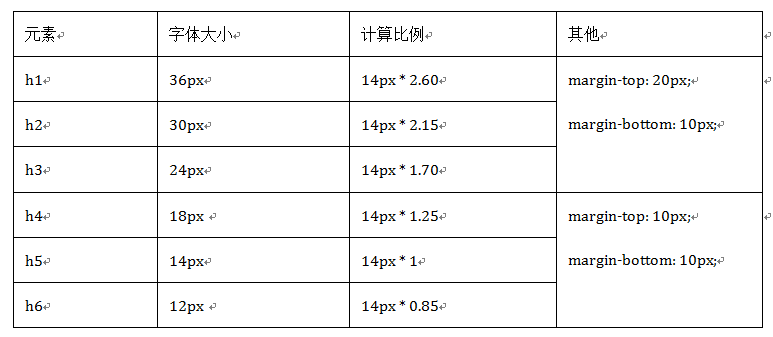
Bootstrap

标题（一）

Bootstrap和普通的HTML页面一样，定义标题都是使用标签<h1>到<h6>,只不过Bootstrap覆盖了其默认的样式，使用其在所有浏览器下显示的效果一样，具体定义的规则可以如下表所示：

[](http://img.mukewang.com/53acce330001429807730337.jpg)

通过比较可以发现，Bootstrap标题样式进行了以下显著的优化重置：

1、重新设置了**margin-top**和**margin-bottom**的值，  **h1~h3**重置后的值都是**20px**；**h4~h6**重置后的值都是**10px。**  
2、所有标题的行高都是**1.1**（也就是font-size的1.1倍）,而且文本颜色和字体都继承父元素的颜色和字体。  
3、固定不同级别标题字体大小，**h1=36px，h2=30px，h3=24px，h4=18px，h5=14px**和**h6=12px。**

标题（二）

除此之外，我们在Web的制作中，常常会碰到在一个标题后面紧跟着一行小的副标题。在Bootstrap中他也考虑了这种排版效果，使用了**<small>**标签来制作副标题。这个副标题具有其自己的一些独特样式：

**1、**行高都是**1**，而且**font-weight**设置了**normal**变成了常规效果（不加粗），同时颜色被设置为**灰色（#999）。**  
**2、**由于**<small>**内的文本字体在**h1~h3**内，其大小都设置为当前字号的**65%；**而在**h4~h6**内的字号都设置为当前字号的**75%；**  
详细代码请参阅**bootstrap.css**文件中第**407行~第443**行。

h1 small,

.h1 small,

h2 small,

.h2 small,

h3 small,

.h3 small,

h1 .small,

.h1 .small,

h2 .small,

.h2 .small,

h3 .small,

.h3 .small {

font-size: 65%;

}

h4,

.h4,

h5,

.h5,

h6,

.h6 {

margin-top: 10px;

margin-bottom: 10px;

}

h4 small,

.h4 small,

h5 small,

.h5 small,

h6 small,

.h6 small,

h4 .small,

.h4 .small,

h5 .small,

.h5 .small,

h6 .small,

.h6 .small {

font-size: 75%;

}

## 段落（正文文本）

段落是排版中另一个重要元素之一。在Bootstrap中为文本设置了一个全局的文本样式（这里所说的文本是指正文文本）：

1、全局文本字号为**14px(font-size)**。

2、行高为**1.42857143（line-height）**，大约是**20px**(大家看到一串的小数或许会有疑惑，其实他是通过LESS编译器计算出来的，当然Sass也有这样的功能)。

3、颜色为**深灰色（#333）**；

4、字体为**"Helvetica Neue", Helvetica, Arial, sans-serif;（font-family）**，或许这样的字体对我们中文并不太合适，但在实际项目中，大家可以根据自己的需求进行重置，在此我们不做过多阐述，我们回到这里。该设置都定义在**<body>**元素上，由于这几个属性都是**继承属性**，所以Web页面中文本（包括段落p元素）如无重置都会具有这些样式效果。

/\*源码请查看bootstrap.css文件中第274行~280行\*/

body {

font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;

font-size: 14px;

line-height: 1.42857143;

color: #333;

background-color: #fff;

}

另外在Bootstrap中，为了让段落**p**元素之间具有一定的间距，便于用户阅读文本，特意设置了**p**元素的**margin**值（默认情况之下，**p**元素具有一个上下外边距，并且保持一个行高的高度）：  
/\*源码请查看bootstrap.css文件中第467行~469行\*/

p {

margin: 0 0 10px;

}

如果你对CSS预处理器有所了解，那么你完全可以根据Bootstrap提供的预编译版本LESS(或者Sass)进行自定义排版设置。在Bootstrap中，排版设置的默认值都存在**variables.less**文件中(Sass版本存在**\_variables.scss中**)的两个变量：

**LESS版本：**

@font-size-base: 14px; @line-height-base: 1.428571429; // 20/14

**Sass版本：**

$font-size-base: 14px !default; $line-height-base: 1.428571429 !default; // 20/14

第一条语句用于设置**字体大小**，第二条语句用于设置**行高**。系统默认使用这两个值产生整个页面相应的**margin、padding**和**line-height**的值。换句话说，你只需要修改这两个变量的值，然后重新编译，就可以自定义自己的Bootstrap排版样式。（有兴趣的同学可以尝试一下，此处对于LESS或Sass版本运用不做过多阐述）。

## 强调内容

在实际项目中，对于一些重要的文本，希望突出强调的部分都会做另外的样式处理。Bootstrap同样对这部分做了一些轻量级的处理。

如果想让一个段落**p**突出显示，可以通过添加类名“**.lead**”实现，其作用就是增大文本字号，加粗文本，而且对行高和**margin**也做相应的处理。用法如下：

<p>我是普通文本，我的样子长成这样我是普通文本，我的样子长成这样我是普通文本，</p>

<p **class="lead"**>我是特意要突出的文本，我的样子成这样。我是特意要突出的文本，我的样子长成这样。</p>

效果查看最右侧结果窗口。

**“.lead”对应的样式如下：**  
/\*源码查看bootstrap.css文件第470行~480行\*/

.lead {

margin-bottom: 20px;

font-size: 16px;

font-weight: 200;

line-height: 1.4;

}

@media (min-width: 768px) {/\*大中型浏览器字体稍大\*/

.lead {

font-size: 21px;

}

}

除此之外，Bootstrap还通过元素标签:**<small>**、**<strong>**、**<em>**和**<cite>**给文本做突出样式处理。  
/\*源码查看bootstrap.css文件第55行~第58行\*/

b,strong {

font-weight: bold; /\*文本加粗\*/

}

/\*源码查看bootstrap.css文件第481行~第484行\*/

small,.small {

font-size: 85%; /\*标准字体的85%,也就是14px \* 0.85px，差不多12px\*/

}

/\*源码查看bootstrap.css第485行~第487行\*/

cite {

font-style: normal;

}

## 粗体

粗体就是给文本加粗，在普通的元素中我们一般通过font-weight设置为bold关键词给文本加粗。在Bootstrap中，可以使用**<b>**和**<strong>**标签让文本直接加粗。  
/\*源码查看bootstrap.css文件第55行~第58行\*/

b,strong {

font-weight: bold; /\*文本加粗\*/

}

例如，下面的代码使用**<strong>**标签定义了强调文本：

<p>我在学习**<strong>**Bootstrap**</strong>**，我要掌握**<strong>**Bootstrap**</strong>**的所有知识。</p>

## 斜体

在排版中，除了用加粗来强调突出的文本之外，还可以使用斜体。斜体类似于加粗一样，除了可以给元素设置样式**font-style**值为**italic**实现之外，在Bootstrap中还可以通过使用标签**<em>**或**<i>**来实现。  
例如，下面的代码使用了**<em>**和**<i>**标签定义了强调文本：

<p>我在慕课网上跟**<em>**大漠**</em>**一起学习**<i>**Bootstrap**</i>**的使用。我一定要学会**<i>**Bootstrap**</i>**。</p>

## 强调相关的类

在Bootstrap中除了使用标签**<strong>、<em>**等说明正文某些字词、句子的重要性，Bootstrap还定义了一套类名，这里称其为强调类名（类似前面说的“**.lead**”）,这些强调类都是通过颜色来表示强调，具本说明如下：

* .text-muted：提示，使用浅**灰色（#999）**
* .text-primary：主要，使用**蓝色（#428bca）**
* .text-success：成功，使用**浅绿色(#3c763d)**
* .text-info：通知信息，使用**浅蓝色（#31708f）**
* .text-warning：警告，使用**黄色（#8a6d3b）**
* .text-danger：危险，使用**褐色（##a94442）**

具本源码查看bootstrap.css文件第500行~第532行：

.text-muted {

color: #999;

}

.text-primary {

color: #428bca;

}

a.text-primary:hover {

color: #3071a9;

}

.text-success {

color: #3c763d;

}

a.text-success:hover {

color: #2b542c;

}

.text-info {

color: #31708f;

}

a.text-info:hover {

color: #245269;

}

.text-warning {

color: #8a6d3b;

}

a.text-warning:hover {

color: #66512c;

}

.text-danger {

color: #a94442;

}

a.text-danger:hover {

color: #843534;

}

## 文本对齐风格

在排版中离不开文本的对齐方式。在CSS中常常使用text-align来实现文本的对齐风格的设置。其中主要有四种风格：

  ☑  左对齐，取值left

  ☑  居中对齐，取值center

  ☑  右对齐，取值right

  ☑  两端对齐，取值justify

为了简化操作，方便使用，Bootstrap通过定义四个类名来控制文本的对齐风格：

  ☑   .text-left：左对齐

  ☑   .text-center：居中对齐

  ☑   .text-right：右对齐

  ☑   .text-justify：两端对齐

具体源码查看bootstrap.css文件第488行~第499行：

.text-left {

text-align: left;

}

.text-right {

text-align: right;

}

.text-center {

text-align: center;

}

.text-justify {

text-align: justify;

}

例如下面的四行代码，分别定义文本的四种不同的对齐风格：

<p class="text-left">我居左</p>

<p class="text-center">我居中</p>

<p class="text-right">我居右</p>

<p class="text-justify">我两端对齐</p>

特别声明：目前两端对齐在各浏览器下解析各有不同，特别是应用于中文文本的时候。所以项目中慎用。

## 列表--简介

在HTML文档中，列表结构主要有三种：有序列表、无序列表和定义列表。具体使用的标签说明如下：  
**无序列表**

<ul>

<li>…</li>

</ul>

**有序列表**

<ol>

<li>…</li>

</ol>

**定义列表**

<dl>

<dt>…</dt>

<dd>…</dd>

</dl>

Bootstrap根据平时的使用情形提供了**六种**形式的列表：

   ☑  普通列表

   ☑  有序列表

   ☑  去点列表

   ☑  内联列表

   ☑  描述列表

   ☑  水平描述列表

接下来，我们一起来学习Bootstrap提供的这六种列表的使用方法。

## 列表--去点列表

小伙伴们可以看到，在Bootstrap中默认情况下**无序列表**和**有序列表**是带有项目符号的，但在实际工作中很多时候，我们的列表是不需要这个编号的，比如说用无序列表做导航的时候。Bootstrap为众多开发者考虑的非常周道，通过给无序列表添加一个类名“**.list-unstyled**”,这样就可以去除默认的列表样式的风格。  
/\*源码请查看bootstrap.css文件第580行~第583行\*/

.list-unstyled {

padding-left: 0;

list-style: none;

}

从示例中可以看出，除了项目编号之外，还将列表默认的**左边内距**也**清０**了。

## 列表--内联列表

Bootstrap像去点列表一样，通过添加类名“**.list-inline**”来实现内联列表，简单点说就是**把垂直列表换成水平列表**，**而且去掉项目符号（编号）**，**保持水平显示**。也可以说内联列表就是为制作水平导航而生。  
/\*源码查看bootstrap.css文件第584行~第593行\*/

.list-inline {

padding-left: 0;

margin-left: -5px;

list-style: none;

}

.list-inline > li {

display: inline-block;

padding-right: 5px;

padding-left: 5px;

