
$('#div.label').click(function() {  
    var paraWidth = $('#div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
  
    ...  
    .fadeTo('slow', 1.0)  
    .slideUp('slow')  
    .queue(function(next) {  
        $switcher.css({backgroundColor: '#f00'});  
        next();  
    })  
    .slideDown('slow');  
});
```

在这里，`.queue()`方法把一个回调函数添加到元素的效果队列中。回调函数有一个函数参数，它用于将效果连接起来，否则动画将中断。



4.4.5 排队效果

3、处理多组元素

当为不同组的元素应用动画效果时，这些效果几乎会同时发生。

//在例4-4-17.js中：

```
$('#p').eq(2)
.css('border', '1px solid #333')
.click(function() {
    $(this).slideUp('slow').next().slideDown('slow');
});
$('#p').eq(3).css('backgroundColor', '#ccc').hide();
```

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

[read more](#)

The brave men, living and dead, who struggled here have consecrated it, far above our poor power

It is rather for us to be here dedicated to the great task remaining before us—that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion—that we here highly resolve that these dead shall not have died in vain—that this nation, under God, shall have a new birth of freedom and that government of the people, for the people, shall not perish from the earth.

注意：这里的点击事件针对的是第三段文本，而第二段文本起初是被隐藏的。



4.4.5 排队效果

为了对不同组元素的动画效果进行排队，jQuery为每个动画方法提供了回调函数。作为参数的回调函数将在动画执行之后作为本次动画内容被调用。

//在例4-4-18.js中：

```
$('#p').eq(2)
.css('border', '1px solid #333')
.click(function() {
    var $clickedItem = $(this);
    $clickedItem.next().slideDown('slow', function() {
        $clickedItem.slideUp('slow');
    });
});
$('#p').eq(3).css('backgroundColor', '#ccc').hide();
```



4.4.5 排队效果

有了回调函数就可以解决前面提到的对非动画方法的排队问题，而不使用.queue()方法。

//在例4-4-19.js中：

```
$('#div.label').click(function() {  
    var paraWidth = $('#div.speech p').outerWidth();  
    var $switcher = $(this).parent();  
  
    ...  
    .fadeTo('slow', 1.0)  
    .slideUp('slow', function() {  
        $switcher.css({backgroundColor: '#f00'});  
    })  
    .slideDown('slow');  
});
```



第五章 jQuery Mobile简介

5.1 jQuery Mobile是什么

jQuery Mobile是一个支持所有流行移动设备平台的统一的用户界面系统，是一个构建于jQuery之上的UI层。它是一个帮助开发者更容易地在移动设备和平板电脑上交付跨平台Web应用的框架，只使用标准的HTML5代码。

移动Web应用与通常的移动网站在目标上并不相同。一个Web应用通常会模仿原生的手机应用，有着更加事务性的用户界面。它虽然是由网页技术(HTML、CSS、JavaScript、AJAX)创建的，但向用户提供了一个类似应用程序的体验。

它不是面向所有移动应用的解决方案，但它比使用原生代码开发在跨平台方面具有优势，因为它使用W3C的标准。



5.2 jQuery Mobile初步

5.2.1 准备文档

1、系统文件

- jquery-1.9.1.min.js: jQuery核心JavaScript文件
 - jquery.mobile-1.3.2.min.js: jQuery Mobile核心JavaScript文件
 - jquery.mobile-1.3.2.min.css: jQuery Mobile核心CSS文件
- 对本地应用来说，需要将它们保存到项目的根目录下。
其它可能涉及的文件包括：
- jquery-XX.js
 - jquery.mobile-XX.js
 - jquery.mobile-XX.css
 - jquery.mobile.structure-XX.css
 - jquery.mobile.structure-XX.min.css
 - 图片文件夹



5.2.1 准备文档

2、在HTML5文档中

<head>

```
<meta charset="utf-8" />
```

```
<title>Your Title</title>
```

```
<link rel="stylesheet" href="jquery.mobile-1.3.2.min.css" />
```

```
<script type="text/javascript" src="jquery-1.9.1.min.js"></script>
```

```
<script type="text/javascript" src="jquery.mobile-1.3.2.min.js"></script>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

</head>

其中:

- **viewport**(视口)是页面在屏幕上所占的区域, 可以指定它的宽和高, 使它比屏幕的可见区域更大。
- **scale** (比例)指图形和文本的放大比例, 视口也随之放大。
可以指定禁止用户改变页面比例:

```
<meta name="viewport" content="width=device-width, initial-scale=1,  
user-scalable=no">
```



5.2.2 文档结构

jQuery Mobile的文档结构主要由页面(page)构成。

一个页面仅仅是一个带有data-role="page"属性的div元素。一个HTML文档可以包含多个页面。

使用简单的HTML标记（例如超链接标记a），可以连接到同一个HTML文档中的另一个页面，也可以连接到外部HTML文档中的页面。

1、 data-role属性

jQuery Mobile使用标准的HTML标记，如div标记。只要给这些标记增加data-role属性，就可以使其具有不同的作用。例如<div data-role="page">

HTML5有一个自定义数据属性的特性，可以用来在标记上添加任何形如data-*的属性。data-role只不过是jQuery Mobile使用该特性添加的一个自定义数据属性。jQuery Mobile中大量使用这种特性。

5.2.2 文档结构

2、主题（theme）

jQuery Mobile使用主题机制来定义用户界面的可视化表现。每一个界面元素（如页面、按钮或组件）都可以使用主题中不同的色样（swatch）。

所谓主题，是一组排版、样式以及颜色，它定义了一种风格。每个主题都包含一组色样。色样为元素显示提供了不同的选择，在应用中可以随时更换色样。色样由a到z的字母标识，默认主题包含了从a到e的定义（见下图），为自定义色样预留了很多字母。



默认主题的从a到e的色样



5.2.2 文档结构

色样通过在jQuery Mobile的HTML元素上使用data-theme属性来指定，例如：`data-theme="e"`。

色样具有CSS的层叠特性，如果为父元素指定了一个色样，除非另外指定，它的子元素也将使用这个色样。

使用默认主题的色样时，有一些默认的约定。见下表：

字母	描述	颜色
a	最高视觉优先级（缺省用于工具栏）	黑色
b	次最高视觉优先级	蓝色
c	基准优先级（适用于大多数情况）	银色
d	次最高优先级的替代方案	灰色
e	强调	黄色

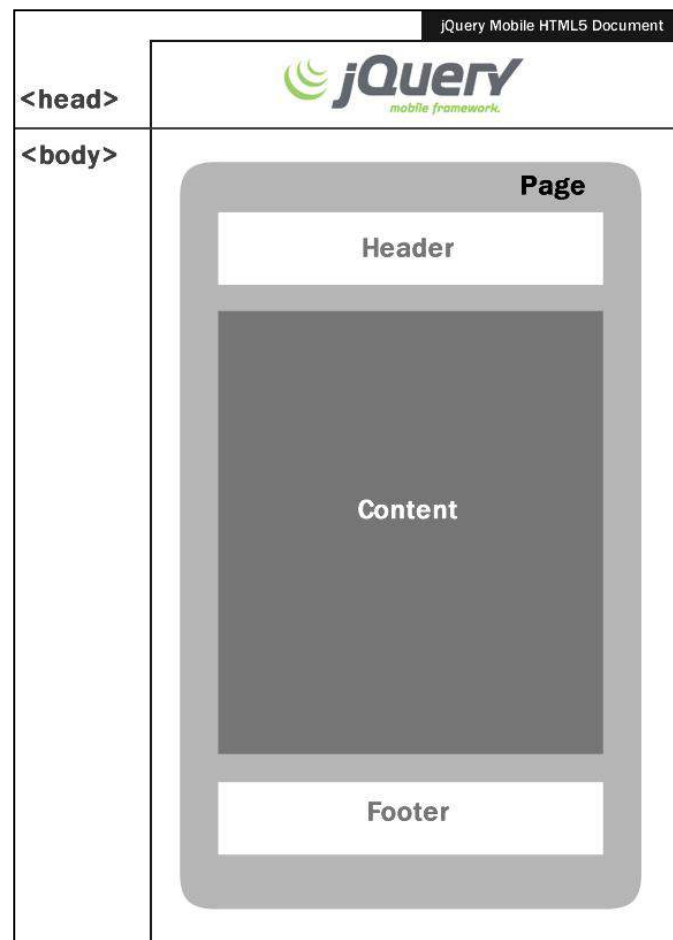
5.2.2 文档结构

3、页面(page)

页面是jQuery Mobile的主要单位。一个典型的页面包括页头、内容和页脚三个部分。通常只有内容部分是必不可少的。各个部分使用data-role属性指定。

```
<div data-role="page">  
  <div data-role="header">  
  </div>  
  <div data-role="content">  
  </div>  
  <div data-role="footer">  
  </div>  
</div>
```

当然，可以为每个部分指定不同的色样。



5.2.2 文档结构

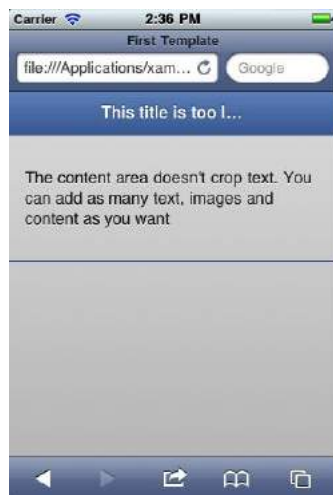
① 页头和页脚

在页头与页脚可以插入任何的HTML内容，但由于标准jQuery Mobile样式表的限制，最好在页头中使用h1，在页脚中使用h4，以便得到最好的显示效果。

在Web应用的导航界面中，页头通常是必须的，而页脚则是可选的。页头结构已经被预定义并划分为三个子区域：左侧、标题和右侧。

jQuery Mobile会自动从hx(例如h1)标签中提取标题。

由于空间有限，如果标题太长，它会被自动截断。在大多数设备上，被截断的标题末尾会显示省略号。页脚也有相同的行为。



5.2.2 文档结构

② 内容

在内容区域可以包含任何的HTML代码。一般会使用一些框架自带的带样式的控件，如按钮、列表或表单。例如：

```
<!-- 5-2-1.html -->
```

```
<div data-role="page">
```

```
  <div data-role="header">
```

```
    <h1>Our first webapp</h1>
```

```
  </div>
```

```
  <div data-role="content">
```

```
    <p>This is the main content of the page</p>
```

```
  </div>
```

```
  <div data-role="footer">
```

```
    <h4>More on mobilexweb.com</h4>
```

```
  </div>
```

```
</div>
```