}

看个示例：

<ul **class="list-inline"**>

    <li>W3cplus</li>

    <li>Blog</li>

    <li>CSS3</li>

    <li>jQuery</li>

    <li>PHP</li>

</ul>

## 列表--水平定义列表

水平定义列表就像内联列表一样，Bootstrap可以给<dl>添加类名“**.dl-horizontal**”给定义列表实现水平显示效果。  
/\*源码请查看bootstrap.css文件第608行~第621行\*/

@media (min-width: 768px) {

.dl-horizontal dt {

float: left;

width: 160px;

overflow: hidden;

clear: left;

text-align: right;

text-overflow: ellipsis;

white-space: nowrap;

}

.dl-horizontal dd {

margin-left: 180px;

}

}

此处添加了一个媒体查询。也就是说，只有屏幕**大于768px**的时候，添加类名“**.dl-horizontal**”才具有水平定义列表效果。其实现主要方式：  
**1、将dt设置了一个左浮动，并且设置了一个宽度为160px  
2、将dd设置一个margin-left的值为180px，达到水平的效果  
3、当标题宽度超过160px时，将会显示三个省略号**

其用法如下：

<dl class="dl-horizontal">

<dt>W3cplus</dt>

<dd>一个致力于推广国内前端行业的技术博客。它以探索为己任，不断活跃在行业技术最前沿，努力提供高质量前端技术博文</dd>

<dt>慕课网</dt>

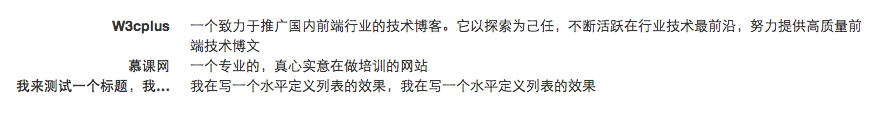
<dd>一个专业的，真心实意在做培训的网站</dd>

<dt>我来测试一个标题，我来测试一个标题</dt>

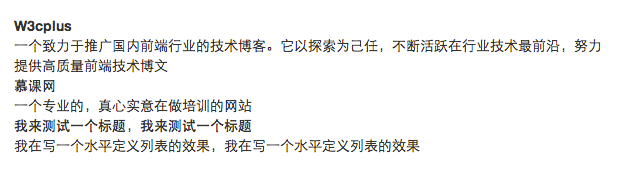
<dd>我在写一个水平定义列表的效果，我在写一个水平定义列表的效果</dd>

</dl>

宽屏下的效果（屏幕**大于768px**）：

[](http://img.mukewang.com/53ad0ff90001418208740117.jpg)

当你缩小你的浏览器屏幕时，水平定义列表将回复到原始的状态。

[](http://img.mukewang.com/53ad10270001db6f06200172.jpg)

本节内容相对来说比较简单，一般在个人博客上使用的较为频繁，用于显示代码的风格。在Bootstrap主要提供了**三种代码风格**：  
1、使用**<code></code>**来显示单行内联代码  
2、使用**<pre></pre>**来显示多行块代码  
3、使用**<kbd></kbd>**来显示用户输入代码

预编译版本的Bootstrap将代码的样式单独提取出来：  
1、LESS版本，请查阅code.less文件  
2、Sass版本，请查阅\_code.scss文件  
编译出来的CSS代码请查阅bootstrap.css文件第688行~第730行，由于代码太长，此处不一一列举。

在使用代码时，用户可以根据具体的需求来使用不同的类型：  
1、**<code>**：一般是针对于**单个单词或单个句子**的代码  
2、**<pre>**：一般是针对于**多行代码**（也就是成块的代码）  
3、**<kbd>**:一般是表示用**户要通过键盘输入的内容**

虽然不同的类型风格不一样，但其使用方法是类似的。  
**code风格：**

<div>Bootstrap的代码风格有三种：**<code>&lt;code&gt;</code>**、**<code>&lt;pre&gt;</code>**和**<code>&lt;kbd&gt;</code>**</div>

**pre风格：**

<div>

**<pre>**

&lt;ul&gt;

&lt;li&gt;...&lt;/li&gt;

&lt;li&gt;...&lt;/li&gt;

&lt;li&gt;...&lt;/li&gt;

&lt;/ul&gt;

**</pre>**

</div>

**kbd风格：**

<div>请输入**<kbd>**ctrl+c**</kbd>**来复制代码，然后使用**<kbd>**ctrl+v**</kbd>**来粘贴代码</div>

不管使用哪种代码风格，在代码中碰到小于号（<）要使用**硬编码**“**&lt;**”来替代，大于号(>)使用“**&gt;**”来替代。而且对于<pre>代码块风格，标签前面留多少个空格，在显示效果中就会留多少个空格。建议在编写HTML标签时，就控制好，例如查看右侧代码编辑器上的15-19行。

## 代码（二）

正如前面所示，<pre>元素一般用于显示大块的代码，并保证原有格式不变。但有时候代码太多，而且不想让其占有太大的页面篇幅，就想控制代码块的大小。Bootstrap也考虑到这一点，你只需要在pre标签上添加类名“**.pre-scrollable**”，就可以控制代码块区域最大高度为**340px**，一旦超出这个高度，就会在**Y轴出现滚动条**。  
/\*源码请查看bootstrap.css第731行~第734行\*/

.pre-scrollable {

max-height: 340px;

overflow-y: scroll;

}

友情提示：如果您了解LESS或Sass这样的CSS预定定义处理器，你完全可以通过**code.less**或者**\_code.scss**文件修改样式，然后重新编译，你就可以得到属于自己的代码样式风格。

## 表格

表格是Bootstrap的一个基础组件之一，Bootstrap为表格提供了**1种基础样式**和**4种附加样式**以及**1个支持响应式的表格**。在使用Bootstrap的表格过程中，只需要添加对应的类名就可以得到不同的表格风格，在接下来的内容中，我们会详细介绍Bootstrap的表格使用。

同样的，如果你对CSS预处理器熟悉，你可以使用Bootstrap提供的预处理版本：  
  ☑  LESS版本，对应的文件是 **tables.less**  
  ☑  Sass版本，对应的文件是 **\_tables.scss**

如果你不懂LESS或Sass也不妨，你在bootstrap.css文件中第1402行~第1630行中可以查阅到所有有关于table的样式代码。由于代码太长，此处不一一列举。

刚已经说了，Bootstrap为表格不同的样式风格提供了不同的类名，主要包括：

  ☑  **.table**：基础表格

  ☑  **.table-striped**：斑马线表格

  ☑  **.table-bordered**：带边框的表格

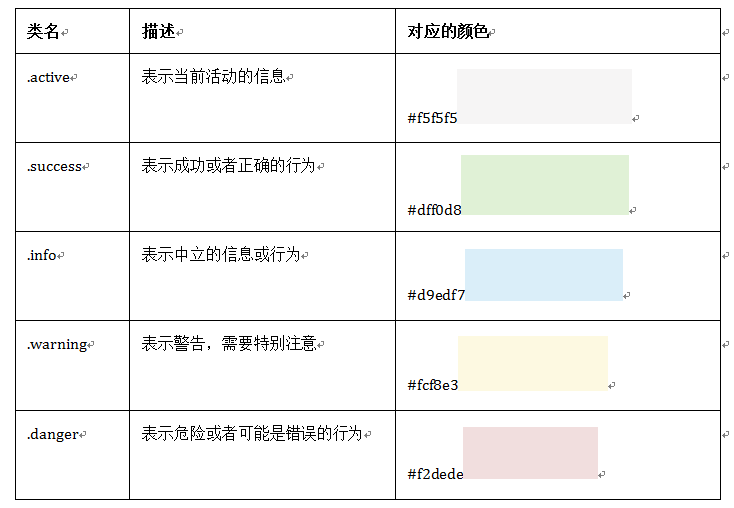
  ☑  **.table-hover**：鼠标悬停高亮的表格

  ☑  **.table-condensed**：紧凑型表格

  ☑  **.table-responsive**：响应式表格

## 表格--表格行的类

Bootstrap还为表格的行元素**<tr>**提供了五种不同的类名，每种类名控制了行的不同背景颜色，具体说明如下表所示：

[](http://img.mukewang.com/53ad213f0001b08807340508.jpg)

其使用非常的简单，只需要在<tr>元素中添加上表对应的类名，就能达到你自己需要的效果：

<tr **class="active"**>

<td>…</td>

</tr>

对应的源码，请查看bootstrap.css文件中第1484行~第1583行。

**特别提示**：除了”.active”之外，其他四个类名和”.table-hover”配合使用时，Bootstrap针对这几种样式也做了相应的悬浮状态的样式设置，所以如果需要给tr元素添加其他颜色样式时，在”.table-hover”表格中也要做相应的调整。

注意要实现悬浮状态，需要在<table>标签上加入table-hover类。

如下代码：

<table class="table-hover">

## 表格--基础表格

大家对表格并不太陌生，但对于Bootstrap中的表格如何使用，或许还有点陌生，接下来的内容，将根据不同的表格类型向大家介绍Bootstrap表格的实际使用方法。  
对表格的结构，跟我们平时使用表格是一样的：

<table>

<thead>

<tr>

<th>表格标题</th>

…

</tr>

</thead>

<tbody>

<tr>

<td>表格单元格</td>

…

</tr>

…

</tbody>

</table>

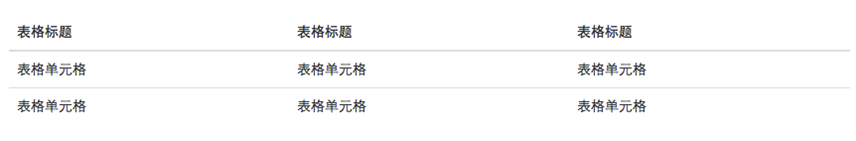
如无特别声明，下面介绍表格类型的时候，结构都是类似上面的代码。  
**基础表格**  
在Bootstrap中，对于基础表格是通过类名“.table”来控制。如果在<table>元素中不添加任何类名，表格是无任何样式效果的。想得到基础表格，我们只需要在<table>元素上添加“.table”类名，就可以得到Bootstrap的基础表格：

<table **class="table"**>

…

</table>

Bootstrap的基础表格，大致长得像下图所示的样子：

[](http://img.mukewang.com/53c617ea0001a48108560141.jpg)

主要源码查看bootstrap.css文件第1402行～第1441行，由于代码太长，此处不一一列举。

**“.table”主要有三个作用：**

**☑  给表格设置了margin-bottom:20px以及设置单元内距**

**☑  在thead底部设置了一个2px的浅灰实线**

**☑  每个单元格顶部设置了一个1px的浅灰实线**

## 表格--斑马线表格

有时候为了让表格更具阅读性，需要将表格制作成类似于斑马线的效果。简单点说就是让表格带有背景条纹效果。在Bootstrap中实现这种表格效果并不困难，只需要在**<table class="table">**的基础上增加类名“**.table-striped**”即可：

<table class="**table table-striped**">

…

</table>

其效果与基础表格相比，仅是在tbody**隔行有一个浅灰色的背景色**。其实现原理也非常的简单，利用CSS3的结构性选择器“:nth-child”来实现，所以对于IE8以及其以下浏览器，没有背景条纹效果。

源码请查看bootstrap.css文件第1465行~第1468行：

.table-striped > tbody > tr:nth-child(odd) > td,

.table-striped > tbody > tr:nth-child(odd) > th {

background-color: #f9f9f9;

}

## 表格--带边框的表格

基础表格仅让表格部分地方有边框，但有时候需要整个表格具有边框效果。Bootstrap出于实际运用，也考虑这种表格效果，即所有单元格具有一条**1px**的边框。  
Bootstrap中带边框的表格使用方法和斑马线表格的使用方法类似，只需要在基础表格<table class="table">基础上添加一个“**.table-bordered**”类名即可：

<table class="table **table-bordered**">

…

</table>

样式如下图所示：

[](http://img.mukewang.com/53c6218300019ab105870211.jpg)