注意：jQuery Mobile样式表无法处理那些在页面之内，而在页头、内容和页脚之外的内容，除非自己设计样式，否则界面将出问题。



5.2.3 导航

导航在jQuery Mobile中使用标准的HTML元素a来实现。

1、后退按钮

jQuery Mobile使用栈来维护访问过的页面，这样就可以随时返回访问过的页面。但只能后退，不能前进。

在页面上使用data-add-back-btn="true"属性将在页头左侧添加一个“后退”按钮。按钮的文本和色样可以使用页面上的data-back-btn-text和data-back-btn-theme属性来设置。如：

```
<div data-role="page" data-add-back-btn="true"  
    data-back-btn-text="Previous" data-back-btn-theme="e">  
</div>
```

jQuery Mobile会自动将一个指向前一个页面的链接行为转换为后退，点击时会显示一个后退动画，不会在浏览器访问历史中添加新纪录。

如果希望一个链接的行为后退，可以为这个a元素添加data-rel="back"属性，这时的href属性值无关紧要，而仅仅是用来保持一个超链接的下划线。

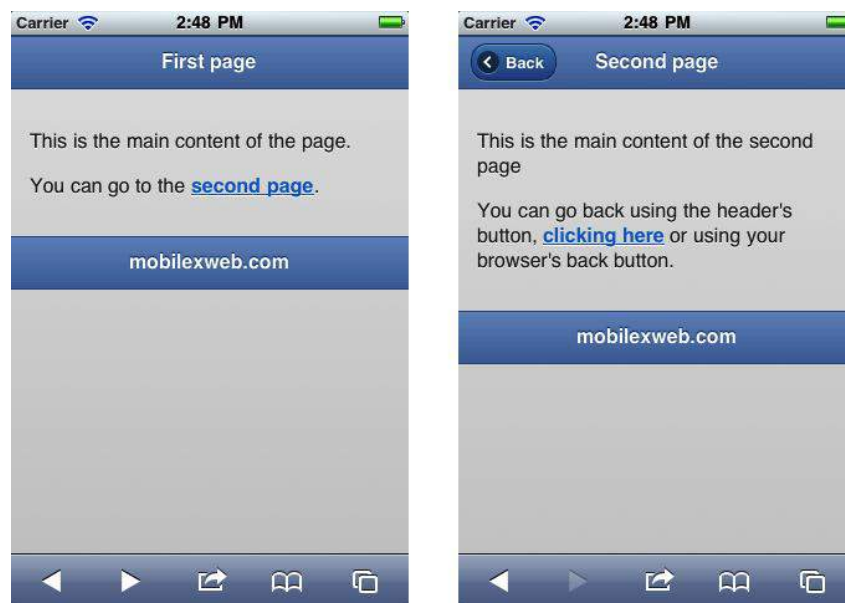
5.2.3 导航

2、内部页面链接

要实现页面内部的跳转，需要为每个页面设置一个id属性。然后在a标记的href属性中使用“#id”即可，例如：``。

默认情况下，jQuery Mobile使用页头中的标题(h1中的文本) 作为浏览器的标题。可以使用页面的可选属性data-title改变缺省标题。

见例：5-2-2.html





5.2.3 导航

3、外部页面链接

外部页面链接指链接到另一个**本地HTML**文档。例如：

```
<a href="otherDocument.html">Go to other html</a>
```

外部链接目标应该是一个jQuery Mobile文档，并且仅包含一个page页面，所以才称作外部页面链接。如果一个外部文档包含多个页面，jQuery Mobile只会从加载的文档中取出第一个页面，其它内容（包括head元素中的）都会忽略。

其实在内部，jQuery Mobile只是将该页面的内容以AJAX方式加入到当前的DOM中，并将该页面添加到已访问页面堆栈中。

见例：5-2-3.html

请求正在进行时，如果加载时间过长，用户会看到一个旋转的，表示正在加载的漂亮图标。

jQuery Mobile不支持下面这种链接：

```
<a href="other.html#otherpage">Go to other html</a>
```




5.2.3 导航

4、绝对外部链接

当需要链接到其它站点或非jQuery Mobile文档时，就需要使用绝对外部链接。它可以通过在a标记中添加data-rel="external"来实现：

```
<a href="http://www.mobilexweb.com" data-rel="external">  
    Check my blog</a>
```

还有些链接，比如指定了target属性或链接到不同域名的文档，无论这些文档是否jQuery Mobile文档，都会被当做绝对外部链接：

```
<a href="http://www.mobilexweb.com" target="_blank">  
    Check my blog</a>
```

```
<a href="http://www.otherdomain.com/whatever">Check my  
    blog</a>
```

另一个强制将链接转换为绝对外部链接的方法是在连接中使用data-ajax=false属性。这在链接本地多页面jQuery Mobile文档时有用：

```
<a href="otherpage.html" data-ajax="false">Other page</a>
```

实际上，在点击一个绝对外部链接时，jQuery Mobile将卸载当前实例，浏览器将转向指定目标。



5.2.3 导航

5、页面间的过渡效果

用户从一个页面转到另一个页面时，jQuery Mobile会使用一个平滑的动画过渡。缺省情况下，所有的过渡都使用从右到左的动画：在导航深入前进时，会看到从右到左的动画；返回到以前的页面时，则是从左到右的动画。

jQuery Mobile允许为每个链接指定使用动画的过渡效果。要实现这一点，需要在链接中使用自定义属性**data-transition**：

- **slide**：默认的从右到左的动画。
- **slideup**：从下到上的动画。
- **slidedown**：从上到下的动画。
- **pop**：从屏幕中心的一个小点开始变大，直到占满屏幕。
- **fade**：淡入淡出的渐变动画。
- **flip**：2D到3D的旋转动画。在某些设备上体现为2D旋转。

另外，每种过渡效果都有一个对应的用于后退的相反的效果。相反效果由链接上的自定义属性**data-direction="reverse"**实现。

见例：5-2-4.html



5.2.4 对话框

对话框是Web应用中显示页面的另一种方式。它与普通页面的最大区别是：

- 不是全屏显示，而是在原页面上弹出的窗口。
- 不会在访问历史中留下记录。
- 如果指定了**data-add-back-btn="true"**属性，则在左上角后退按钮的位置出现一个关闭按钮（一个叉）。

要打开对话框，只需在a标记中加上**data-rel="dialog"**属性。例如：

`删除此页`

在这里**data-rel="dialog"**指示将链接目标在对话框中打开，这是一种最常用的方式。

为了避免引起困惑，最好将对话框的默认过渡动画改为**slide**之外的效果，以便与层级页面之间的动画区别开来。

还可以使用**data-role="dialog"**而不是**data-role="page"**来生成对话框。它将目标本身指定为对话框。

对话框是出现在当前页面上的覆盖层，可以通过**data-overlay-theme**指定其色样。

5.2.4 对话框

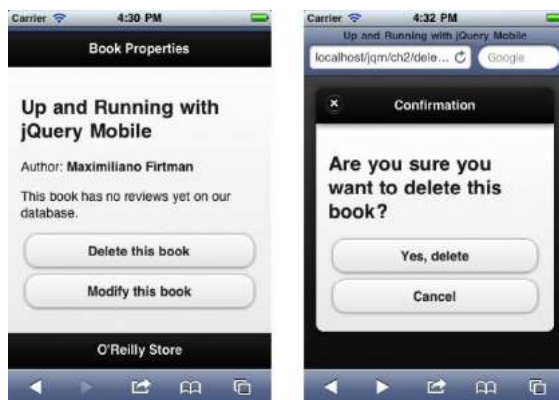
1、关闭对话框

以对话框方式打开的页面不能后退，只能关闭。

对话框属于打开它的页面，关闭对话框时，只应该返回原来的页面。

jQuery Mobile会将指向原页面的链接当作关闭处理，不会在访问历史记录中增加新的记录。

见例：[5-2-5.html](#)和[5-2-5-confirm.html](#)



2、从对话框打开页面

对话框可以有指向其它页面的普通链接。用户点击指向非原有页面的链接时，**jQuery Mobile**会关闭对话框，回到原有页面，然后打开新页面，就像这个链接本来就在原页面上一样。



5.2.4 对话框

3、JavaScript的对话框

不要忘了可以使用JavaScript的对话框，包括：`alert()`、`confirm()`和`prompt()`。特别是在调试程序时，它们特别有用。

4、弹出对话框

也称为弹出组件（**Popup Widget**）。它能够将包含在

中的任何内容，通过超链接弹出。

与对话框一样，它也不影响访问历史，因此称它为对话框。

为了实现此功能，div需要`data-role="popup"`，并且有一个id属性；而打开它的超链接需要`data-rel="popup"`，并且其链接目标为div的id值。例如：

```
<a href="#popupBasic" data-rel="popup">Open Popup</a>
<div data-role="popup" id="popupBasic">
  <p>This is a completely basic popup, no options set.</p>
</div>
```

这一组件的功能强大，可以各种形式弹出提示(**tooltip**)和图片等。见例5-2-6.html。



5.2.4 对话框

5、边栏

也称为面板组件(**Panel Widget**)。它与页眉、页脚和内容平级，不会影响浏览历史。通常写在页面的前部或后部，作为边栏滑出。

为了实现此功能，组件需要`data-role="panel"`，并且有一个`id`属性，其`id`值是打开它的超链接的链接目标。例如：

```
<a href="#mypanel">Open panel</a>
<div data-role="panel" id="mypanel">
  <!-- panel content goes here -->
</div>
```

打开面板的超链接是以切换(**toggle**)方式绑定的，再次点击则关闭边栏。其实关闭边栏有很多方法，包括滑动、在空白处点击等。

边栏有一个固定的宽度:17em(272 pixels)，见例5-2-7.html。

设置边栏的宽度样式比较复杂，但如果需要这些CSS也可以重写。



5.2.5 与电话整合

实现这一点的一个方法是使用**URL**机制。

a标签的**href**属性中可以使用不同的协议。这些协议中有些是大多数设备都支持的，还有些则是依赖平台的（如**iOS**）。依赖平台的协议在创建本地**Web**应用并知道它会在什么平台上运行时很有用。

见例：5-2-8.html

1、拨打电话

首选的方法是使用**tel:<电话号码>**方案：

Call me!

建议插入的电话号码使用国际格式：加号(+)，国家代码，区号，最后是本地号码。

用户激活电话链接时将看到一个确认页面，询问是否要拨打这个电话，以防止用户拨打国际电话或收费电话。

如果手机不能拨打国际电话，也可以在这里删除国家代码。



5.2.5 与电话整合

2、发送电子邮件

需要使用mailto:协议，语法是：

mailto:<destination email>[?parameters]

方括号表示可选参数。

不同的设备支持不同的参数，但基本上都包含**cc**（抄送）、**bcc**（暗送）、**subject**（标题）以及**body**（信件内容）参数。

参数以URL格式(**key=value&key=value**)定义，值必须经过URI编码。

例如：

```
<a href="mailto:info@mobilexweb.com">Mail us</a>
```

```
<a href="mailto:info@mobilexweb.com?
```

```
  subject=Contact%20from%20mobile">Mail us</a>
```

```
<a href="mailto:info@mobilexweb.com?subject=Contact&
```

```
  body=This%20is%20the%20body">Mail us</a>
```

通常，用户点击发送电子邮件链接时，支持的手机将打开发送邮件应用，相应字段也已填充。



5.2.5 与电话整合

3、发送消息

对支持jQuery Mobile的智能手机，可以安全地使用sms://协议。发送消息的语法是sms://[<destination number>][?parameters]方括号表示可选，其参数格式与发送电子邮件一样。例如：

```
<a href="sms://">Send an SMS</a>
```

```
<a href="sms://?
```

```
    body=Visit%20the%20best%20site%20at%20http://mobilex  
    web.com">Invite a friend by SMS<a>
```

```
<a href="sms://+3490322111">Contact us by SMS</a>
```

```
<a href="sms://+3490322111?
```

```
    body=Interested%20in%20Product%20AA2">More info for  
    product AA2</a>
```

用户点击发送消息链接时，只是打开编辑短消息的窗口，相应字段也已填充，用户必须手动完成整个过程。



5.3 UI组件

jQuery Mobile为Web应用准备了相当多的UI组件。尽管可以使用普通的HTML和CSS来添加内容及实现各种创意，但为了兼容各个平台，建议优先使用jQuery Mobile提供的各种组件。

jQuery Mobile组件可以分为以下几类：

- 工具栏组件
- 格式化组件
- 按钮组件
- 列表组件
- 表单组件

本章介绍前三类组件，列表和表单组件有独立的两章介绍。



5.3.1 工具栏

工具栏就是页头和页脚，它们是可选的，但基本上每个Web应用都有页头。

1、定位

不论是页头或页脚，工具栏都可以有4种方法定位：

- 内联模式
- 固定模式
- 全屏下的固定模式
- 真正的固定模式

- ① 内联模式：是工具栏的默认模式，工具栏位于页面的顶部或底部，随页面滚动。如果页面的长度超出可见高度时页脚将被隐藏，直到向下拖动滚动条，同时页头只在滚动条在顶部时才可见。
- ② 固定模式：工具栏不随页面滚动，总是浮在可见区域的顶部或底部。页面滚动时，工具栏会自动淡出隐藏，滚动完成时，工具栏会再次在顶部或底部出现。

要定义一个固定工具栏，只需在页头或页脚中使用 `data-position="fixed"` 属性。缺省属性是 `data-position="inline"`。