其源码可以查看bootstrap.css文件第1450行~第1464行：

.table-bordered {

border: 1px solid #ddd;/\*整个表格设置边框\*/

}

.table-bordered > thead > tr > th,

.table-bordered > tbody > tr > th,

.table-bordered > tfoot > tr > th,

.table-bordered > thead > tr > td,

.table-bordered > tbody > tr > td,

.table-bordered > tfoot > tr > td {

border: 1px solid #ddd; /\*每个单元格设置边框\*/

}

.table-bordered > thead > tr > th,

.table-bordered > thead > tr > td {

border-bottom-width: 2px;/\*表头底部边框\*/

}

## 表格--鼠标悬浮高亮的表格

当鼠标悬停在表格的行上面有一个高亮的背景色，这样的表格让人看起来就是舒服，时刻告诉用户正在阅读表格哪一行的数据。Bootstrap的确没有让你失望，他也考虑到这种效果，其提供了一个“**.table-hover**”类名来实现这种表格效果。  
鼠标悬停高亮的表格使用也简单，仅需要<table class="table">元素上添加类名“**table-hover**”即可：

<table class="**table table-hover**">

…

</table>

效果图如下：

[](http://img.mukewang.com/53c6224a0001ec1608570206.jpg)

从效果图中可以看出，当你鼠标悬浮在某一单元格上时，单元格所在行的背景色都会变成**浅灰色**。

鼠标悬浮高亮的效果主要是通过“hover”事件来实现，设置了“**tr:hover**”时的th、td的背景色为新颜色。  
其源码请查看bootstrap.css文件中第1469行~第1472行：

.table-hover > tbody > tr:hover > td,

.table-hover > tbody > tr:hover > th {

background-color: #f5f5f5;

}

**注：**其实，鼠标悬浮高亮表格，可以和Bootstrap其他表格混合使用。简单点说，只要你想让你的表格具备悬浮高亮效果，你只要给这个表格添加“**table-hover**”类名就好了。例如，将前面介绍的几种表格结合使用：

<table class="**table table-striped table-bordered** **table-hover**">

…

</table>

## 表格--紧凑型表格

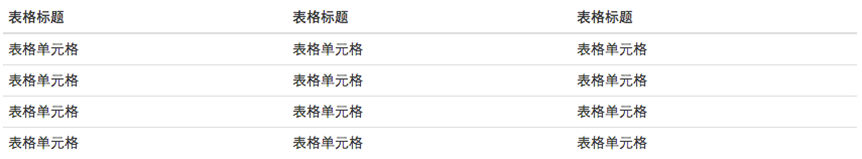
何谓紧凑型表格，简单理解，就是**单元格没内距**或者**内距较其他表格的内距更小**。换句话说，要实现紧凑型表格只需要重置表格单元格的内距**padding**的值。那么在Bootstrap中，通过类名“**table-condensed**”重置了单元格内距值。  
紧凑型表格的运用，也只是需要在<table class="table">基础上添加类名“**table-condensed**”：

<table class="**table table-condensed**">

…

</table>

效果图如下：

[](http://img.mukewang.com/53c62647000158e708620159.jpg)

从上面效果图可以看出，Bootstrap中紧凑型的表格与基础表格差别不大，因为只是将单元格的内距由**8px调至5px**。

源码请查看bootstrap.css文件第1442行~第1449行：

.table-condensed > thead > tr > th,

.table-condensed > tbody > tr > th,

.table-condensed > tfoot > tr > th,

.table-condensed > thead > tr > td,

.table-condensed > tbody > tr > td,

.table-condensed > tfoot > tr > td {

padding: 5px;

}

正如上一小节中悬浮高亮表格中所讲解的，在Bootstrap中制作表格中，可以将上面几种表格样式结合在一起使用，从而制作出更为精美的表格。结合的方法也很简单，就是在<table class="table">基础上添加你需要的表格样式类型。

另外从上面的示例中大家可能也发现了，不管制作哪种表格都离不开类名“table”。所以大家在使用Bootstrap表格时，千万注意，你的<table>元素中一定不能缺少类名“table”。

## 表格--响应式表格

随着各种手持设备的出现，要想让你的Web页面适合千罗万像的设备浏览，**响应式设计**的呼声越来越高。在Bootstrap中也为表格提供了响应式的效果，将其称为**响应式表格**。  
Bootstrap提供了一个容器，并且此容器设置类名“.table-responsive”,此容器就具有响应式效果，然后将<table class="table">置于这个容器当中，这样表格也就具有响应式效果。  
Bootstrap中响应式表格效果表现为：当你的浏览器可视区域小于**768px**时，表格底部会出现水平滚动条。当你的浏览器可视区域大于768px时，表格底部水平滚动条就会消失。示例如下：

<div class="table-responsive">

<table class="table table-bordered">

…

</table>

</div>

运行效果如下：

[](http://img.mukewang.com/53ad2eab0001c55105540130.jpg)

（宽屏效果）

[](http://img.mukewang.com/53ad2f540001847402420197.jpg)

（窄屏效果）

## 基础表单

表单主要功能是用来与用户做交流的一个网页控件，良好的表单设计能够让网页与用户更好的沟通。表单中常见的元素主要包括：**文本输入框**、**下拉选择框、单选按钮、复选按钮**、**文本域**和**按钮**等。其中每个控件所起的作用都各不相同，而且不同的浏览器对表单控件渲染的风格都各有不同。

同样，表单也是Bootstrap框架中的核心内容，下面向大家介绍Bootstrap框架中表单的制作。

**表单源码查询**

Bootstrap框架的表单，其源码占据了大量的代码，同样的，根据不同的Bootstrap版本，你可以轻松获取相应的源码：

   ☑   LESS版本：对应源文件 **forms.less**

   ☑ ​  Sass版本：对应源文件 **\_forms.scss**

编译后的Bootstrap版本，可以查阅bootstrap.css文件第1630行~第1991行

不过在样式表中，还对表单做了一些初始化，详细代码可以查阅bootstrap.css文件第110行~第178行。

对于基础表单，Bootstrap并未对其做太多的定制性效果设计，仅仅对表单内的**fieldset**、**legend**、**label**标签进行了定制。如：

/\*源码请查阅bootstrap.css文件第1631行~第1652行\*/

fieldset {

min-width: 0;

padding: 0;

margin: 0;

border: 0;

}

legend {

display: block;

width: 100%;

padding: 0;

margin-bottom: 20px;

font-size: 21px;

line-height: inherit;

color: #333;

border: 0;

border-bottom: 1px solid #e5e5e5;

}

label {

display: inline-block;

margin-bottom: 5px;

font-weight: bold;

}

主要将这些元素的**margin**、**padding**和**border**等进行了细化设置。

当然表单除了这几个元素之外，还有**input**、**select**、**textarea**等元素，在Bootstrap框架中，通过定制了一个类名`**form-control**`，也就是说，如果这几个元素使用了类名“form-control”，将会实现一些设计上的定制效果。

**1、宽度变成了100%**

**2、设置了一个浅灰色（#ccc）的边框**

**3、具有4px的圆角**

**4、设置阴影效果，并且元素得到焦点之时，阴影和边框效果会有所变化**

**5、设置了placeholder的颜色为#999**

详细请查阅bootstrap.css文件第1690行~第1732行。

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b230bb0001cea602970184.jpg)

## 内联表单

有时候我们需要将表单的控件都在一行内显示，类似这样的：

[http://img.mukewang.com/53b2532a000107b003190032.jpg](http://img.mukewang.com/53b2532a000107b003190032.jpg)

在Bootstrap框架中实现这样的表单效果是轻而易举的，你只需要在<form>元素中添加类名“form-inline”即可。  
内联表单实现原理非常简单，欲将表单控件在一行显示，就需要将表单控件设置成内联块元素（display:inline-block）。  
/\*源码请查阅bootstrap.css文件第1928行~第1962行\*/

如果你要在input前面添加一个label标签时，会导致input换行显示。如果你必须添加这样的一个label标签，并且不想让input换行，你需要将label标签也放在容器“form-group”中，如：

<div **class="form-group"**>

<label class="sr-only" for="exampleInputEmail2">Email address</label>

</div>

<div **class="form-group"**>

<inputtype="email" class="form-control" id="exampleInputEmail2" placeholder="Enter email">

</div>

接下来，我们还是以实例说话：

<form **class="form-inline"** role="form">

<div **class="form-group"**>

<label class="sr-only" for="exampleInputEmail2">邮箱</label>

<input type="email" class="form-control" id="exampleInputEmail2" placeholder="请输入你的邮箱地址">

</div>

<div **class="form-group"**>

<label class="sr-only" for="exampleInputPassword2">密码</label>

<input type="password" class="form-control" id="exampleInputPassword2" placeholder="请输入你的邮箱密码">

</div>

<div class="checkbox">

<label>

<input type="checkbox">记住密码

</label>

</div>

<button type="submit" class="btnbtn-default">进入邮箱</button>

</form>

运行效果如下或查看右侧结果窗口：(查看效果需要把结果窗口设置为全屏)

[http://img.mukewang.com/53b253ae00011c8003170053.jpg](http://img.mukewang.com/53b253ae00011c8003170053.jpg)

回过头来看示例，你或许会问，为什么添加了label标签，而且没有放置在”form-group”这样的容器中，input也不会换行；还有label标签怎么没显示出来。如果你仔细看，在label标签运用了一个类名“sr-only”，标签没显示就是这个样式将标签隐藏了。  
/\*源码请查阅bootstrap.css文件第342行~第350行\*/

.sr-only {

position: absolute;

width: 1px;

height: 1px;

padding: 0;

margin: -1px;

overflow: hidden;

clip: rect(0, 0, 0, 0);

border: 0;

}

注意：那么Bootstrap为什么要这么做呢？这样不是多此一举吗？其实不是的，如果没有为输入控件设置label标签，**屏幕阅读器**将无法正确识别。这也是Bootstrap框架另一个优点之处，为残障人员进行了一定的考虑。

## 表单控件(输入框input)

每一个表单都是由表单控件组成。离开了控件，表单就失去了意义。接下来的我们简单的来了解Bootstrap框架中表单控件的相关知识。

**单行输入框**，常见的文本输入框，也就是**input**的**type**属性值为**text**。在Bootstrap中使用input时也必须添加type类型，如果没有指定type类型，将无法得到正确的样式，因为Bootstrap框架都是通过**input[type=“?”]**(其中?号代表type类型，比如说text类型，对应的是**input[type=“text”]**)的形式来定义样式的。

为了让控件在各种表单风格中样式不出错，需要添加类名“**form-control**”，如：

<form role="form">

<div class="form-group">

<input **type="email"** class="**form-control**" placeholder="Enter email">

</div>

</form>

运行效果如下或查看右侧结果窗口:

[http://img.mukewang.com/53b2571700018ab503070043.jpg](http://img.mukewang.com/53b2571700018ab503070043.jpg)

## 表单控件(下拉选择框select)

Bootstrap框架中的下拉选择框使用和原始的一致，多行选择设置**multiple**属性的值为**multiple**。Bootstrap框架会为这些元素提供统一的样式风格。如：

<form role="form">

<div class="form-group">

  <select class="**form-control**">

    <option>1</option>

    <option>2</option>

    <option>3</option>

    <option>4</option>

    <option>5</option>

  </select>

  </div>

  <div class="form-group">

  <select **multiple** class="**form-control**">

    <option>1</option>

    <option>2</option>

    <option>3</option>

    <option>4</option>

    <option>5</option>

  </select>

</div>

</form>

运行效果如下或查看右侧结果窗口:

[](http://img.mukewang.com/53b257ea000136bd02570072.jpg)

## 表单控件(文本域textarea)

文本域和原始使用方法一样，设置**rows**可定义其高度，设置**cols**可以设置其宽度。但如果**textarea**元素中添加了类名“**form-control**”类名，则无需设置cols属性。因为Bootstrap框架中的“form-control”样式的表单控件宽度为**100%**或**auto**。

<form role="form">

<div class="form-group">

**<textarea class="form-control" rows="3"></textarea>**

</div>

</form>

运行效果如下或查看右侧结果窗口：

[http://img.mukewang.com/53b25d4d0001dbb503010055.jpg](http://img.mukewang.com/53b25d4d0001dbb503010055.jpg)

## 表单控件(复选框checkbox和单选择按钮radio)

Bootstrap框架中checkbox和radio有点特殊，Bootstrap针对他们做了一些特殊化处理，主要是checkbox和radio与label标签配合使用会出现一些小问题（最头痛的是对齐问题）。使用Bootstrap框架，开发人员无需考虑太多，只需要按照下面的方法使用即可。