5.3.1 工具栏

当用户点击页面上某个非交互的内容区域时，固定模式和内联模式将相互切换，以便在需要时有更多的空间来显示内容。

- ③ 全屏下的固定模式：在全屏模式下（页面具有 `data-fullscreen="true"` 属性），固定工具栏只在用户点击屏幕时才出现，再次点击则消失。例如：

```
<div data-role="page" data-fullscreen="true">
  <div data-role="header" data-position="fixed">
  </div>
  <div data-role="content">
  </div>
  <div data-role="footer" data-position="fixed">
  </div>
</div>
```

固定工具栏可见时，内容会被遮挡。在大多数浏览器中，页头会使用半透明，可以透过页头看到后面的内容。



5.3.1 工具栏

- ④ 真正的固定模式：框架将创建一个较小的滚动区域来代替全屏，这样工具栏就可以总是位于一个位置。

使用该模式，首先需要在页头或页脚中使用 `data-position="fixed"` 属性，然后使用后面介绍的JavaScript API开启 `touchOverflow` 特性，通常这个特性是关闭的：

```
$.mobile.touchOverflowEnabled = true;
```

开启该特性后，不支持它的浏览器将降级为固定工具栏。

5.3.1 工具栏

2、在页头中添加内容

标准的jQuery Mobile页头包含一个

标题，以及一到两个可选的位于页头两侧的按钮。也可以按需要定制页头的外观和内容。

① 添加按钮

通常在右侧使用肯定动作按钮，在左侧使用否定动作按钮。

页头的按钮就是一个超链接。如果只提供了一个元素，则它将位于左侧，如果添加了两个按钮，则第一个位于左侧，第二个位于右侧。如果想强制指定按钮的位置，可以使用CSS类：class="ui-btn-right"或class="ui-btn-left"。

例如：

```
<div data-role="header">  
<a href="logout">Log out</a>  
<h1>Title</h1>  
<a href="settings" data-icon="gear">Settings</a>  
</div>
```



5.3.1 工具栏

默认情况下，工具栏的每个按钮都继承工具栏的主题。如果想让一个按钮与众不同，可以使用**data-theme**来改变它的色样。

例如：

```
<div data-role="header">
```

```
<a href="cancel" data-icon="delete">Cancel</a>
```

```
<h1>New record</h1>
```

```
<a href="save" data-icon="check" data-theme="b">Save</a>
```

```
</div>
```



如果想自定义一个后退按钮，一定要在对应的a元素上使用**data-rel="back"**属性；如果已经在页面中使用**data-add-back-btn="true"**属性添加了后退按钮，则不应该再在左侧添加其它按钮。

5.3.1 工具栏

② 添加logo

使用图片或logo代替文本是很常见的。在带有标准左右按钮的页头中显示图片，最好的方式是在对应的h1中使用img标记，例如：

```
<div data-role="header">  
<a href="cancel" data-icon="delete">Cancel</a>  
<h1></h1>  
<a href="save" data-icon="check" data-theme="b">Save</a>  
</div>
```

页头会自动适应图片的高度，但不要使用高度超过125像素的图片。



5.3.1 工具栏

③ 自定义页头

如果不想要典型的jQuery Mobile页头渲染效果，可以禁用自动页头的行为（如对**hx**和**a**的处理），方法是将页头的内容放在一个块容器（通常为一个**div**元素）中。例如：

```
<div data-role="header">  
<div>  
<h1>A custom title</h1>  
<a href="#">A non-button link</a>  
</div>  
</div>
```

自定义页头需要自己编写HTML及CSS代码处理。页头的高度会自动适应自定义的内容。

A custom title

[A non-button link](#)

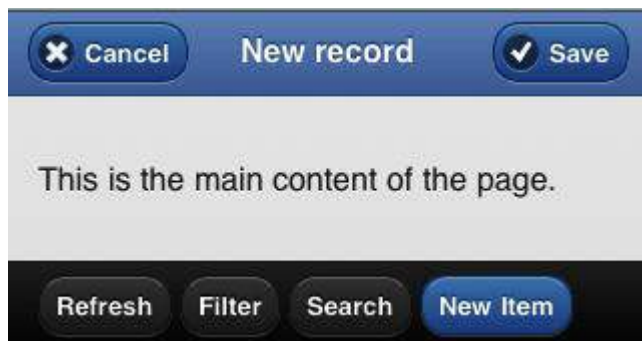
5.3.1 工具栏

3、在页脚中添加内容

页脚远比页头灵活。和页头一样，每个a元素都会被渲染为按钮，页脚中没有左右按钮位，每个按钮都以内联（**inline**）方式添加，一个接一个。

默认情况下，jQuery Mobile不会在按钮与页脚边框之间添加内边距。为了让视觉效果更好一些，应该在页脚上添加一个**ui-bar**类。例如：

```
<div data-role="footer" class="ui-bar">  
<a href="refresh">Refresh</a>  
<a href="filter">Filter</a>  
<a href="search">Search</a>  
<a href="add" data-theme="b">New Item</a>  
</div>
```





5.3.1 工具栏

4、导航栏

导航栏是一组专有链接，可放置在工具栏中，通常是页脚。导航栏是**Web**应用的主要导航结构，其中不应该放置普通的动作按钮。

导航栏只是一个容器，通常为一个**div**元素，其中包含各个导航链接的无序列表。容器的角色需要被指定为**navbar**：

```
<div data-role="navbar">  
<ul>  
<li><a href="link1">Option 1</a>  
<!-- ... -->  
<li><a href="linkn">Option n</a>  
</ul>  
</div>
```

导航按钮与标准按钮的外观不同。它们的宽度相同，占满整个导航栏，而且互斥。

如果按钮太多（**6**个或更多）则每行两个，以便适合触摸。

5.3.1 工具栏

① 使用图标

使用标准的jQuery Mobile图标机制，可以为每个导航元素添加图标。在默认情况下，图标位于文本的上方。例如：

```
<div data-role="header" data-position="fixed">
```

```
<h1>Home</h1>
```

```
<div data-role="navbar">
```

```
<ul>
```

```
<li><a href="#index" data-icon="home">Home</a>
```

```
<li><a href="#contacts" data-icon="search">Contacts</a>
```

```
<li><a href="#events" data-icon="info">Events</a>
```

```
<li><a href="#news" data-icon="grid">News</a>
```

```
</ul>
```

```
</div>
```

```
</div>
```



可以使用标准的图标名，jQuery Mobile也有创建自定义图标的标准方法，互联网上也有大量的免费图标，例如：<http://glyphish.com>。

5.3.1 工具栏

② 被选中的元素

每个导航栏都可以有一个被选中的元素，使用类名**ui-btn-active**标识。激活的元素会在当前主题的**UI**基础上高亮显示。

将一个导航栏作为**Web**应用的主导航时，最好将主页作为导航栏的第一个元素，并表示为已选中。例如：

```
<div data-role="footer" data-position="fixed">
<div data-role="navbar">
<ul>
<li><a href="#index" class="ui-btn-active">Home</a>
<li><a href="#contacts">Contacts</a>
<li><a href="#events">Events</a>
<li><a href="#news">News</a>
</ul>
</div>
</div>
```



用户操作导航元素时，当前选中元素会自动切换为选中状态，不用自己更新**ui-btn-active**类。

5.3.1 工具栏

5、固定页脚

当页面切换时，页脚也会随之运动。如果使用外观一致的导航栏，这种效果是不希望出现的。为此可以使用固定页脚。固定页脚一旦定义，将不会随页面的改变而改变，不会出现视觉问题。

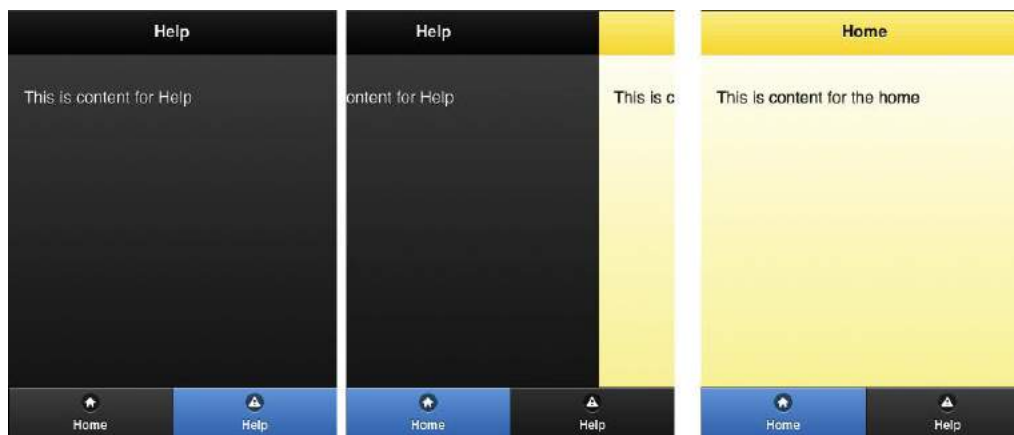
要创建固定页脚，需要在所有需要它的页面的页脚中添加具有相同值的**data-id**属性，还需要这些页脚具有相同的页脚代码。

为保证选中状态，在使用固定页脚的导航栏时，需要在选中项上加两个类：**ui-btn-active ui-state-persist**。“持续”指不止第一次如此。

另外固定页脚应该使用**data-position="fixed"**定义为固定定位。

见例：5-3-1.html

注：jQuery Mobile中没有官方支持的定义固定页头的方法。





5.3.2 格式化内容

有一点很重要：在内容部分，可以包含任何**HTML**代码。

jQuery Mobile的每个主题都包含良好的样式，其中每个标准元素的内边距、外边距、尺寸及颜色都已针对当前主题及移动设备优化过。被定义过样式的元素包括**hx**、链接、粗体及斜体文本、引用、列表以及表格等。

如果想为内容提供自定义的**CSS**样式，需要注意主题系统，以避免修改主题时引起的界面问题。

除了基本的**html**元素，**jQuery Mobile**也提供了一些使用**data-role**定义的组件。

在页头和页脚处，诸如**hx**和**a**在内的一些基本**html**元素具有自动组件渲染行为。

而在内容区域，除了一些表单元素外，其它基本**html**元素都没有自动组件渲染行为，需要显式使用组件。

5.3.2 格式化内容

1、可折叠内容

在移动设备上，空间非常有限，可折叠功能非常重要。

可折叠内容可以通过关联的JavaScript行为，在触摸某个标题或按钮时隐藏或显示。

jQuery Mobile已经为这种UI设计模式提供了支持，不需要我们再编写JavaScript。

要创建可折叠内容，只需要定义一个带有data-role="collapsible"属性的容器。这个容器需要一个hx元素作为标题，同时用作开关按钮。可折叠内容则是容器内除了标题外的任何html代码。

默认情况下，页面加载时jQuery Mobile会打开折叠内容。可以在容器上使用data-collapsed="true"来让它默认收起。

见例：5-3-2.html





5.3.2 格式化内容

和其它控件一样，可以使用**data-theme**属性来改变这个可折叠面板的色样。还可以指定额外的**data-content-theme**属性。这个属性只影响内容，而不影响可折叠面板的打开和收起按钮。

如果可折叠容器中没有定义**hx**元素，则内容始终处于打开状态，不能收起；如果定义了多个**hx**元素，则第一个用于标题按钮，其它的作为内部的内容。

可折叠内容可以嵌套，**jQuery Mobile**会自动在每一级可折叠面板上添加外边距，但一般不要超过两层。

见例：5-3-3.html

5.3.2 格式化内容

2、手风琴组件

手风琴组件可以将多个可折叠内容聚合起来，一次只有一个面板可见。

这种组件就是一个带有`data-role="collapsible-set"`属性的容器，以及一组作为子元素的可折叠面板。

默认情况下，jQuery Mobile会展开可折叠集合中的最后一个面板。如果想默认展开别的面板，只需在想展开的元素上加上`data-collapsed="false"`属性，而在其它面板上加上`data-collapsed="true"`。

见例：5-3-4.html





5.3.2 格式化内容

3、列

jQuery Mobile提供了一些用于将内容按列显示的模板，称为布局网格。可以定义2至5列的网格。网格是不可见的，占满100%的宽度，没有内外边距。

网格只是一个具有类ui-grid-**<letter>**的块容器，典型情况为div。类ui-grid-a表示两列；ui-grid-b表示三列；ui-grid-c表示四列；ui-grid-d表示五列。默认情况下，各列宽度相同。

每个单元格都是一个具有类ui-block-**<letter>**的块容器。其中**<letter>**的值为a到e，分别表示网格的第一到第五列。

见例：5-3-5.html

@firt		
Column 1	Column 2	
Cell 1.1	Cell 1.2	Cell 1.3
Cell 2.1	Cell 2.2	Cell 2.3



5.3.3 按钮

虽然[a](#)元素可用于创建各种链接，但它通常是内联的，只有文字部分是可点击区域，不适合触摸设备。因此，jQuery Mobile提供了按钮。

按钮是一种UI组件，它是一个带文本的更大的可点击区域，可能还包括图标。创建按钮的方式有好几种：

- 使用button元素。
- 使用input元素，带有type="button"或type="submit"等属性。
- 使用带有data-role="button"属性的a元素。

jQuery Mobile使用CSS3样式对元素进行渲染。

默认的情况下，按钮会占满整个屏幕宽度。

例如：

```
<a href="#" data-role="button">Click me!</a>
```

```
<button data-theme="b">Click me too!</button>
```

```
<input type="button" value="Don't forget about me!">
```

5.3.3 按钮

1、内联按钮

内联按钮不会占满整个屏幕宽度，可以在元素上使用属性 **data-inline="true"** 来实现。例如：

```
<a href="" data-role="button" data-inline="true">Button 1</a>
```

```
<a href="" data-role="button" data-inline="true">Button 2</a>
```

```
<a href="" data-role="button" data-inline="true">Button 2</a>
```



如果想要内联按钮占满整个屏幕宽度，可以使用前面介绍的布局网格，每列包含一个按钮。

5.3.3 按钮

2、分组按钮

jQuery Mobile提供了分组控件技术。借助它可以将一组彼此相关的按钮组合起来。

控件组是一个容器，通常是一个带有**data-role="controlgroup"**的div元素。其中的按钮不需要声明为内联。

分组按钮在垂直方向彼此相连，但并没有选中状态。例如：

```
<div data-role="controlgroup">  
<a href="#" data-role="button">Button 1</a>  
<a href="#" data-role="button">Button 2</a>  
<a href="#" data-role="button">Button 2</a>  
</div>
```



也可以通过在控件组上使用**data-type="horizontal"**属性来创建水平布局的分组控件。

5.3.3 按钮

3、显示效果

默认情况下，按个按钮都被渲染为带圆角和阴影。可以通过布尔值属性`data-corners`和`data-shadow`来改变这种行为：

```
<a href="#" data-role="button" data-shadow="false" data-corners="false">Help</a>
```

4、图标

每个按钮（还有一些其它组件）都可以通过`data-icon`属性定义一个图标。这个属性接受一个图标名，名字可以是图库中已有的某个图片名，也可以自定义。下面是jQuery Mobile框架中包含的所有图标：





5.3.3 按钮

5、创建自定义图标

jQuery Mobile的系统图标使用了Sprites技术。该技术的一个图标文件中有一组图标，以此来提高加载效率。

这里，我们准备的图标文件仅包含一个图标。具体要求是：

- 带alpha位面的PNG-8图像格式。
- 18*18像素。
- 白色或透明背景。其实任何背景都可以，jQuery Mobile会自动为图标添加白色背景。
- 不带边框。需要时，系统会自动添加。
- 图案应该比18*18像素小一些，因为要渲染到18像素的圆形中。
- 不需要边角图案，因为要绘制在圆形中，用不上。

另外还需要为高分辨率设备准备一个36*36像素的图标文件作为备选。因此一个图标需要准备两个图标文件。



5.3.3 按钮

准备好图标文件后，需要进行下列工作：

- ① 为每个自定义图标确定一个不重复的图标名，推荐的名称格式为：**<应用名>-<图标名>**，例如：**myapp-tweet**。
- ② 使用①中的图标名，创建一个**CSS**样式，其选择器为**.ui-icon-
<name>**，例如**.ui-con-myapp-tweet**，在其中指明使用的图标文件。
- ③ 使用**CSS3**媒体查询格式为两倍于标准分辨率的设备创建备选的相同选择器的**CSS**样式。需要强制背景尺寸为**18*18**，因为在这些设备上**px**是虚拟的点，每一个对应两个真实的像素。

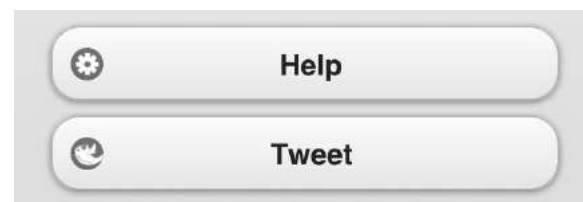
5.3.3 按钮

综合起来的CSS样式看起来类似下面：

```
<style>
.ui-icon-myapp-tweet {
    background-image: url(icons/tweet.png);
}
@media only screen and (-webkit-min-device-pixel-ratio: 2) {
    .ui-icon-myapp-tweet {
        background-image: url(icons-hd/tweet.png) !important;
        background-size: 18px 18px;
    }
}
</style>
```

至此，该自定义图标就可以使用了：

```
<a href="#" data-role="button" data-icon="gear">Help</a>
<a href="#" data-role="button" data-icon="myapp-
tweet">Tweet</a>
```



5.3.3 按钮

6、图标的位置

默认情况下，每个图标都被渲染在按钮文字的左侧。可以使用属性 `data-iconpos` 来改变这个位置，可能的值包括：`right`、`left`（缺省值）、`bottom` 和 `top`。见下例：

```
<a href="#" data-role="button" data-icon="help" data-  
    iconpos="right">Help</a>
```

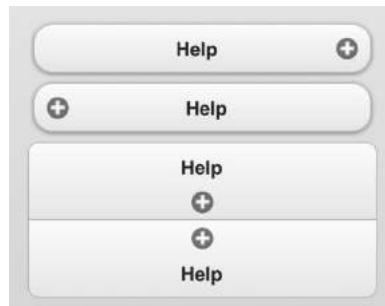
```
<a href="#" data-role="button" data-icon="help" data-  
    iconpos="left">Help</a>
```

```
<div data-role="controlgroup">
```

```
    <a href="#" data-role="button" data-icon="help" data-  
        iconpos="bottom">Help</a>
```

```
    <a href="#" data-role="button" data-icon="help" data-  
        iconpos="top">Help</a>
```

```
</div>
```





5.3.3 按钮

7、纯图标按钮

这种按钮对触摸界面来说太小了，因此很少使用。要创建一个纯图标按钮，只需在按钮上指定`data-iconpos="notext"`属性。所以`data-iconpos`属性还有第五个值！

8、图标阴影

jQuery Mobile有一个可删除图标阴影效果的属性。要实现这个效果，只需要在按钮上指定`data-iconshadow="false"`。



5.4 列表组件

列表在jQuery Mobile框架中功能强大而且用途广泛，几乎所有移动应用中都至少会包含一个列表。

在jQuery Mobile的世界里，列表就是一个具有data-role="listview"属性的html有序（ol）或无序（ul）列表。

见例：5-4-1.html

HTML5 and APIs
Offline Access
Geolocation API
Canvas
Offline Storage
New semantic tags

注意代码中的列表项li元素。由于html5没有使用XML语法，所以该元素可以没有关闭标记。



5.4.1 整页列表与插入列表

默认情况下列表占据整个页面，称为整页列表。如果希望与其它html内容混排，可以使用插入列表。这只需要在对应的列表元素上添加data-inset="true"属性即可。

见例：5-4-2.html

HTML5 and APIs

This is a list of HTML5 new features

Offline Access

Geolocation API

Canvas

Offline Storage

New semantic tags



5.4.2 视觉分隔符

视觉分隔符就是带有属性`data-role="list-divider"`的li元素。

使用这个技术可以将列表元素划分为不同的组。可以通过改变色样高亮显示每个视觉分隔符。

见例：5-4-3.html

World Cup
Group A
Argentina
Nigeria
England
Japan
Group B
United States
Mexico
Korea
Greece



5.4.3 交互行

如果一个列表元素包含一个a元素，则该行成为一个能响应触摸或鼠标点击的交互行。jQuery Mobile会自动在交互行的右侧添加漂亮的箭头（可通过data-icon属性改变或取消data-icon="false"），提醒用户该行是可触摸的。

交互行的高度也与只读行不同，经过了优化设计，更有利于触摸。如果某交互行的文本超过一行，则行末的文本会被省略号代替。并且选中的行可能被添加焦点边框。

见例：5-4-4.html

Interactive	
Internal Page link	>
External Page link	>
Absolute external link	>
Call to a phone number usi...	>
JavaScript link	>

5.4.3 交互行

1、内嵌列表

内嵌列表就是位于某一个jQuery Mobile列表的某一列表项（li）中的另一个jQuery Mobile列表。jQuery Mobile会隐式地为一个内嵌列表生成一个新的页面，并会自动处理各个层级的关系。

需要为内嵌列表所在的li标记中填写适当的文本。它不仅是列表项的标题，而且也是隐式创建的页面标题。

见例：5-4-5.html

Training	Back Cities available
Order Now! >	Boston
Cities available >	New York
Topics >	Miami
Promotions >	San Francisco
	San Jose

5.4.3 交互行

2、分裂行列表

如果一个jQuery Mobile列表的某一列表项（li）中包含两个超链接元素a，则它自动被当作分裂行处理。即同一行中包含两个交互操作。

按照iOS用户界面准则，第一个操作应该是详情，第二个操作应该是编辑。

默认情况下，jQuery Mobile会为右边第二个操作自动添加一个带边框的箭头图标，所以它不必带有文本。

见例：5-4-6.html

Your Friends	
Bill Gates	
Steve Jobs	
Mark Zuckerberg	
Larry Page	

5.4.3 交互行

可以在对应的

标签（而不是- 标签）上通过data-split-theme和data-split-icon来为第二个操作分别指定不同的主题和图标。

列表与按钮使用的是同一组图标，但同一个图标在列表中显示时多了一个圆角边框。

3、有序列表

有序列表的各个列表项会被自动编号。

见例：5-4-7.html

Chapters	
1. The Mobile Jungle	>
2. Mobile Browsing	>
3. Architecture and Design	>
4. Setting up your environm...	>
5. Markups and Standards	>
6. Coding Markup	>
7. CSS for Mobile Browsers	>

5.4.4 使用图片

每一行都可以用指定的图片来标识，而且可以添加两种不同类型的图片：图标和缩略图。

1、行图标

行图标是显示在列表项左侧的图标，而不是交互行右侧的箭头或分裂行列表中的图标。

行图标是位于li标记中的一个**16*16**像素的图片（不是jQuery Mobile中所说的图标），由ui-li-icon类定义。例如：

```
<li>
```

```
    
```

```
    Send by e-mail
```

行图标通常用于对可做操作的提示，例如：删除、编辑、共享等。

5.4.4 使用图片

2、缩略图

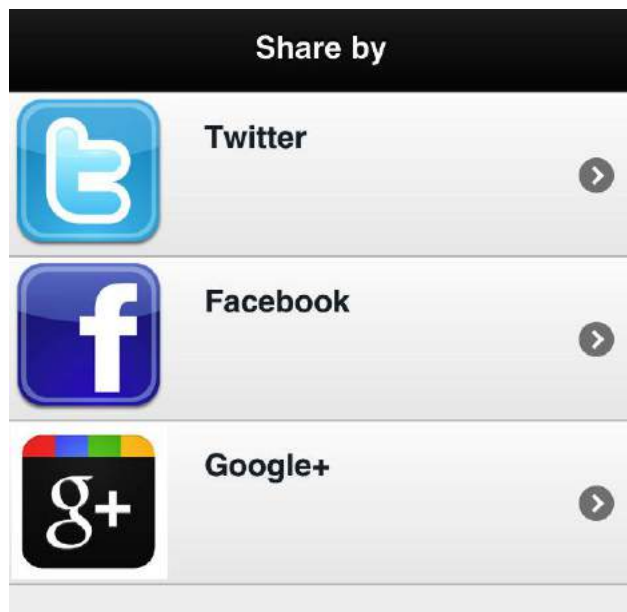
缩略图是一个80*80像素的图片，也显示在行标题的左侧。它是列表项中的图片定义，不需要任何特殊的类。例如：