<form role="form">

<div class="checkbox">

<label>

<input type="checkbox" value="">

记住密码

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios1" value="love" checked>

喜欢

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="optionsRadios" id="optionsRadios2" value="hate">

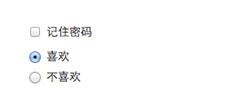
不喜欢

</label>

</div>

</form>

运行效果如下或查看右侧结果窗口（案例1）：

[](http://img.mukewang.com/53b25edb0001faae02520102.jpg)

从上面的示例，我们可以得知：  
1、不管是checkbox还是radio都使用label包起来了  
2、checkbox连同label标签放置在一个名为“.checkbox”的容器内  
3、radio连同label标签放置在一个名为“.radio”的容器内  
在Bootstrap框架中，主要借助“.checkbox”和“.radio”样式，来处理复选框、单选按钮与标签的对齐方式。源码请查看bootstrap.css文件第1742行～第1762行：

.radio,

.checkbox {

display: block;

min-height: 20px;

padding-left: 20px;

margin-top: 10px;

margin-bottom: 10px;

}

.radio label,

.checkbox label {

display: inline;

font-weight: normal;

cursor: pointer;

}

.radio input[type="radio"],

.radio-inline input[type="radio"],

.checkbox input[type="checkbox"],

.checkbox-inline input[type="checkbox"] {

float: left;

margin-left: -20px;

}

.radio + .radio,

.checkbox + .checkbox {

margin-top: -5px;

}

## 表单控件(复选框和单选按钮水平排列)

有时候，为了布局的需要，将复选框和单选按钮需要水平排列。Bootstrap框架也做了这方面的考虑：  
**1、如果checkbox需要水平排列，只需要在label标签上添加类名“checkbox-inline”  
2、如果radio需要水平排列，只需要在label标签上添加类名“radio-inline”**  
如下所示：

<form role="form">

  <div class="form-group">

    <label class="**checkbox-inline**">

      <input type="checkbox"  value="option1">游戏

    </label>

    <label class="**checkbox-inline**">

      <input type="checkbox"  value="option2">摄影

    </label>

    <label class="**checkbox-inline**">

    <input type="checkbox"  value="option3">旅游

    </label>

  </div>

  <div class="form-group">

    <label class="**radio-inline**">

      <input type="radio"  value="option1" name="sex">男性

    </label>

    <label class="**radio-inline**">

      <input type="radio"  value="option2" name="sex">女性

    </label>

    <label class="**radio-inline**">

      <input type="radio"  value="option3" name="sex">中性

    </label>

  </div>

</form>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b2649f00011bae01920069.jpg)

实现源码请查看bootstrap.css文件第1767行～第1780行：

.radio-inline,

.checkbox-inline {

display: inline-block;

padding-left: 20px;

margin-bottom: 0;

font-weight: normal;

vertical-align: middle;

cursor: pointer;

}

.radio-inline + .radio-inline,

.checkbox-inline + .checkbox-inline {

margin-top: 0;

margin-left: 10px;

}

## 表单控件(按钮)

按钮也是表单重要控件之一,制作按钮通常使用下面代码来实现：

☑  **input[type=“submit”]**

☑  **input[type=“button”]**

☑  **input[type=“reset”]**

☑  **<button>**

在Bootstrap框架中的按钮都是采用<button>来实现。

有关于Bootstrap中按钮如何制作，在这里不做过多阐述，因为按钮也是Bootstrap框架中核心部分之一，后面我们专门有一节内容来介绍Bootstrap的按钮。

这里先让大家看看Bootstrap的按钮长成什么样：

[http://img.mukewang.com/53b266f800010e4703160035.jpg](http://img.mukewang.com/53b266f800010e4703160035.jpg)

## 表单控件大小

前面看到的表单控件都正常的大小。可以通过设置控件的**height，line-height，padding**和**font-size**等属性来实现控件的高度设置。不过Bootstrap框架还提供了两个不同的类名，用来控制表单控件的高度。这两个类名是：  
**1、input-sm:让控件比正常大小更小  
2、input-lg:让控件比正常大小更大**

这两个类适用于表单中的**input**，**textarea**和**select**控件，具体使用如下：

<input class="form-control **input-lg**" type="text" placeholder="添加.input-lg，控件变大">

<input class="form-control" type="text" placeholder="正常大小">

<input class="form-control **input-sm**" type="text" placeholder="添加.input-sm，控件变小">

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b269860001e43f02260101.jpg)

源码请查阅bootstrap.css文件第1795~第1824行：

.input-sm {

height: 30px;

padding: 5px 10px;

font-size: 12px;

line-height: 1.5;

border-radius: 3px;

}

select.input-sm {

height: 30px;

line-height: 30px;

}

textarea.input-sm,

select[multiple].input-sm {

height: auto;

}

.input-lg {

height: 46px;

padding: 10px 16px;

font-size: 18px;

line-height: 1.33;

border-radius: 6px;

}

select.input-lg {

height: 46px;

line-height: 46px;

}

textarea.input-lg,

select[multiple].input-lg {

height: auto;

}

从上面的源码中不难发现，不管是“input-sm”还是“input-lg”仅对控件高度做了处理。但往往很多时候，我们需要控件宽度也要做一定的变化处理。这个时候就要借住Bootstrap框架的**网格系统**。所以你要控制表单宽度，可以像下面这样使用：

<form role="form" class="form-horizontal">

  <div class="form-group">

  <div class="**col-xs-4**">

    <input class="form-control input-lg" type="text" placeholder=".col-xs-4">

  </div>

  <div class="**col-xs-4**">

    <input class="form-control input-lg" type="text" placeholder=".col-xs-4">

  </div>

  <div class="**col-xs-4**">

    <input class="form-control input-lg" type="text" placeholder=".col-xs-4">

  </div>

  </div>

    …

</form>

注：网格布局在后面章节中会进行详细讲解。

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b26a6a00018cba02420078.jpg)

前面介绍水平表单时说过，如果表单使用了类名“form-horizontal”，其中“form-group”就相当于网格系统中的“row”。换句话说，如果没有这样做，要通过网格系统来控制表单控件宽度，就需要这样使用：

<div class="row">

<div class="col-xs-4">

<input class="form-control input-lg" type="text" placeholder=".col-xs-4">

</div>

<div class="col-xs-4">

<input class="form-control input-lg" type="text" placeholder=".col-xs-4">

</div>

<div class="col-xs-4">

<input class="form-control input-lg" type="text" placeholder=".col-xs-4">

</div>

</div>

## 表单控件状态(焦点状态)

表单主要用来与用户沟通，好的表单就能更好的与用户进行沟通，而好的表单一定离不开表单的控件状态。

**表单状态的作用：**

每一种状态都能给用户传递不同的信息，比如表单有焦点的状态可以告诉用户可以输入或选择东西，禁用状态可以告诉用户不可以输入或选择东西，还有就是表单控件验证状态，可以告诉用户的操作是否正确等。那么在Bootstrap框架中的表单控件也具备这些状态。

焦点状态是通过伪类“:focus”来实现。Bootstrap框架中表单控件的焦点状态删除了**outline**的默认样式，重新添加阴影效果。

源码请查阅bootstrap.css文件第1707行～第1712行：

.form-control:focus {

border-color: #66afe9;

outline: 0;

-webkit-box-shadow: inset 0 1px 1pxrgba(0,0,0,.075), 0 0 8px rgba(102, 175, 233, .6);

box-shadow: inset 0 1px 1pxrgba(0,0,0,.075), 0 0 8px rgba(102, 175, 233, .6);

}

 从源码中我们可以看出，要让控件在焦点状态下有上面样式效果，需要给控件添加类名“form-control”：

<form role="form" class="form-horizontal">

<div class="form-group">

<div class="col-xs-6">

<input class="**form-control** input-lg" type="text" placeholder="不是焦点状态下效果">

</div>

<div class="col-xs-6">

<input class="**form-control** input-lg" type="text" placeholder="焦点点状态下效果">

</div>

</div>

</form>

运行效果如下或查看右侧结果窗口：

[http://img.mukewang.com/53b271e700018f8a02780042.jpg](http://img.mukewang.com/53b271e700018f8a02780042.jpg)

（鼠标单击输入框，使其获得焦点就可以看到加入蓝色边框效果）

在Bootstrap框架中，**file**、**radio**和**checkbox**控件在焦点状态下的效果也与普通的input控件不太一样，主要是因为Bootstrap对他们做了一些特殊处理：  
/\*源码查看boostrap.css文件第1676行～第1682行\*/

input[type="file"]:focus,

input[type="radio"]:focus,

input[type="checkbox"]:focus {

outline: thin dotted;

outline: 5px auto -webkit-focus-ring-color;

outline-offset: -2px;

}

## 表单控件状态(禁用状态)

Bootstrap框架的表单控件的禁用状态和普通的表单禁用状态实现方法是一样的，在相应的表单控件上添加属性“disabled”。和其他表单的禁用状态不同的是，Bootstrap框架做了一些样式风格的处理：  
/\*源码请查看bootstrap.css文件第1723行～第1729行\*/

.form-control[disabled],

.form-control[readonly],

fieldset[disabled] .form-control {

cursor: not-allowed;

background-color: #eee;

opacity: 1;

}

使用方法为：只需要在需要禁用的表单控件上加上“disabled”即可：

<input class="form-control" type="text" placeholder="表单已禁用，不能输入" disabled>

运行效果如下或查看右侧结果窗口：

[http://img.mukewang.com/53b27abf00013e5f03360048.jpg](http://img.mukewang.com/53b27abf00013e5f03360048.jpg)

在使用了“form-control”的表单控件中，样式设置了禁用表单背景色为灰色，而且手型变成了不准输入的形状。如果控件中不使用类名“form-control”，禁用的控件只会有一个不准输入的手型出来。  
/\*源码请查阅bootstrap.css文件第1781行～第1794行\*/

input[type="radio"][disabled],

input[type="checkbox"][disabled],

.radio[disabled],

.radio-inline[disabled],

.checkbox[disabled],

.checkbox-inline[disabled],

fieldset[disabled] input[type="radio"],

fieldset[disabled] input[type="checkbox"],

fieldset[disabled] .radio,

fieldset[disabled] .radio-inline,

fieldset[disabled] .checkbox,

fieldset[disabled] .checkbox-inline {

cursor: not-allowed;

}

在Bootstrap框架中，如果fieldset设置了disabled属性，整个域都将处于被禁用状态。

<form role="form">

<fieldset disabled>

  <div class="form-group">

  <label for="disabledTextInput">禁用的输入框</label>

    <input type="text" id="disabledTextInput" class="form-control" placeholder="禁止输入">

  </div>

  <div class="form-group">

  <label for="disabledSelect">禁用的下拉框</label>

    <select id="disabledSelect" class="form-control">

  <option>不可选择</option>

  </select>

  </div>

  <div class="checkbox">

  <label>

    <input type="checkbox">无法选择

  </label>

  </div>

  <button type="submit" class="btnbtn-primary">提交</button>

</fieldset>

</form>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b27b2200010df702890118.jpg)

据说对于整个禁用的域中，如果legend中有输入框的话，这个**输入框是无法被禁用**的。我们一起来验证一下：

<form role="form">

<fieldset **disabled**>

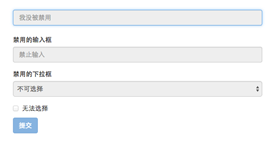
<legend><input type="text" class="form-control" placeholder="显然我颜色变灰了，但是我没被禁用，不信？单击试一下" /></legend>

…

</fieldset>

</form>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b27bc30001113302720144.jpg)

## 表单控件状态(验证状态)

在制作表单时，不免要做表单验证。同样也需要提供验证状态样式，在Bootstrap框架中同样提供这几种效果。  
**1、.has-warning:警告状态（黄色）  
2、.has-error：错误状态（红色）  
3、.has-success：成功状态（绿色）**  
使用的时候只需要在form-group容器上对应添加状态类名。