``

``

George Washington

下图也是缩略图的例子：



5.4.5 附加内容

至此，每个列表都只有一列文本内容。虽然可以添加缩略图或图标，但文本列只有一列。可以为每一行添加一个次级列，用于显示补充信息。

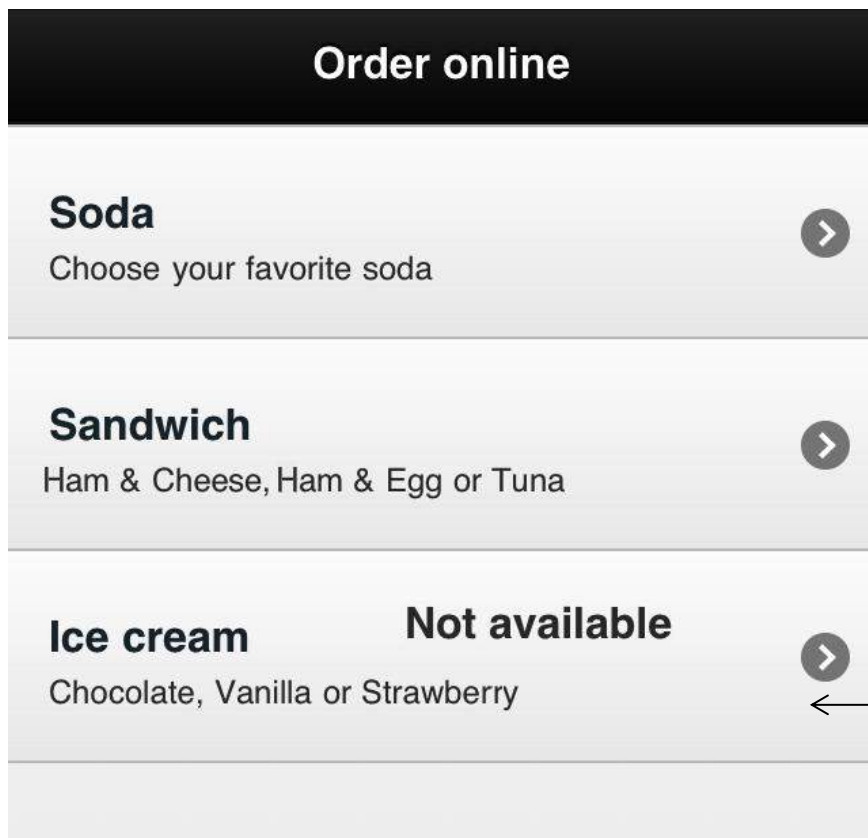
这个功能可以使用任何带有`ui-li-aside`类的HTML元素来实现，例如`span`或`div`元素。

见例：5-4-8.html

Order online		
Soda	\$1.00	>
Sandwich	\$3.20	>
Ice cream	\$1.50	>

5.4.6 标题与描述

如果想在在一行上同时显示标题与描述，可以将标题放在**hx**标签里，将描述文本放在**p**元素里。它们不会被显示为两列。如下图：



不会这样！

5.4.7 使用计数气泡

计数气泡是一个在行的右侧显示的包含数字的圆圈，通常用于显示该交互行包含多少项。

要使用计数气泡，只需在列表项（li）中包含任意带ui-li-count类的html元素，例如span元素。例如：

```
<li><a href="inbox.html">Inbox</a>  
    <span class="ui-li-count">86</span>
```

下图显示了计数气泡在一个交互列表中的显示效果。如果想改变默认的白色背景，可以在对应的ul元素中指定data-count-theme属性。

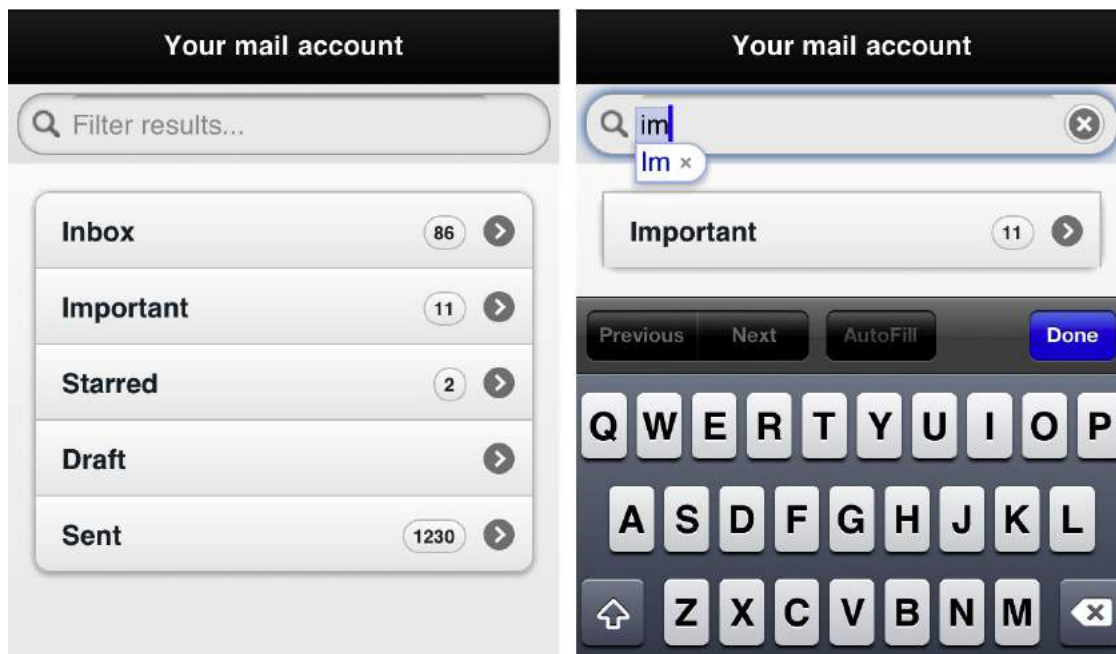
Your mail account		
Inbox	86	>
Important	11	>
Starred	2	>
Draft		>
Sent	1230	>

5.4.8 使用搜索过滤数据

在ul或ol标记内添加**data-filter="true"**属性，不用任何编码就可以在列表的顶部添加一个搜索框。它会根据用户的输入过滤列表元素。

可以在ul中使用**data-filter-placeholder**属性来自定义占位字符，或使用**data-filter-theme**属性来改变搜索栏的色样。

见例：5-4-9.html





5.5 表单组件

jQuery Mobile框架支持标准的HTML表单，在支持的设备上会自动使用AJAX处理，同时标准表单控件的外观也为触摸操作进行了优化。

所谓标准表单指的是位于form元素内的一组表单控件，如input、textarea以及select元素等。表单的数据将被发送到该form元素的action属性所指向的URL。



5.5.1 表单动作

jQuery Mobile表单的动作处理方式与原来一样：当用户按下提交按钮时，表单会被提交。

表单提交时，除非是提交到不同的域名，否则jQuery Mobile会使用AJAX方式过渡到目标页面，就像链接到一个外部页面一样。

表单的提交方式有post和get，由form元素的method属性指定。

get: 以URL的形式发送数据；post: 浏览器与服务器建立联系后，再将表单数据发往服务器。

典型的jQuery Mobile表单与标准的HTML表单很像：

```
<form action="send.php" method="get">
```

```
<!-- 表单控件 -->
```

```
</form>
```

提交后的结果页面也要求是一个jQuery Mobile文档，以便通过AJAX方式过渡。

也可以使用data-transition和data-direction改变表单的过渡方式：

```
<form action="send.php" method="get" data-transition="pop">
```

```
<!-- 表单控件 -->
```

```
</form>
```



5.5.2 表单元素

jQuery Mobile允许使用HTML5中的新表单控件，例如滑块（`input type="range"`），并且自动将表单控件替换为对触摸更为友好的富控件（`rich control`）。默认情况下，每个表单控件独占一行。

1、文本标签

文本标签（`label`）附着在表单控件上。它不仅作为提示，也扩大了所附着控件的焦点的选择范围。就是说，表单元素的可点击范围为控件自身加上`label`标签。所以，应当尽量为所有控件加上文本标签。

例如：

```
<label for="company">Company Name:</label>  
<input type="text" id="company">
```

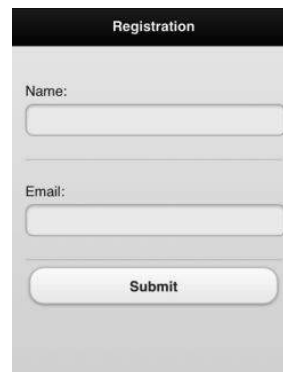
5.5.2 表单元素

2、域容器

域容器可以是任意带**data-role="fieldcontainer"**的块级元素。它会将其内部的文本标签和表单控件自动对齐，并且会自动添加一个细边框作为字段分隔符。

如果设备宽度比较窄，文本标签就会放置在表单控件的顶部，否则就使用两列布局。例如：

```
<div data-role="fieldcontainer">  
<label for="company">Company Name:</label>  
<input type="text" id="company" name="company">  
</div>  
<div data-role="fieldcontainer">  
<label for="email">Email:</label>  
<input type="email" id="email"  
      name="email">  
</div>
```

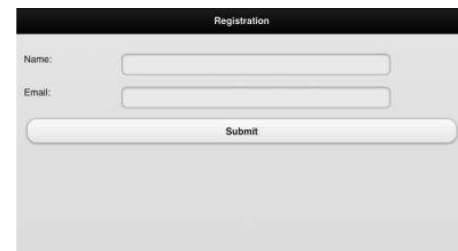


Registration

Name:

Email:

This image shows a registration form on a narrow device. The form has a title "Registration" at the top. Below it, there are two input fields: "Name:" and "Email:". The "Name:" label is positioned above the input field. The "Email:" label is positioned to the left of the input field. At the bottom of the form, there is a "Submit" button.



Registration

Name:

Email:

This image shows a registration form on a wide device. The form has a title "Registration" at the top. Below it, there are two input fields: "Name:" and "Email:". The "Name:" label is positioned to the left of the input field. The "Email:" label is positioned to the left of the input field. At the bottom of the form, there is a "Submit" button.



5.5.2 表单元素

3、文本输入框

jQuery Mobile支持基本的HTML5文本输入控件，并根据当前的主题和色样进行渲染。下面是可用的文本输入框：

- `<input type="text">`
- `<input type="password">`
- `<input type="email">`
- `<input type="tel">`
- `<input type="url">`
- `<input type="search">`
- `<input type="number">`
- `<textarea>`

使用上面的元素时会自动得到jQuery Mobile控件，不用显式指定data-role属性。

type为button、submit、reset和image的input元素，会被自动渲染为jQuery Mobile按钮。

5.5.2 表单元素

email、tel、url、search以及number是HTML5的新输入类型。在移动设备上，指定这些新输入类型时，将得到不同的经过优化的虚拟键盘。例如：



在jQuery Mobile中，search类型输入域的用户界面与其它类型也不相同：



5.5.2 表单元素

4、自增长文本区

`textarea`多行文本输入域有个额外的特性：自动增长。jQuery Mobile会以一个两行高的区域开始，随着输入文本的增加而变高。例如：

```
<div data-role="fieldcontainer">  
<label for="comments">Your comments:</label>  
<textarea id="comments" name="comments"></textarea>  
</div>
```





5.5.2 表单元素

5、HTML5新属性

可以在文本输入框中使用HTML5的新属性。例如：

- **required**: 布尔值，该输入域是必须的。
- **placeholder**: 当输入框中无值时，它将显示为灰色的提示。
- **min和max**: 仅用于<input type="number">，指定数值范围。

```
<div data-role="fieldcontainer">  
  <label for="name">Your Name:</label>  
  <input type="text" id="name" required placeholder="Enter your  
    name">
```

```
</div>
```

```
<div data-role="fieldcontainer">  
  <label for="age">Your Age:</label>  
  <input type="number" id="age" required placeholder="Enter  
    your age" min="10" max="110">
```

```
</div>
```

使用CSS3的伪类，不用编写JavaScript校验代码，就可为有效，无效、必填以及选填的文本输入框创建不同的样式。

5.5.2 表单元素

6、日期输入框

在HTML中输入日期一直是个问题。HTML5加入了对日期输入的支持，只需要使用下列类型的input元素：

- **date**: 日期选择（年/月/日）。
- **datetime**: 日期和时间选择（年/月/日/时/分），包含GMT时区。
- **time**: 时间选择（时/分）
- **datetime-local**: 日期和时间选择，但不包括时区信息
- **month**: 月份选择，通常是个下拉列表
- **week**: 星期选择，通常为下拉列表

这些日期输入框支持min和max属性。例如：

The image shows two screenshots of an iPhone registration form titled "Registration". The top screenshot shows the form with two input fields: "Your Birthdate:" and "Your favorite month:". The bottom screenshot shows the date picker interface for the "Your Birthdate:" field. The date picker displays a grid of months and years. The selected date is February 23, 1970. The bottom screenshot also shows the date picker interface for the "Your favorite month:" field, displaying a list of months and years. The selected month is June 2012.

Month	Day	Year
December	21	1968
January	22	1969
February	23	1970
March	24	1971
April	25	1972

Month	Year
April	2010
May	2011
June	2012
July	2013
August	2014

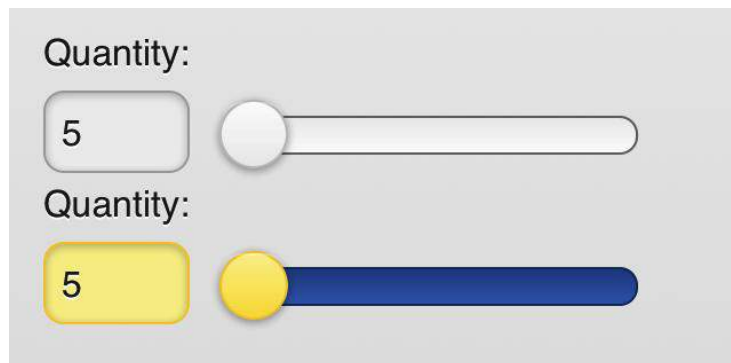
5.5.2 表单元素

7、滑块

滑块用于输入处于某个范围的数字的值。它会提供一个接受数字的文本输入框，右侧会有一个水平滑块。滑块是一个`<input type="range">`的HTML5控件，它接受`min`、`max`以及`step`属性值。

可以使用`data-theme`和`data-track-theme`属性本别控制滑块左右两个部分的色样。例如：

```
<div data-role="fieldcontainer">
  <label for="qty">Quantity:</label>
  <input type="range" id="qty" name="qty" min="1" max="100"
    value="5" data-theme="e" data-track-theme="b">
</div>
```



5.5.2 表单元素

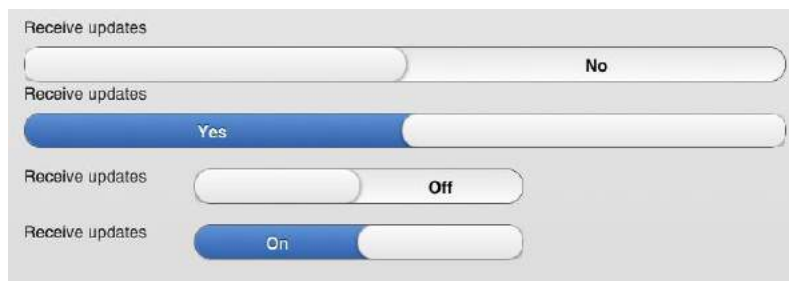
8、轻弹切换开关

这种开关是一个布尔值（**true**或**false**）选择器。功能上与复选框类似，但用户界面完全不同。

它是第一个需要制定角色的表单控件：**data-role="slider"**。它是一个**select**元素，其中只包含两个**option**子元素，第一个值为**false**，第二个值为**true**。

不指定域容器的话，开关将占满整个页面宽度。更常见的情况是将它放入一个域容器中。例如：

```
<label for="updated">Receive updates</label>
<select id="updated" name="updated" data-role="slider">
  <option value="no">No</option>
  <option value="yes">Yes</option>
</select>
```



5.5.2 表单元素

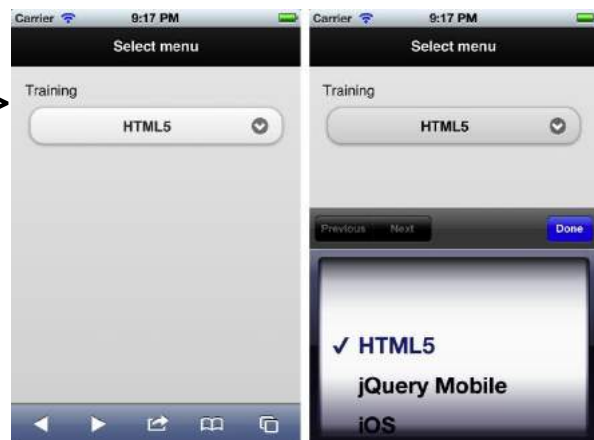
9、选择菜单

选择菜单由**select**元素创建，用于从一个弹出列表选择一个或多个选项。**jQuery Mobile**将选择菜单的外观改变为按钮样式的风格，右边带有一个提示用户可以打开选项菜单的图标。点击时，将激活一个本地菜单供选择。

默认情况下，选择菜单会占用整个宽度，除非将它包含在一个容器中。

见例：5-5-1.html

```
<label for="training">Training</label>
<select id="training" name="training">
  <option value="1">HTML5</option>
  <option value="2">jQuery Mobile</option>
  <option value="3">iOS</option>
  <option value="4">Android</option>
  <option value="5">BlackBerry</option>
  <option value="6">Qt for Meego</option>
</select>
```



5.5.2 表单元素

当使用**multiple**布尔值属性后，界面会显示使用逗号分隔的已选中元素（初始空白）以及一个显示已选中多少元素的计数气泡。

见例：5-5-2.html

```
<label for="lang">Languages you like</label>
<select id="lang" name="lang" multiple>
  <option value="1">C/C++</option>
  <option value="2">Objective-C</option>
  <option value="3">Java</option>
  <option value="4">C#</option>
  <option value="5">Visual Basic</option>
  <option value="6">ActionScript</option>
  <option value="7">Delphi</option>
  <option value="8">Phyton</option>
  <option value="9">JavaScript</option>
  <option value="10">Ruby</option>
  <option value="11">PHP</option>
</select>
```

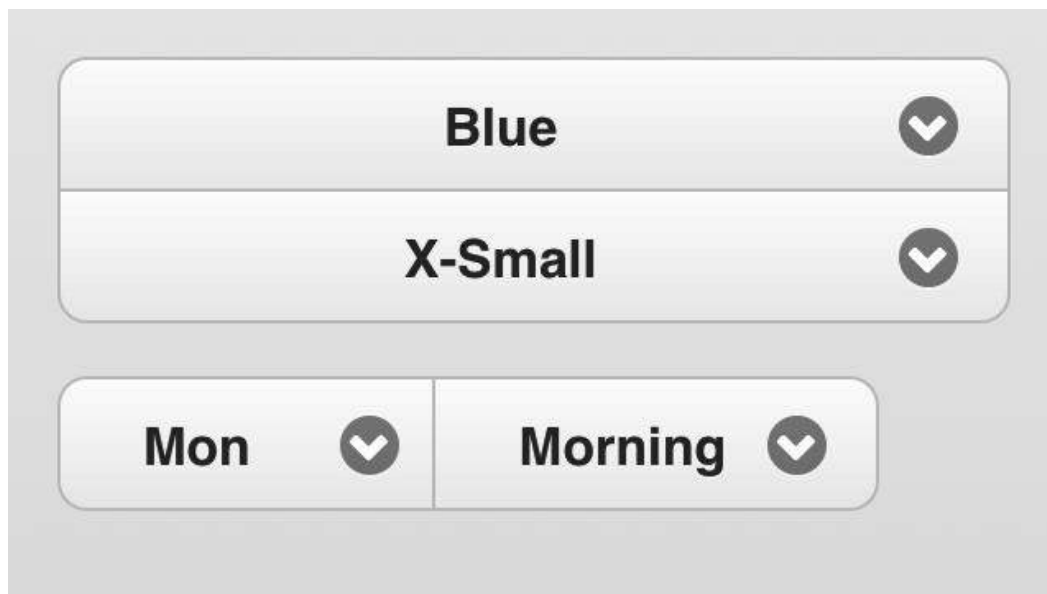


5.5.2 表单元素

① 选择菜单的组合

多个选项菜单可以使用具有`data-role="controlgroup"`属性的元素进行垂直或水平组合。组合后，选项菜单各自的标签将被隐藏，可以使用`legend`元素为整个组合定义一个公共的标签。

见例：5-5-3.html



The image displays a user interface with a light gray background. It features two types of select menu combinations. The first is a vertical combination where two select menus are stacked. The top menu shows 'Blue' and the bottom one shows 'X-Small'. The second is a horizontal combination where two select menus are placed side-by-side. The left menu shows 'Mon' and the right one shows 'Morning'. Each menu has a dark gray arrow icon on its right side, indicating it is a dropdown menu.

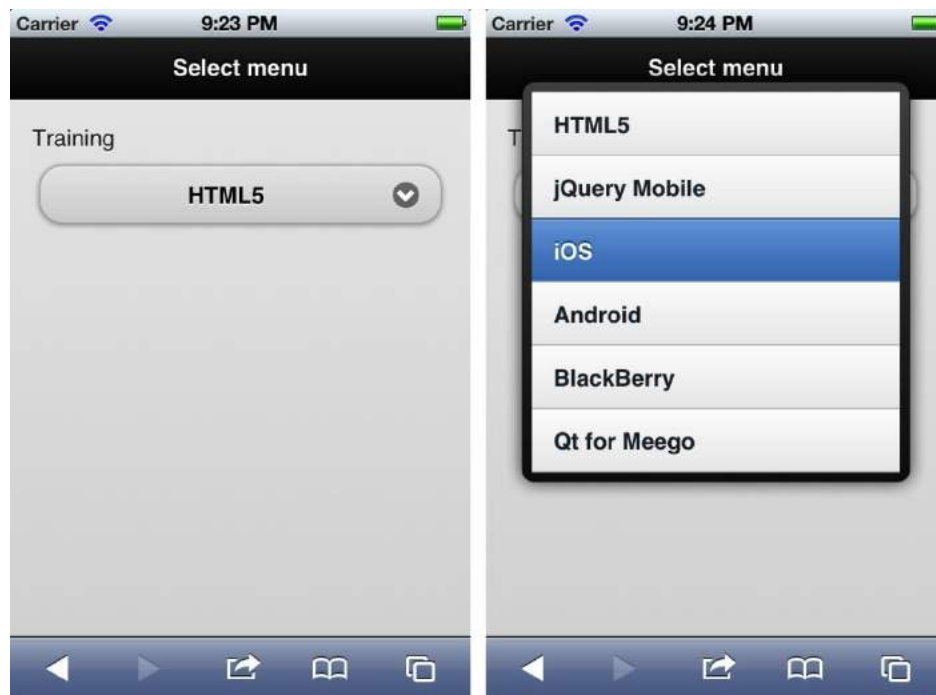
5.5.2 表单元素

② 非本地选择菜单

当点击选项菜单时，将激活一个本地菜单供选择。这个行为可以通过 `data-native-menu="false"` 属性来改变。

改变后，将弹出一个列表供选择。当列表太长时，它将使用一个类似对话框的外观。

见例：5-5-4.html



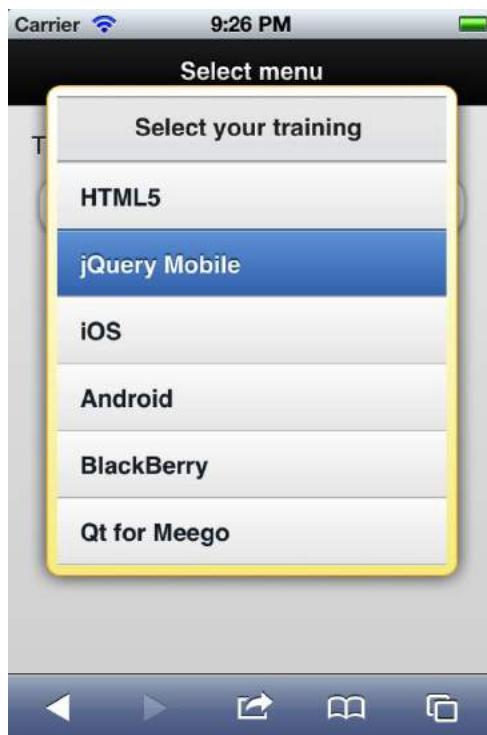
5.5.2 表单元素

打开非本地选择菜单时，当前表单上出现一个覆盖层。

可以通过`data-overlay-theme`来指定这个覆盖层的色样。

如果某个`option`元素显式地指定了空值（`value=""`）或带有属性`data-placeholder="true"`，则它将作为覆盖层的标题，而不再是一个可选项。