<form role="form">

<div class="form-group has-success">

<label class="control-label" for="inputSuccess1">成功状态</label>

<input type="text" class="form-control" id="inputSuccess1" placeholder="成功状态" >

</div>

<div class="form-group has-warning">

<label class="control-label" for="inputWarning1">警告状态</label>

<input type="text" class="form-control" id="inputWarning1" placeholder="警告状态">

</div>

<div class="form-group has-error">

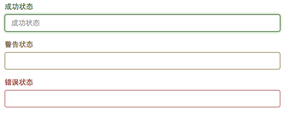
<label class="control-label" for="inputError1">错误状态</label>

<input type="text" class="form-control" id="inputError1" placeholder="错误状态">

</div>

</form>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b27e170001133702870115.jpg)

从效果可以看出，三种状态下效果都是一样的，只是颜色不一样而以。

源码请查阅bootstrap.css文件第1481行～第1918行。

其他两种状态省略源码不在此展示。  
很多时候，在表单验证的时候，不同的状态会提供不同的icon，比如成功是一个对号（√），错误是一个叉号（×）等。在Bootstrap框中也提供了这样的效果。如果你想让表单在对应的状态下显示icon出来，只需要在对应的状态下添加类名“has-feedback”。请注意，此类名要与“has-error”、“has-warning”和“has-success”在一起：

<form role="form">

<div class="form-group has-success has-feedback">

<label class="control-label" for="inputSuccess1">成功状态</label>

<input type="text" class="form-control" id="inputSuccess1" placeholder="成功状态" >

<span class="glyphiconglyphicon-ok form-control-feedback"></span>

</div>

<div class="form-group has-warning has-feedback">

<label class="control-label" for="inputWarning1">警告状态</label>

<input type="text" class="form-control" id="inputWarning1" placeholder="警告状态">

<span class="glyphiconglyphicon-warning-sign form-control-feedback"></span>

</div>

<div class="form-group has-error has-feedback">

<label class="control-label" for="inputError1">错误状态</label>

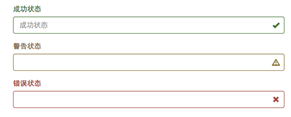
<input type="text" class="form-control" id="inputError1" placeholder="错误状态">

<span class="glyphiconglyphicon-remove form-control-feedback"></span>

</div>

</form>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b27e8600013fdf02910117.jpg)

从效果图中可以看出，图标都居右。在Bootstrap的小图标都是使用**@font-face**来制作（后面的内容中将会着重用一节内容来介绍）。而且必须在表单中添加了一个span元素：

<span class="glyphiconglyphicon-warning-sign form-control-feedback"></span>

## 表单提示信息

平常在制作表单验证时，要提供不同的提示信息。在Bootstrap框架中也提供了这样的效果。使用了一个"help-block"样式，将提示信息以块状显示，并且显示在控件底部。

<form role="form">

<div class="form-group has-success has-feedback">

<label class="control-label" for="inputSuccess1">成功状态</label>

<input type="text" class="form-control" id="inputSuccess1" placeholder="成功状态" >

<span class="help-block">你输入的信息是正确的</span>

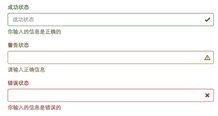
<span class="glyphiconglyphicon-ok form-control-feedback"></span>

</div>

…

</form>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b2829200018aaf02180116.jpg)

具体样式代码请查看bootstrap.css文件第1922行～第1927行：

.help-block {

display: block;

margin-top: 5px;

margin-bottom: 10px;

color: #737373;

}

在Bootstrap V2.x版本中还提供了一个行内提示信息，其使用了类名“help-inline”。一般让提示信息显示在控件的后面，也就是同一水平显示。如果你想在BootstrapV3.x版本也有这样的效果，你可以添加这段代码：

.help-inline{

display:inline-block;

padding-left:5px;

color: #737373;

}

如果你不想为bootstrap.css增加自己的代码，而且设计又有这种样的需求，那么只能借助于Bootstrap的网格系统。(网格系统在后面的章节中会详细讲解)

<form role="form">

<div class="form-group">

<label class="control-label" for="inputSuccess1">成功状态</label>

<div class="row">

<div class="col-xs-6">

<input type="text" class="form-control" id="inputSuccess1" placeholder="成功状态" >

</div>

**<span class="col-xs-6 help-block">你输入的信息是正确的</span>**

</div>

</div>

</form>

运行效果如下或查看右侧结果窗口：

[http://img.mukewang.com/53b283010001402902670049.jpg](http://img.mukewang.com/53b283010001402902670049.jpg)

结束语：有关于Bootstrap框架中表单的运用除了按钮部分，到此就算是介绍完了。如果你觉得这样的表单效果并不是你需要的，你完全可以通过**forms.less**或者**\_forms.scss**文件进行定制，然后重新编译就可以得到你需要的表单效果。在接下来的一节中，我们Bootstrap框架中另一个核心内容——**按钮**。

## 基本按钮

Bootstrap框架V3.x版本的基本按钮和V2.x版本的基本按钮一样，都是通过类名“btn”来实现。不同的是在V3.x版本要简约很多，去除了V2.x版本中的大量的CSS3中的部分特效，比如说文本阴影（text-shadow）、渐变背景（background-image）、边框阴影（box-shadow）等。  
难能可贵的是，Bootstrap框架中的考虑了不同浏览器的解析差异，进行了比较安全的兼容性处理，使按钮效果在不同的浏览器中所呈现的效果基本相同。

源码请查阅bootstrap.css文件第1992行～第2010行：

.btn {

display: inline-block;

padding: 6px 12px;

margin-bottom: 0;

font-size: 14px;

font-weight: normal;

line-height: 1.42857143;

text-align: center;

white-space: nowrap;

vertical-align: middle;

cursor: pointer;

-webkit-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

background-image: none;

border: 1px solid transparent;

border-radius: 4px;

}

Bootstrap框架的按钮使用非常的简单，使用方式如下：

<button class="btn" type="button">我是一个基本按钮</button>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b28609000143a602800084.jpg)

## 多标签支持

一般制作按钮除了使用<button>标签元素之外，还可以使用<input type="submit">和<a>标签等。同样，在Bootstrap框架中制作按钮时，除了刚才所说的这些标签元素之外，还可以使用在其他的标签元素上，唯一需要注意的是，要在制作按钮的标签元素上添加类名“btn”。如果不添加是不会有任何按钮效果。

我们一起来看看其他标签制作的基本按钮效果：

<button **class="btn btn-default"** type="button">button标签按钮</button>

<input type="submit" **class="btn btn-default"** value="input标签按钮"/>

<a href="##" **class="btn btn-default"**>a标签按钮</a>

<span **class="btn btn-default"**>span标签按钮</span>

<div **class="btn btn-default"**>div标签按钮</div>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b287f800014bc402460068.jpg)

注意：虽然在Bootstrap框架中使用任何标签元素都可以实现按钮风格，但个人并不建议这样使用，为了避免浏览器兼容性问题，个人强烈建议使用**button**或**a**标签来制作按钮。

## 定制风格

在介绍按钮开篇就说过，Web页面可能会有不同的按钮风格。那么在Bootstrap框架也考虑了。在Bootstrap框架中除了默认的按钮风格之外，还有其他六种按钮风格，每种风格的其实都一样，不同之处就是按钮的背景颜色、边框颜色和文本颜色。

在Bootstrap框架中不同的按钮风格都是通过不同的类名来实现，在使用过程中，开发者只需要选择不同的类名即可：

[](http://img.mukewang.com/53b367bd0001d59c07530312.jpg)

[](http://img.mukewang.com/53b367d10001846a08020810.jpg)

使用起来就很简单，就像前面介绍的默认按钮一样的使用方法，只需要在基础按钮“.btn”基础上追加**对应的类名**，就可以得到需要的按钮风格。如：

<button class="**btn**" type="button">基础按钮.btn</button>

<button class="**btn btn-default**" type="button">默认按钮.btn-default</button>

<button class="**btn btn-primary**" type="button">主要按钮.btn-primary</button>

<button class="**btn btn-success**" type="button">成功按钮.btn-success</button>

<button class="**btn btn-info**" type="button">信息按钮.btn-info</button>

<button class="**btn btn-warning**" type="button">警告按钮.btn-warning</button>

<button class="**btn btn-danger**" type="button">危险按钮.btn-danger</button>

<button class="**btn btn-link**" type="button">链接按钮.btn-link</button>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b368510001fe2701800274.jpg)

## 按钮大小

上一节介绍了按钮的定制风格，也就是如何实现Web页面中多种风格按钮。在Bootstrap框架中，对于按钮的大小，也是可以定制的。类似于input一样，通过在基础按钮“.btn”的基础上追加类名来控制按钮的大小。

在Bootstrap框架中提供了三个类名来控制按钮大小：

[](http://img.mukewang.com/53b36a7600014af106910605.jpg)

从上表中不难发现，在Bootstrap框架中控制按钮的大小都是通过修改按钮的**padding**、**line-height**、**font-size**和**border-radius**几个属性。  
/\*源码查阅bootstrap.css文件中第2319行～第2339行\*/

.btn-lg,

.btn-group-lg> .btn {

padding: 10px 16px;

font-size: 18px;

line-height: 1.33;

border-radius: 6px;

}

.btn-sm,

.btn-group-sm> .btn {

padding: 5px 10px;

font-size: 12px;

line-height: 1.5;

border-radius: 3px;

}

.btn-xs,

.btn-group-xs> .btn {

padding: 1px 5px;

font-size: 12px;

line-height: 1.5;

border-radius: 3px;

}

那么在实际使用中，这几个类名可以配合按钮中其他颜色类名一起使用，但唯一一点不能缺少“.btn”类名：

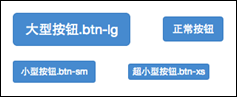
<button class="btn **btn-primary** **btn-lg**" type="button">大型按钮.btn-lg</button>

<button class="btn **btn-primary**" type="button">正常按钮</button>

<button class="btn **btn-primary btn-sm**" type="button">小型按钮.btn-sm</button>

<button class="btn **btn-primary btn-xs**" type="button">超小型按钮.btn-xs</button>

效果如下图或查看右侧结果窗口：

[](http://img.mukewang.com/53b36b220001f92302370097.jpg)

## 块状按钮

从前面几节的内容中，大家可能发现了，每个示例中的按钮宽度都是依靠按钮文本和padding的值来决定。但有时候在制作按钮的时候需要按钮宽度充满整个父容器**（width:100%）**，特别是在移动端的制作中。那么前面的方法我们都无法很好的实现，除非重新定义按钮的宽度。其实在Bootstrap中并不需要这样做，Bootstrap框架中提供了一个类名“btn-block”。按钮使用这个类名就可以让按钮充满整个容器，并且这个按钮不会有任何的**padding**和**margin**值。在实际当中，常把这种按钮称为**块状按钮**。具体源码请查阅bootstrap.css文件第2340行～第2353行：

.btn-block {

display: block;

width: 100%;

padding-right: 0;

padding-left: 0;

}

.btn-block + .btn-block {

margin-top: 5px;

}

input[type="submit"].btn-block,

input[type="reset"].btn-block,

input[type="button"].btn-block {

width: 100%;

}

使用方法和前面的类似，只需要在原按钮类名上添加“.btn-block”类名，当然“.btn”类名是不可或缺的：

<button class="**btnbtn-primary btn-lg btn-block**" type="button">大型按钮.btn-lg</button>

<button class="**btnbtn-primary btn-block**" type="button">正常按钮</button>

<button class="**btnbtn-primary btn-sm btn-block**" type="button">小型按钮.btn-sm</button>

<button class="**btnbtn-primary btn-xs btn-block**" type="button">超小型按钮.btn-xs</button>

运行效果如下：

[](http://img.mukewang.com/53b36c2b000130f202260110.jpg)

## 按钮状态——禁用状态

和input等表单控件一样，在Bootstrap框架的按钮中也具有禁用状态的设置。禁用状态与其他状态按钮相比，就是背景颜色的透明度做了一定的处理，opcity的值从100%调整为65%。

**在Bootstrap框架中，要禁用按钮有两种实现方式：**

**方法1：**在标签中添加disabled属性

**方法2：**在元素标签中添加类名“disabled”

**两者的主要区别是：**

“.disabled”样式不会禁止按钮的默认行为，比如说提交和重置行为等。如果想要让这样的禁用按钮也能禁止按钮的默认行为，则需要通过JavaScript这样的语言来处理。对于<a>标签也存在类似问题，如果通过类名“.disable”来禁用按钮，其链接行为是无法禁止。而在元素标签中添加“disabled”属性的方法是可以禁止元素的默认行为的。

**下面是两种方法的举例：**

<button class="btnbtn-primary btn-lgbtn-block" type="button" disabled="disabled">通过disabled属性禁用按钮</button>

<button class="btnbtn-primary btn-block disabled" type="button">通过添加类名disabled禁用按钮</button>

<button class="btnbtn-primary btn-smbtn-block" type="button">未禁用的按钮</button>

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b36e9d0001132e02690107.jpg)

对应的样式代码请查阅bootstrap.css文件第2030行～第2039行：

.btn.disabled,

.btn[disabled],

fieldset[disabled] .btn {

pointer-events: none;

cursor: not-allowed;

filter: alpha(opacity=65);

-webkit-box-shadow: none;

box-shadow: none;

opacity: .65;

}

同样的，其他风格按钮也具有这样的效果，只是颜色做了一定的调整，比如信息按钮(.btn-info)所示：  
/\*源码请查看bootstrap.css文件第2182行～第2199行\*/

.btn-info.disabled,

.btn-info[disabled],

fieldset[disabled] .btn-info,

.btn-info.disabled:hover,

.btn-info[disabled]:hover,

fieldset[disabled] .btn-info:hover,

.btn-info.disabled:focus,

.btn-info[disabled]:focus,

fieldset[disabled] .btn-info:focus,

.btn-info.disabled:active,

.btn-info[disabled]:active,

fieldset[disabled] .btn-info:active,

.btn-info.disabled.active,

.btn-info[disabled].active,

fieldset[disabled] .btn-info.active {

background-color: #5bc0de;

border-color: #46b8da;

}

到此有关于按钮的基础知识就算是介绍完了，同样的，大家可以通过buttons.less或者\_buttons.scss来自定义按钮风格。

## 图像

图像在网页制作中也是常要用到的元素，在Bootstrap框架中对于图像的样式风格提供以下几种风格：

**1、img-responsive：**响应式图片，主要针对于响应式设计  
**2、img-rounded：**圆角图片  
**3、img-circle：**圆形图片  
**4、img-thumbnail：**缩略图片

**使用方法：**

使用方法非常简单，只需要在**<img>**标签上添加对应的类名，如下代码：

<img alt="140x140" src="http://placehold.it/140x140">

<img class="img-rounded" alt="140x140" src="http://placehold.it/140x140">

<img class="img-circle" alt="140x140" src="http://placehold.it/140x140">

<img class="img-thumbnail" alt="140x140" src="http://placehold.it/140x140">

<img class="img-responsive" alt="140x140" src="http://placehold.it/140x140">

运行效果如下或查看右侧结果窗口：

[](http://img.mukewang.com/53b36fc300017b1e02980139.jpg)

每种状态对应的源码可以查阅bootstrap.css文件第306行～第335行：

img {

vertical-align: middle;

}

.img-responsive,

.thumbnail>img,

.thumbnail a >img,

.carousel-inner > .item >img,

.carousel-inner > .item > a >img {

display: block;

max-width: 100%;

height: auto;

}

.img-rounded {

border-radius: 6px;

}

.img-thumbnail {

display: inline-block;

max-width: 100%;

height: auto;

padding: 4px;

line-height: 1.42857143;

background-color: #fff;

border: 1px solid #ddd;

border-radius: 4px;

-webkit-transition: all .2s ease-in-out;

transition: all .2s ease-in-out;

}

.img-circle {

border-radius: 50%;

}

**设置图片大小：**

由于样式没有对图片做大小上的样式限制，所以在实际使用的时候，需要通过其他的方式来处理图片大小。比如说控制图片容器大小。（注意不可以通过css样式直接修改img图片的大小，这样操作就不响应了）

**注意：**

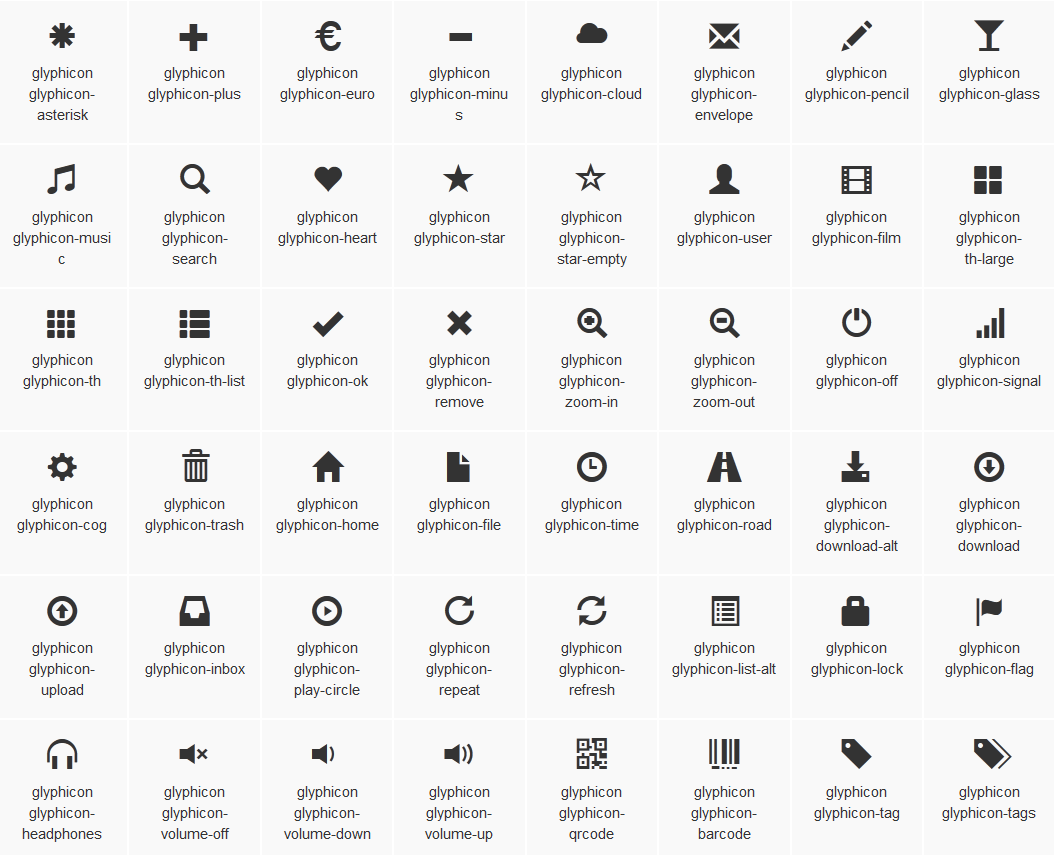
对于圆角图片和圆形图片效果，因为是使用了CSS3的圆角样式来实现的，所以注意对于IE8以及其以下版本不支持，是没有圆角效果的。

Bootstrap框架为了大家更好的维护图像的样式，同样将这部分样式独立出来：  
1、LESS版本，可以查阅scaffolding.less  
2、Sass版本，可以查阅\_scaffolding.scss  
大家可以修改其中之一，重新编译就可以得到自己需要的图片样式效果。

## 图标（一）

**200个icon**：

这里说的图标就是Web制作中常看到的小icon图标，可以说这些小icon图标是一个优秀Web中不可缺少的一部分，起到画龙点睛的效果。在Bootstrap框架中也为大家提供了近**200**个不同的icon图片，而这些图标都是使用**CSS3**的@font-face属性配合字体来实现的icon效果。

[](http://img.mukewang.com/53db0e5b0001aff810560855.jpg)

**放心使用**：

在具体介绍Bootstrap的icon图标之前，我们首先要感谢**glyphicons.com**网站，因为Bootstrap框架中图标都是glyphicons.com这个商业网站提供的，并且免费授权给Bootstrap框架使用，所以大家可以放心使用在自己的项目当中。

Bootstrap框架将内部样式也提取出来：  
1、LESS版本：对应源文件为glyphicons.less文件  
2、Sass版本：对应源文件为\_glyphicons.scss文件  
3、编译后的Bootstrap版本：查看bootstrap.css文件第2375行～第2992行

**原理分析**：

Bootstrap框架中的图标都是字体图标，其实现原理就是通过**@font-face**属性加载了字体。  
/\*源码请查看bootstrap.css文件第2357行～第2380行\*/

@font-face {

font-family: 'Glyphicons Halflings';

src: url('../fonts/glyphicons-halflings-regular.eot');

src: url('../fonts/glyphicons-halflings-regular.eot?#iefix') format('embedded-opentype'), url('../fonts/glyphicons-halflings-regular.woff') format('woff'), url('../fonts/glyphicons-halflings-regular.ttf') format('truetype'), url('../fonts/glyphicons-halflings-regular.svg#glyphicons\_halflingsregular') format('svg');

}

大家或许会问，这些字体我去哪里获取。如果你是从前面一直阅读过来，我们在介绍文件结构那一节就已介绍过。在Bootstrap框架中有一个**fonts**的目录，这个目录中提供的字体文件就是用于制作icon的字体文件。  
自定义完字体之后，需要对**icon**设置一个默认样式，在Bootstrap框架中是通过给元素添加**“glyphicon”**类名来实现，然后通过伪元素“**:before**”的“**content**”属性调取对应的icon编码：  
/\*源码请查看bootstrap.css文件第2381行～第2992行\*/

.glyphicon {

position: relative;

top: 1px;

display: inline-block;

font-family: 'Glyphicons Halflings';

font-style: normal;

font-weight: normal;

line-height: 1;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

.glyphicon-asterisk:before {

content: "\2a";

}

## 图标（二）

在网页中使用图标也非常的简单，在任何内联元素上应用所对应的样式即可：

<span class="glyphicon glyphicon-search"></span>

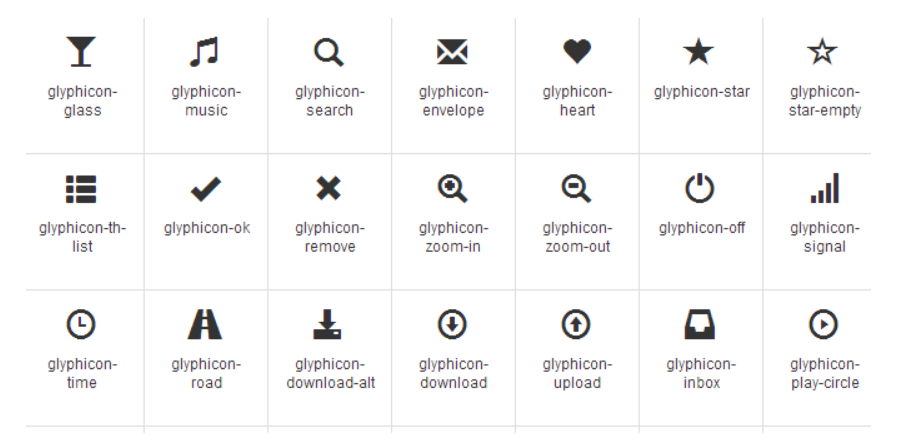
<span class="glyphicon glyphicon-asterisk"></span>

<span class="glyphicon glyphicon-plus"></span>

<span class="glyphicon glyphicon-cloud"></span>

运行效果如下或查看右侧结果窗口：[](http://img.mukewang.com/53b371270001aff102680096.jpg)

所有icon都是以”glyphicon-”前缀的类名开始，然后后缀表示图标的名称。具体说明如下：

[](http://img.mukewang.com/53da0a7e0001300509180435.jpg)

**所有名称查看：**

请点击：[**http://getbootstrap.com/components/#glyphicons**](http://getbootstrap.com/components/#glyphicons) 链接查阅或根据bootstrap.css文件第2393行～第2992行查阅。

**特别说明**：

除了使用glyphicon.com提供的图标之外，还可以使用第三方为Bootstrap框架设计的图标字体，如Font Awesome(http://www.bootcss.com/p/font-awesome/)。使用方法和上面介绍的一样，不过记得将字体下载到你本地。 感兴趣的可以阅读官网相关介绍。