见例：5-5-5.html



5.5.2 表单元素

非本地选择菜单也支持多选，只需添加属性**multiple**即可。这时覆盖层对话框会有一个关闭按钮，并且不会在选择了一项之后就自动关闭。如下图：



5.5.2 表单元素

10、单选按钮

在jQuery Mobile中使用单选按钮有几个条件：

- 每个选项必须是一个<input type="radio">。
- 同一组每个选项的name属性值必须相同。
- 每个选项必须具有唯一的id和一个关联的label标签。

单选按钮与它的label标签被显示在一个按钮里。可以使用一个legend元素来为整个选项组提供一个文本标签。

见例：5-5-6.html



Menu type

- ☐ Native menu, single selection
- ☐ Native menu, multiple selection
- ☒ Non-native menu, single selection
- ☐ Non-native menu, multiple selection

5.5.2 表单元素

如果将所有radio元素都包含在一个controlgroup容器中，用户界面会更友好一些。

见例：5-5-7.html

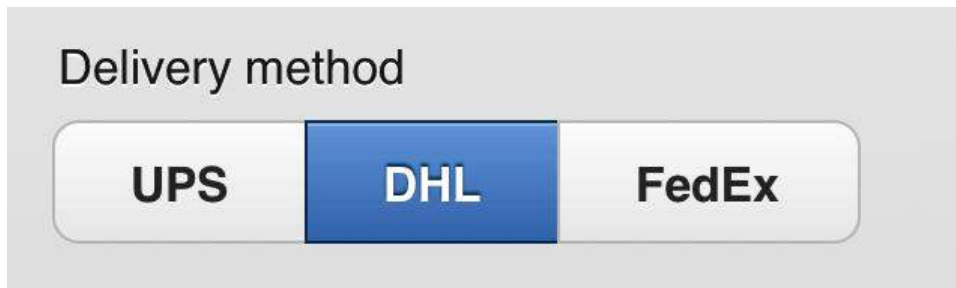


Menu type

- ☐ Native menu, single selection
- ☒ Native menu, multiple selection
- ☐ Non-native menu, single selection
- ☐ Non-native menu, multiple selection

如果将controlgroup容器的type改为horizontal，外观将大不一样。元素将不再是典型的单选按钮，而是变成了单选开关按钮。

见例：5-5-8.html



Delivery method

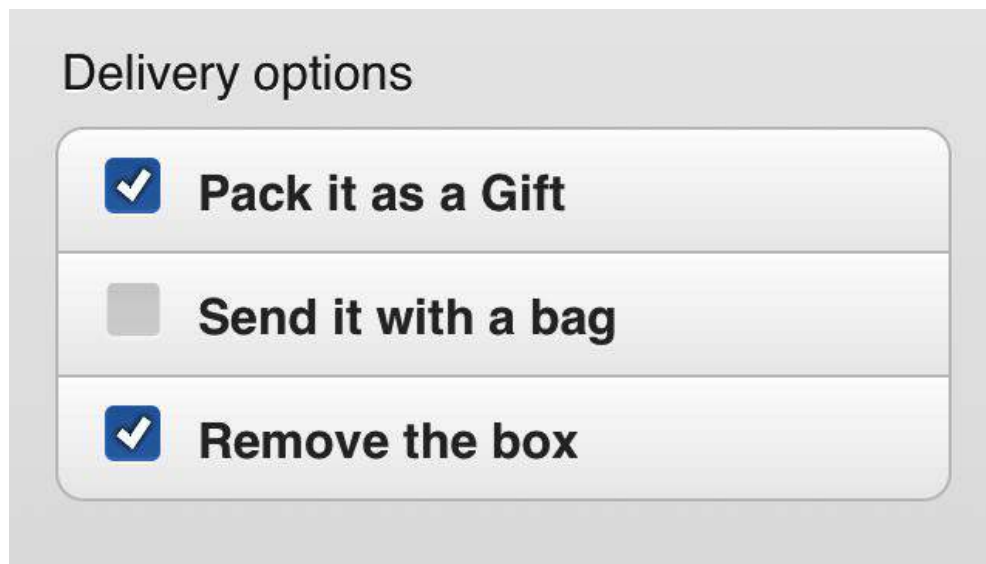
UPS DHL FedEx

5.5.2 表单元素

11、复选框

复选框与单选按钮工作方式类似，不过允许复选。

见例5-5-9.html:



Delivery options

- ☒ Pack it as a Gift
- ☐ Send it with a bag
- ☒ Remove the box

Radio元素和Checkbox元素都属于切换按钮。一组Radio元素总是具有相同的name属性值，以便具有互斥功能；通常也将Checkbox元素进行分组，同组各元素具有相同的name属性值。这些同名元素被放在一个元素数组中，数组名即是name属性的值。



5.6 jQuery Mobile API

jQuery Mobile提供了使用JavaScript与框架通讯以及进行内容管理的API。它们不仅可以用于创建与框架兼容的动态内容，而且提供了新的事件以及可以自行设置的全局配置。

5.6.1 文档事件

操作jQuery Mobile文档时，首先需要理解和处理一个新的事件：**mobileinit**。这个事件会在jQuery Mobile框架载入之后，准备执行我们的初始化代码之前触发。

它需要放在jQuery核心文件之后，jQuery Mobile核心文件之前处理。例如：

```
<script src="jquery-1.6.4.min.js"></script>
<script>
$(document).bind("mobileinit", function() {
    // Our initialization code here
});
</script>
<script src="jquery.mobile-1.0.min.js"></script>
```



5.6.2 配置

jQuery Mobile在jQuery主对象\$(或者jQuery)上添加了一个新的mobile对象，jQuery Mobile API的大部分工作都是通过\$.mobile(或者jQuery.mobile)来完成的。这个对象只有在mobileinit事件触发后才能使用。

jQuery Mobile的界面主要由框架管理的组件(widget)构成。组件通常由data-role属性标识，但也有不带此属性的表单控件。

每种组件其实都是一个JavaScript类型，都有一个构造函数及默认配置，这些默认配置可以在mobileinit中更改，更改后会影响到页面上的每个组件实例。

下面是jQuery Mobile 1.0中可用组件的名称：

page、dialog、collapsible、fieldcontain、navbar、listview、checkboxradio、button、slider、textinput、selectmenu、controlgroup

其中有些构造函数可以构造不止一种界面元素。例如，所有文本输入类型（包括textarea）都使用一个组件textinput，复选框和单选按钮都使用checkboxradio组件。



5.6.2 配置

组件的默认配置保存在构造函数的原型(**prototype**)属性中, 可以通过 `$.mobile.<widget_name>.prototype` 来访问。通常, 每个组件原型都有一个 **options** 对象, 通过这个对象可以访问组件的默认属性。例如, `$.mobile.page.prototype.options` 可用于设置应用到每个页面实例(`data-role="page"`)的默认属性。

1、全局配置

指全局性属性配置。全局属性与组件无关, 不需要使用原型。

① 用户界面

缺省情况下, **jQuery Mobile** 会为页面上的活动和非活动元素分配不同的类名 (即 **class** 属性) 以便通过 **CSS** 来为它们提供不同的外观。通过设置字符串属性 **activePageClass** 和 **activeBtnClass** 可以改变当前活动页面和活动按钮的类名, 它们的缺省值分别为 **ui-page-active** 和 **ui-btn-active**。很多其它组件也会使用活动按钮状态, 例如在导航条、单选和复选按钮的内部。

另两个经常改动的全局属性是页面或对话框的默认加载过渡效果, 其缺省值分别是 **slide** 和 **pop**, 可以通过 **defaultPageTransition** 和 **defaultDialogTransition** 属性来改变。



5.6.2 配置

这些全局属性都可以在mobileinit中进行改变。例如：

```
$(document).bind("mobileinit", function() {  
    $.mobile.defaultPageTransition = "fade";  
    $.mobile.activeBtnClass = "active-button";  
});
```



5.6.2 配置

② 核心及AJAX功能

一些核心功能可以通过全局属性来操作。

如果同时在使用另一个可能会与jQuery Mobile冲突的框架，可以使用ns全局属性来定义一个名字空间（默认情况下，这一属性为空）。例如：

```
$(document).bind("mobileinit", function() {  
    $.mobile.ns = "firt";  
});
```

以后所有的jQuery Mobile自定义属性data-*都变成了data-
<namespace>-*。对上例，data-role属性将变为data-firt-role。
在新的名字空间下，一个典型的页面变为：

```
<div data-firt-role="page">  
    <div data-firt-role="header" data-firt-theme="a">  
        </div>  
    <div data-firt-role="content">  
        </div>  
</div>
```

这时，你还必须手工修改所有的CSS文件（包括结构和主题）。



5.6.2 配置

缺省情况下jQuery Mobile使用AJAX加载外部页面，这一功能可以通过使用`$.mobile.ajaxEnabled=false`禁止。当然通常不会这么做！

但通常情况下，AJAX不支持跨域加载。可以尝试使用`allowCrossDomainPages`属性强制框架支持。

在通过AJAX加载页面时，有一些默认属性可以通过`$.mobile.loadPage.defaults`对象进行设置。例如：

属性	可接受值	默认值	描述
type	"get"/"post"	"get"/"post"	请求类型
data	object/string		Type为"post"时，设置发送数据。
reloadPage	true/false	false	是否重新加载已有页面
role	string	data-role的值	目标页面角色
showLoadMsg	true/false	true	超时显示加载信息？
loadMsgDelay	微秒数	50	显示加载信息前等待多少毫秒
theme	a到z	c	默认色样
domCache	true/false	false	是否将页面缓存在DOM中



5.6.2 配置

③ 本地化字符串

jQuery Mobile有一些用于提示的本地化字符串。下面是它们的缺省值：

```
// Global strings
```

```
$.mobile.loadingMessage = "loading";
```

```
$.mobile.pageLoadErrorMessage = "Error Loading Page";
```

```
// Widget strings
```

```
$.mobile.page.prototype.options.backBtnText = "Back";
```

```
$.mobile.dialog.prototype.options.closeBtnText = "Close"
```

```
$.mobile.collapsible.prototype.options.expandCueText =  
"click to expand contents";
```

```
$.mobile.collapsible.prototype.options.collapseCueText = "click to  
collapse contents";
```

```
$.mobile.listview.prototype.options.filterPlaceholder = "Filter  
items...";
```

```
$.mobile.selectmenu.prototype.options.closeText = "Close";
```

这些都可以在mobileinit中进行改变，例如：

```
$.mobile.loadingMessage = "正在加载";
```



5.6.2 配置

2、页面配置

页面由**data-role="page"**定义。缺省情况下，页面使用默认属性。如果想改变默认属性，可以在页面中使用**data-***属性来设置。

这里所说的是改变所有页面使用的默认属性。这要通过页面原型的**options**对象来改变。例如下例为页面增加了后退按钮并设置了默认主题。

```
$(document).bind('mobileinit', function() {  
    $.mobile.page.prototype.options.addBackBtn = true;  
    $.mobile.page.prototype.options.backBtnTheme = "e";  
    $.mobile.page.prototype.options.headerTheme = "b";  
    $.mobile.page.prototype.options.footerTheme = "d";  
});
```