[](http://img.mukewang.com/53db07d100018d5010500434.jpg)

## 网格工作原理

Bootstrap框架的网格系统工作原理如下：

1、数据行(.row)必须包含在容器（.container）中，以便为其赋予合适的对齐方式和内距(padding)。如：

<div class="container">

<div class="row"></div>

</div>

2、在行(.row)中可以添加列(.column)，但列数之和不能超过平分的总列数，比如12。如：

<div class="container">

<div class="row">

<div class="col-md-4"></div>

<div class="col-md-8"></div>

3、具体内容应当放置在列容器（column）之内，而且只有列（column）才可以作为行容器(.row)的直接子元素

4、通过设置内距（padding）从而创建列与列之间的间距。然后通过为第一列和最后一列设置负值的外距（margin）来抵消内距(padding)的影响

为了更好的理解Bootstrap框架的网格系统工作原理，我们来看一张草图：

[](http://img.mukewang.com/53b0f9c000018b9305540282.jpg)

简单对图解释一下：

1、最外边框，带有一大片白色区域，就是相当于浏览器的可视区域。在Bootstrap框架的网格系统中带有响应式效果，其带有四种类型的浏览器（超小屏，小屏，中屏和大屏），其断点（像素的分界点）是768px、992px和1220px。

2、第二个边框(1)相当于容器(.container)。针对不同的浏览器分辨率，其宽度也不一样：自动、750px、970px和1170px。在bootstrap.css的第736行～第756行进行设置：

.container {

padding-right: 15px;

padding-left: 15px;

margin-right: auto;

margin-left: auto;

@media (min-width: 768px) {

.container {

width: 750px;

}

@media (min-width: 992px) {

.container {

width: 970px;

}

@media (min-width: 1200px) {

.container {

width: 1170px;

}

3、２号横条阐述的是，将容器的行（.row）平分了12等份，也就是列。每个列都有一个“padding-left:15px”(图中粉红色部分)和一个“padding-right:15px”(图中紫色部分)。这样也导致了第一个列的padding-left和最后一列的padding-right占据了总宽度的30px，从而致使页面不美观，当然，如果你需要留有一定的间距，这个做法是不错的。如bootstrap.css中第767行~第772行所示：

.col-xs-1, .col-sm-1, .col-md-1, .col-lg-1, .col-xs-2, .col-sm-2, .col-md-2, .col-lg-2, .col-xs-3, .col-sm-3, .col-md-3, .col-lg-3, .col-xs-4, .col-sm-4, .col-md-4, .col-lg-4, .col-xs-5, .col-sm-5, .col-md-5, .col-lg-5, .col-xs-6, .col-sm-6, .col-md-6, .col-lg-6, .col-xs-7, .col-sm-7, .col-md-7, .col-lg-7, .col-xs-8, .col-sm-8, .col-md-8, .col-lg-8, .col-xs-9, .col-sm-9, .col-md-9, .col-lg-9, .col-xs-10, .col-sm-10, .col-md-10, .col-lg-10, .col-xs-11, .col-sm-11, .col-md-11, .col-lg-11, .col-xs-12, .col-sm-12, .col-md-12, .col-lg-12 {

position: relative;

min-height: 1px;

padding-right: 15px;

padding-left: 15px;

4、３号横条就是行容器(.row),其定义了“margin-left”和”margin-right”值为”-15px”，用来抵消第一个列的左内距和最后一列的右内距。在bootstrap.css的第763行~第767行可以看到：

.row {

margin-right: -15px;

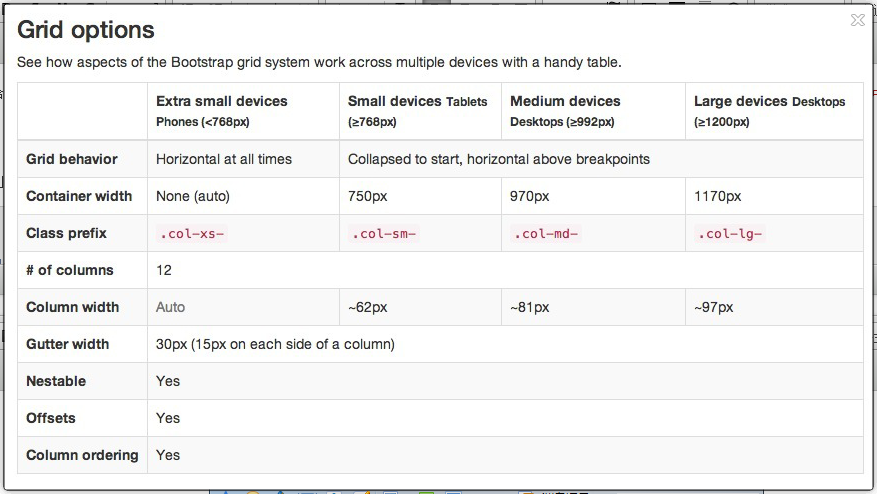
margin-left: -15px;

5、将行与列给合在一起就能看到横条4的效果。也就是我们期望看到的效果，第一列和最后一列与容器（.container）之间没有间距。

横条５只是想向大家展示，你可以根据需要，任意组合列与列，只是他们的组合数之和不要超过总列数。

## 基本用法

网格系统用来布局，其实就是列的组合。Bootstrap框架的网格系统中有四种基本的用法。由于Bootstrap框架在不同屏幕尺寸使用了不同的网格样式，在这一节中所涉及到的示例，我们都以中屏（970px）为例进行介绍，其他屏幕的使用也类似这一种。关于屏幕尺寸如下图：

[](http://img.mukewang.com/53e483500001c7f408770494.jpg)

**1、列组合**

列组合简单理解就是更改数字来合并列（原则：列总和数不能超12），有点类似于表格的colspan属性，例如：

<div class="container">

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-8">.col-md-8</div>

</div>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4">.col-md-4</div>

</div>

<div class="row">

<div class="col-md-3">.col-md-3</div>

<div class="col-md-6">.col-md-6</div>

<div class="col-md-3">.col-md-3</div>

</div>

</div>

使用上面的结构，你将看到下图的效果：

[http://img.mukewang.com/53b0fbdc00015f2805540051.jpg](http://img.mukewang.com/53b0fbdc00015f2805540051.jpg)

(在右侧结果窗口中查看时需要设置为全屏)

实现列组合方式非常简单，只涉及两个CSS两个特性：**浮动**与**宽度百分比**。在bootstrap.css文件的第1088行～1126行：

/\*确保所有列左浮动\*/

.col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8, .col-md-9, .col-md-10, .col-md-11, .col-md-12 {

float: left;

}

/\*定义每个列组合的宽度（使用的百分比）\*/

.col-md-12 {

width: 100%;

}

.col-md-11 {

width: 91.66666667%;

}

.col-md-10 {

width: 83.33333333%;

}

.col-md-9 {

width: 75%;

}

.col-md-8 {

width: 66.66666667%;

}

.col-md-7 {

width: 58.33333333%;

}

.col-md-6 {

width: 50%;

}

.col-md-5 {

width: 41.66666667%;

}

.col-md-4 {

width: 33.33333333%;

}

.col-md-3 {

width: 25%;

}

.col-md-2 {

width: 16.66666667%;

}

.col-md-1 {

width: 8.33333333%;

}

## 列偏移

有的时候，我们不希望相邻的两个列紧靠在一起，但又不想使用margin或者其他的技术手段来。这个时候就可以使用列偏移（offset）功能来实现。使用列偏移也非常简单，只需要在列元素上添加类名“col-md-offset-\*”(其中星号代表要偏移的列组合数)，那么具有这个类名的列就会向右偏移。例如，你在列元素上添加“col-md-offset-4”，表示该列向右移动4个列的宽度。

<div class="container">

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-2 **col-md-offset-4**">列向右移动四列的间距</div>

<div class="col-md-2">.col-md-3</div>

</div>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4 **col-md-offset-4**">列向右移动四列的间距</div>

</div>

</div>

如上面的示例代码，得到的效果如下或在最右侧结果窗口中可以看到(**鼠标移到结果窗口，单击出现的全屏按钮**):

[http://img.mukewang.com/53b0fe8d00018ca605530045.jpg](http://img.mukewang.com/53b0fe8d00018ca605530045.jpg)

实现原理非常简单，就是利用十二分之一（1/12）的margin-left。然后有多少个offset，就有多少个margin-left。在bootstrap.css中第1205行～1241行所示：

.col-md-offset-12 {

margin-left: 100%;

}

.col-md-offset-11 {

margin-left: 91.66666667%;

}

.col-md-offset-10 {

margin-left: 83.33333333%;

}

.col-md-offset-9 {

margin-left: 75%;

}

.col-md-offset-8 {

margin-left: 66.66666667%;

}

.col-md-offset-7 {

margin-left: 58.33333333%;

}

.col-md-offset-6 {

margin-left: 50%;

}

.col-md-offset-5 {

margin-left: 41.66666667%;

}

.col-md-offset-4 {

margin-left: 33.33333333%;

}

.col-md-offset-3 {

margin-left: 25%;

}

.col-md-offset-2 {

margin-left: 16.66666667%;

}

.col-md-offset-1 {

margin-left: 8.33333333%;

}

.col-md-offset-0 {

margin-left: 0;

}

注意：

不过有一个细节需要注意，使用”col-md-offset-\*”对列进行向右偏移时，要保证列与偏移列的总数不超过12，不然会致列断行显示，如：

<div class="row">

<div class="col-md-3">.col-md-3</div>

<div class="col-md-3 **col-md-offset-3**">col-md-offset-3</div>

<div class="col-md-4">col-md-4</div>

</div>

上面代码中列和偏移列总数为3+3+3+4 = 13>12，所以发生了列断行。

如上面的示例代码，得到的效果如下或在最右侧结果窗口中可以看到(**鼠标移到结果窗口，单击出现的全屏按钮**):

[http://img.mukewang.com/53b0ff3f00015e2f05530050.jpg](http://img.mukewang.com/53b0ff3f00015e2f05530050.jpg)

## 列排序

列排序其实就是改变列的方向，就是改变左右浮动，并且设置浮动的距离。在Bootstrap框架的网格系统中是通过添加类名“col-md-push-\*”和“col-md-pull-\*” (其中星号代表移动的列组合数)。

我们来看一个简单的示例：

<div class="container">

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-8">.col-md-8</div>

</div>

</div>

默认情况之下，上面的代码效果如下：

[http://img.mukewang.com/53b10a400001af8005540037.jpg](http://img.mukewang.com/53b10a400001af8005540037.jpg)

“col-md-4”居左，“col-md-8”居右，如果要互换位置，需要将“col-md-4”向右移动８个列的距离，也就是8个offset ,也就是在“<div class=“col-md-4”>”添加类名“col-md-push-8”，调用其样式。

[http://img.mukewang.com/53b10a79000152b805540042.jpg](http://img.mukewang.com/53b10a79000152b805540042.jpg)

也要将“col-md-8”向左移动４个列的距离，也就是4个offset，在“<div class=”col-md-8”>”上添加类名“col-md-pull-4”：

[http://img.mukewang.com/53b10aa80001687005530038.jpg](http://img.mukewang.com/53b10aa80001687005530038.jpg)

Bootstrap仅通过设置left和right来实现定位效果。在boostrap.css文件的第1127行~第1204行可以看到具体的代码：

.col-md-pull-12 {

right: 100%;

}

.col-md-pull-11 {

right: 91.66666667%;

}

.col-md-pull-10 {

right: 83.33333333%;

}

.col-md-pull-9 {

right: 75%;

}

.col-md-pull-8 {

right: 66.66666667%;

}

.col-md-pull-7 {

right: 58.33333333%;

}

.col-md-pull-6 {

right: 50%;

}

.col-md-pull-5 {

right: 41.66666667%;

}

.col-md-pull-4 {

right: 33.33333333%;

}

.col-md-pull-3 {

right: 25%;

}

.col-md-pull-2 {

right: 16.66666667%;

}

.col-md-pull-1 {

right: 8.33333333%;