5.6.2 配置

3、组件配置

jQuery Mobile中的每个组件都有自己的默认配置属性。可以通过组件的prototype属性中的options对象来改变这些默认属性。例如：

```
$(document).bind('mobileinit', function() {  
    $.mobile.listview.prototype.options.filter = true;  
    $.mobile.selectmenu.prototype.options.nativeMenu=false;  
    $.mobile.selectmenu.prototype.options.theme="e";  
});
```

其实页面也是一种组件。每种组件都有哪些默认配置属性可以查阅jQuery Mobile文档。



5.6.3 实用工具

jQuery Mobile提供了许多使用JavaScript来管理应用的实用工具，这些工具通过方法及只读属性提供，使jQuery Mobile程序设计更加方便。

1、操作自定义属性

使用jQuery Mobile时，经常需要处理各种data-*自定义属性。例如：

```
var buttons = $("a[data-role='button']");
```

jQuery Mobile添加了一个新的过滤器jqmData，它能够处理名字空间，因此使用起来更安全。例如，下面的语句与上面的等效但更安全：

```
var buttons = $("a:jqmData(role='button')");
```

同理，在jQuery集合上使用jqmData和jqmRemoveData比使用jQuery函数data和removeData更加安全。例如：

```
$("#button1").jqmData("theme", "a");  
$("a").jqmRemoveData("transition");
```



5.6.3 实用工具

2、页面工具

当需要访问当前页面时，可以使用`$.mobile.activePage`属性，该属性自动与当前可见的`data-role="page"`元素关联，是一个JavaScript对象，通常是一个div元素。例如：

```
var currentPageId = $.mobile.activePage.id;
```

另一个属性：`$.mobile.pageContainer`则代表当前页面的容器，通常是body元素。

框架中最有用的工具应该是`$.mobile.changePage`方法，它允许我们转向另一个页面，就像点击了响应的链接一样。

该方法的第一个参数是必选的：可以是一个代表外部页面的字符串，也可以是指向内部页面的jQuery对象。例如：

```
$.mobile.changePage("external.html");
```

```
$.mobile.changePage($("#pageId"));
```

注意：不能通过字符串来加载内部页面，必须使用jQuery对象。这就是上例中使用`$("#pageId")`的原因。



5.6.3 实用工具

`changePage`的第二个参数是可选的，是一个对象，通常为JSON格式，用于定义页面过渡和AJAX加载的可选项。例如：

```
$.mobile.changePage($("#page2"), {  
    transition: "slide",  
    reverse: true  
});
```

页面过渡和AJAX加载的可选项有很多，具体可以查阅文档。

还有一个`$.mobile.loadPage`方法，它只将页面加载到DOM，但并不会转向它。需要使用`$.mobile.changePage`来显示它。



5.6.4 动态内容

1、创建页面

/* 见例：5-6-1.html*/

本例使用的方法有：

- `$()`：jQuery函数。当参数为THML标记时，创建只有一个DOM元素的jQuery对象；当参数为选择器时，创建选择结果的jQuery对象。
例如：`$("<div>");` //创建一个具有一个div元素的jQuery对象。
`$("body");` //创建一个包含所有body元素的jQuery对象。
- `.jqmData(属性名, 属性值)`：为元素添加自定义属性，jQuery Mobile函数。
例如：`.jqmData("role", "page");` //为元素添加属性data-role="page"。
- `.attr(属性名, 属性值)`：为元素添加属性，jQuery函数。
例如：`attr("id", "page0");` //为元素添加id="page0"属性。
- `.append(jQuery对象)`：将参数对象添加到父亲的末尾，jQuery函数。
`.append("$("<p>");`为父亲追加一个段落对象。
- `.html(HTML字符串)`：参数字符串作为匹配元素的内容。
例如：`$("<h1>").html("Page0");` 创建只有一个h1元素的jQuery对象，其文本为page0。



5.6.4 动态内容

创建页面的最佳方法是先创建一个可分辨的链接，然后捕获它的 `pagebeforechange` 事件，在响应函数中修改框架中页面的内容并显示。

/* 见例：5-6-2.html */

本例中需要解释的有：

- `pagebeforechange` 事件的响应函数可以有两个参数，第二个参数中包含各种关于启动和目标页面的信息。
- `$.mobile.path`：jQuery Mobile 的路径工具属性。
- `dataUrl`：`changePage()` 操作的可选属性，表示写入地址的 URL。
- `event.preventDefault()`：阻止缺省行为的发生。



5.6.4 动态内容

2、创建组件

创建动态组件需要使用组件的构造函数。只要将jQuery对象传递给组件的构造函数即可。例如：

```
$("#list1").listview();  
$("#a").button();  
/* 见例： 5-6-3.html*/
```

3、更新组件

对已经创建并渲染的组件，在进行了更新后，需要使用“refresh”字符串调用组件的构造函数进行刷新。例如：

```
$("#list1").listview('refresh');  
$("#checkbox").val('true').checkboxradio('refresh');  
/* 见例： 5-6-4.html*/
```



5.6.5 改变页面内容

如果改变了包含多个组件的一大块HTML，就需要刷新整个容器。

要刷新一个容器，只需要在页面上触发create事件。这时每个组件就会检查是否需要创建新的实例。例如：

```
$("#content").html(newHTMLcontentWithWidgets);
```

```
$("#page1").trigger("create");
```

其中：`.trigger()`方法用来模拟发生各种事件，包括create事件。页面将根据新的内容重新创建。



5.6.6 处理事件

jQuery Mobile提供了一些新的事件。这些事件可以使用**bind**或**live**等典型的jQuery方法来处理。

bind方法用于将事件绑定到已有的DOM对象；**live**方法将事件委托给**document**对象进行处理，是一个事件委托方法。另外**live**方法还可以被用来处理将要创建的DOM对象。

1、手势事件

手势事件主要包括：**swipe**、**swipeleft**、**swiperight**、**tap**、**taphold**

- ① **swipe**: 一秒钟以内，水平滑动30像素以上（要求垂直移动不超过75像素）。其中的参数可以设置。
- ② **swipeleft**: 一秒钟以内，向左水平滑动30像素以上（要求垂直移动不超过75像素）。
- ③ **swiperight**: 一秒钟以内，向右水平滑动30像素以上（要求垂直移动不超过75像素）。
- ④ **tap**: 在屏幕上快速地触摸一下。
- ⑤ **taphold**: 在屏幕上触摸并保持一秒钟时触发。

/* 见例：5-6-5.html */



5.6.6 处理事件

2、页面事件

要处理页面事件，可以调用`$(document).bind`，也可以更准确地调用`$(":jqmData(role='page']").bind`。或者用`live`代替`bind`，以便可以绑定那些将来要添加到DOM中的页面。

① 创建事件（包括创建和初始化事件）：

- `pagebeforecreate`：页面已插入DOM，但组件尚未创建。
- `pagecreate`：页面（包括其中的组件）已被创建，但尚未被渲染。
- `pageinit`：页面已完全加载。
- `pageremove`：框架打算从DOM中删除一个外部页面前。

例如：

```
$("#page2").live("pageinit", function(event) {  
  
});
```



5.6.6 处理事件

② 加载事件

用于使用AJAX加载的页面：

- **pagebeforeload**: 就在AJAX加载页面前。
- **pageload**: 页面已加载。
- **pageloadfailed**: 页面加载失败时。

③ 显示事件

用于处理页面的显示或隐藏事件：

- **pagebeforechange**: 页面改变及过渡之前。
- **pagechange**: 页面刚刚改变。
- **Pagechangefailed**: 页面改变失败时。
- **pagebeforeshow**: 在页面以过渡方式显示之前。
- **Pageshow**: 页面刚刚以过渡方式在屏幕上显示。
- **Pagebeforehide**: 页面即将隐藏前。
- **Pagehide**: 页面刚刚完成卸载及隐藏。



5.6.6 处理事件

3、组件事件

每个能动态显示或隐藏内容的组件，例如**collapsible**等，都会触发一个**updatelayout**事件，表示页面布局已经发生了变化。有时可能需要监听这个事件，以便更新**UI**上的其它内容。

4、方向事件

移动设备可以旋转，至少会有两个不同的方向。**jQuery Mobile**提供了一个**orientationchange**事件，可以附加在**document**上。例如：

```
$(document).bind("orientationchange", function(orientation) {  
    if (orientation=="landscape") {  
        // We are now in landscape  
    } else {  
        // We are now in portrait  
    }  
});  
});
```



5.6.6 处理事件

5、虚拟点击事件

在大多数移动浏览器中，使用点击事件（例如：**click**、**mouseover**）时都有**300-500**毫秒的延迟，但在使用触摸事件（例如：**touchstart**、**touchmove**）时就没有延迟。而且并非所有触摸浏览器都支持触摸事件。为了在移动设备中规范化点击和触摸事件，**jQuery Mobile**提供了虚拟点击事件。

虚拟点击事件是一个包装，可用于取代触摸或点击事件，它们将针对不同的平台进行正确的选择。它还针对单点触摸标准化了位置信息。

虚拟点击事件的用法与点击事件完全相同，只不过名称中多了一个前缀**v**。**jQuery Mobile**中包括以下虚拟点击事件：

vclick, **mouseover**, **mousedown**, **mousemove**, **mouseup**, **mouseleave**。



5.7 创建主题

jQuery Mobile允许开发者通过主题和CSS来定制整个用户界面。

主题是一组配色方案（也称为色样），包括：

- 文本颜色。
- 背景颜色及变化。
- 字体。

每个主题可以定义多达**26**个配色方案，用字母**a**到**z**表示。通常至少会定义**5**个配色方案。

每个主题还包含一个全局定义，适用于主题的所有配色方案。包括：

- 文本效果和盒效果，例如阴影和圆角。
- 按钮和其它控件的激活状态。

全局定义可以确保无论主题套用哪个色样，都可以保持一致的用户体验。比如，在默认主题中，无论按钮使用什么色样，它的选中状态始终显示为蓝色。



5.7 创建主题

主题保存在一个**CSS**文件中。

使用默认主题时，只需要在**HTML**文档中包含框架提供的核心**CSS**文件。

使用自定义主题时，不仅需要包含自定义的主题**CSS**文件，而且需要包含系统提供的结构**CSS**文件。

主题**CSS**文件主要包含颜色和形状样式；结构**CSS**主要描述文档结构，例如元素大小和位置样式。

如果对系统提供的结构**CSS**文件不满意，也不要更改它。建议另写一个**CSS**文件，然后在载入系统结构**CSS**文件后再引入它，从而实现样式的覆盖。

jQuery Mobile提供了免费在线的图形化主题编辑器ThemeRoller:

<http://jquerymobile.com/themeroller>

另外新版本的Adobe Fireworks也包含一个jQuery Mobile主题编辑器。



5.7 创建主题

jQuery Mobile使用类来定义样式。HTML标记中的组件属性都会被转换成若干CSS类名。主题CSS文件就是在为这些类定义样式。

注意：在页面内容区域，可以使用任何想要的HTML代码，因此也能使用任何不同于框架CSS的自定义CSS样式。

在主题CSS文件中，每个类名都有一个前缀ui，后缀是所套用的色样字母，类似于：ui-**<元素类型>**-**<色样>**。例如：对按钮来说，套用色样a的按钮元素的类名就是ui-btn-a，套用色样c的按钮类名就是ui-btn-c。



5.7 创建主题

主题文件中包含的，可以修改，从而定制用户界面的类包括：

类名	描述
ui-bar-<x>	页头、页脚及其它条状栏
ui-btn-up-<x>	常态下的按钮
ui-btn-hover-<x>	选停下的按钮
ui-btn-down-<x>	按下状态的按钮
ui-btn-active	激活状态下的按钮，适用于所有色样
ui-body-<x>	整个页面
ui-link-<x>	链接
ui-icon-<x>	按钮和其它组件中用到的图标
ui-corner-all	适用于所有带圆角的控件
ui-corner-<tl/tr/bl/br>	适用于左上/右上/左下/右下的圆角
ui-corner-<top/bottom>	适用于顶部/底部的圆角
ui-corner-<left/right>	适用于左边/右边的圆角
ui-shadow	适用于所有有阴影的元素
ui-disabled	适用于所有禁用（无法交互的）元素