}

.col-md-pull-0 {

right: 0;

}

.col-md-push-12 {

left: 100%;

}

.col-md-push-11 {

left: 91.66666667%;

}

.col-md-push-10 {

left: 83.33333333%;

}

.col-md-push-9 {

left: 75%;

}

.col-md-push-8 {

left: 66.66666667%;

}

.col-md-push-7 {

left: 58.33333333%;

}

.col-md-push-6 {

left: 50%;

}

.col-md-push-5 {

left: 41.66666667%;

}

.col-md-push-4 {

left: 33.33333333%;

}

.col-md-push-3 {

left: 25%;

}

.col-md-push-2 {

left: 16.66666667%;

}

.col-md-push-1 {

left: 8.33333333%;

}

.col-md-push-0 {

left: 0;

}

## 列的嵌套

Bootstrap框架的网格系统还支持列的嵌套。你可以在一个列中添加一个或者多个行（row）容器，然后在这个行容器中插入列（像前面介绍的一样使用列）。但在列容器中的行容器（row），宽度为100%时，就是当前外部列的宽度。来看一个简单示例：

<div class="container">

<div class="row">

<div class="col-md-8">

我的里面嵌套了一个网格

<div class="row">

<div class="col-md-6">col-md-6</div>

<div class="col-md-6">col-md-6</div>

</div>

</div>

<div class="col-md-4">col-md-4</div>

</div>

<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-8">

我的里面嵌套了一个网格

<div class="row">

<div class="col-md-4">col-md-4</div>

<div class="col-md-4">col-md-4</div>

<div class="col-md-4">col-md-4</div>

</div>

</div>

</div>

</div>

效果如下：

[](http://img.mukewang.com/53b10c9e0001e28b05540070.jpg)

注意：嵌套的列总数也需要遵循不超过12列。不然会造成末位列换行显示。

## 下拉菜单（基本用法）

小伙伴们注意，在Bootstrap框架中的**下拉菜单组件**是一个独立的组件，根据不同的版本，它对应的文件：

  ☑  LESS版本：对应的源码文件为 **dropdowns.less**

  ☑  Sass版本：对应的源码文件为 **\_dropdowns.scss**

  ☑  编译后的Bootstrap版本：查看bootstrap.css文件第3004行～第3130行

在使用Bootstrap框架的下拉菜单时，必须调用Bootstrap框架提供的bootstrap.js文件。当然，如果你使用的是未编译版本，在js文件夹下你能找到一个名为“dropdown.js”的文件。你也可以调用这个js文件。不过在我们的教程中，我们统一调用压缩好的“bootstrap.min.js”文件：

<script src="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>

特别声明：因为Bootstrap的组件交互效果都是依赖于jQuery库写的插件，所以在使用bootstrap.min.js之前一定要先加载jquery.min.js才会生效果。

我们先来看官网上一个简单的示例：

<div class="**dropdown**">

<**button** class="btn btn-default **dropdown-toggle**" type="button" id="dropdownMenu1" **data-toggle="dropdown"**>

下拉菜单

<span class="caret"></span>

</**button**>

<**ul** class="**dropdown-menu**" role="menu" aria-labelledby="dropdownMenu1">

<li role="presentation"><a role="menuitem" tabindex="-1" href="#">下拉菜单项</a></li>

…

<li role="presentation" class="divider"></li>

<li role="presentation"><a role="menuitem" tabindex="-1" href="#">下拉菜单项</a></li>

</**ul**>

</div>

**使用方法：**

在使用Bootstrap框架中的下拉菜单组件时，其结构运用的正确与否非常的重要，如果结构和类名未使用正确，直接影响组件是否能正常运用。我们来简单的看看：

1、使用一个名为“**dropdown**”的容器包裹了整个下拉菜单元素，示例中为:

<div class="dropdown"></div>

2、使用了一个**<button>**按钮做为父菜单，并且定义类名“**dropdown-toggle**”和自定义“**data-toggle**”属性，且值必须和最外容器类名一致，此示例为:

**data-toggle="dropdown"**

3、下拉菜单项使用一个**ul**列表，并且定义一个类名为“**dropdown-menu**”，此示例为:

<ul class="**dropdown-menu**">

## 下拉菜单（原理分析）

Bootstrap框架中的下拉菜单组件，其下拉菜单项默认是隐藏的，如下所示：

[](http://img.mukewang.com/53e1f1850001230803900164.jpg)

因为“dropdown-menu”默认样式设置了“display:none”，其详细源码请查看bootstrap.css文件第3010行～第3029行：

.dropdown-menu {

position: absolute;/\*设置绝对定位，相对于父元素div.dropdown\*/

top: 100%;/\*让下拉菜单项在父菜单项底部，如果父元素不设置相对定位，该元素相对于body元素\*/

left: 0;

z-index: 1000;/\*让下拉菜单项不被其他元素遮盖住\*/

display: none;/\*默认隐藏下拉菜单项\*/

float: left;

min-width: 160px;

padding: 5px 0;

margin: 2px 0 0;

font-size: 14px;

list-style: none;

background-color: #fff;

background-clip: padding-box;

border: 1px solid #ccc;

border: 1px solid rgba(0, 0, 0, .15);

border-radius: 4px;

-webkit-box-shadow: 0 6px 12px rgba(0, 0, 0, .175);

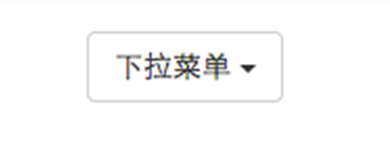
box-shadow: 0 6px 12px rgba(0, 0, 0, .175);

}

当用户点击父菜单项时，下拉菜单将会被显示出来，如下所示：

[](http://img.mukewang.com/53e1f3dc0001c10204150412.jpg)

当用户再次点击时，下拉菜单将继续隐藏，如下所示：

[](http://img.mukewang.com/53e1f1850001230803900164.jpg)

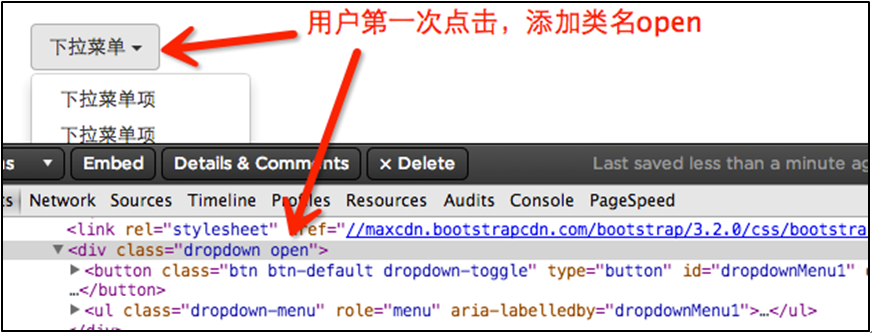
**原理分析：**

现在我们来分析一下实现原理，非常简单，通过js技术手段，给父容器“div.dropdown”添加或移除类名“open”来控制下拉菜单显示或隐藏。也就是说，默认情况，“div.dropdown”没有类名“open”，当用户第一次点击时，“div.dropdown”会添加类名“open”；当用户再次点击时，“div.dropdown”容器中的类名“open”又会被移除。我们可以通过浏览器的firebug查看整个过程：

默认情况：

[](http://img.mukewang.com/53e314020001537208700352.jpg)

用户第一次点击：

[](http://img.mukewang.com/53e314360001a81808720333.jpg)

用户再次点击：

[](http://img.mukewang.com/53e31461000180e608710333.jpg)

在bootstrap.css文件第3076行～第3078行，我们可以很容易发现：

.open > .dropdown-menu {

display: block;

}

## 下拉菜单（下拉分隔线）

在Bootstrap框架中的下拉菜单还提供了下拉分隔线，假设下拉菜单有两个组，那么组与组之间可以通过添加一个空的<li>，并且给这个<li>添加类名“divider”来实现添加下拉分隔线的功能。对应的样式代码：

/\*源码bootstrap.css文件第3034行～第3039行\*/

**.dropdown-menu .divider {**

**height: 1px;**

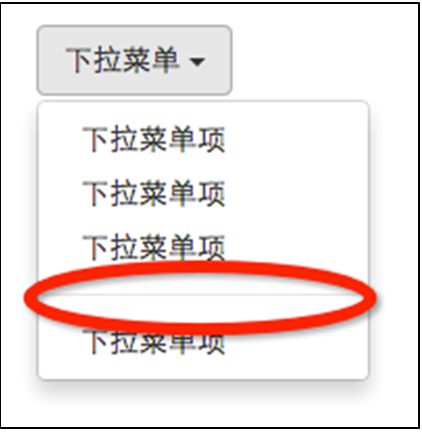
**margin: 9px 0;**

**overflow: hidden;**

**background-color: #e5e5e5;**

**}**

效果如下：

[](http://img.mukewang.com/53e346260001aed304220432.jpg)

## 下拉菜单（菜单标题）

上一小节讲解通过添加“divider”可以将下拉菜单分组，为了让这个分组更明显，还可以给每个组添加一个头部（标题）。如下：

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu1" data-toggle="dropdown">

下拉菜单

<span class="caret"></span>

</button>

<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1">

<li role="presentation" **class="dropdown-header"**>第一部分菜单头部</li>

<li role="presentation"><a role="menuitem" tabindex="-1" href="#">下拉菜单项</a></li>

…

<li role="presentation" class="divider"></li>

<li role="presentation" **class="dropdown-header"**>第二部分菜单头部</li>

…

<li role="presentation"><a role="menuitem" tabindex="-1" href="#">下拉菜单项</a></li>

</ul>

</div>

对应的样式如下：

/\*查看bootstrap.css文件第3090行～第3096行\*/

.dropdown-header {

display: block;

padding: 3px 20px;

font-size: 12px;

line-height: 1.42857143;

color: #999;

}

运行效果如下：

[](http://img.mukewang.com/53e34b1e0001ccdd07440651.jpg)

## 下拉菜单（对齐方式）

**实现右对齐方法：**

Bootstrap框架中下拉菜单默认是左对齐，如果你想让下拉菜单相对于父容器**右对齐**时，可以在“dropdown-menu”上添加一个“pull-right”或者“dropdown-menu-right”类名，如下所示：

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu1" data-toggle="dropdown">

下拉菜单

<span class="caret"></span>

</button>

<ul class="dropdown-menu **pull-right**" role="menu" aria-labelledby="dropdownMenu1">

…

</ul>

</div>

上面代码中的“pull-right”类可以用“dropdown-menu-right”代替，两个类的作用是一样的，可从下面的源代码中看出。

**实现原理：**

对应的样式如下：

/\*源码请查看bootstrap.css文件第3030行～第3033行和3082行～第3085行\*/

.dropdown-menu.pull-right {

right: 0;

left: auto;

}

.dropdown-menu-right {

right: 0;

left: auto;

}

同时一定要为.dropdown添加float:leftcss样式。

.dropdown{

float: left;

}

运行效果如下所示：

[](http://img.mukewang.com/53e34c370001522204970469.jpg)

**下拉菜单与父容器左边对齐:**

与此同时，还有一个类名刚好与“dropdown-menu-right”相反的类名“dropdown-menu-left”，其效果就是让下拉菜单与父容器左边对齐，其实就是默认效果。

/\*请查看bootstrap.css文件第3086行～第3089行\*/

.dropdown-menu-left {

right: auto;

left: 0;

}

## 下拉菜单（菜单项状态）

下拉菜单项的默认的状态（不用设置）有悬浮状态（:hover）和焦点状态（:focus）：

/\*查看bootstrap.css文件第3049行～第3054行\*/

.dropdown-menu > li > a:hover,

.dropdown-menu > li > a:focus {

color: #262626;

text-decoration: none;

background-color: #f5f5f5;

}

下拉菜单项除了上面两种状态，还有当前状态（.active）和禁用状态（.disabled）。这两种状态使用方法只需要在对应的菜单项上添加对应的类名：

<div class="dropdown">

<button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu1" data-toggle="dropdown">

下拉菜单

<span class="caret"></span>

</button>

<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1">

<li role="presentation" **class="active"**><a role="menuitem" tabindex="-1" href="#">下拉菜单项</a></li>

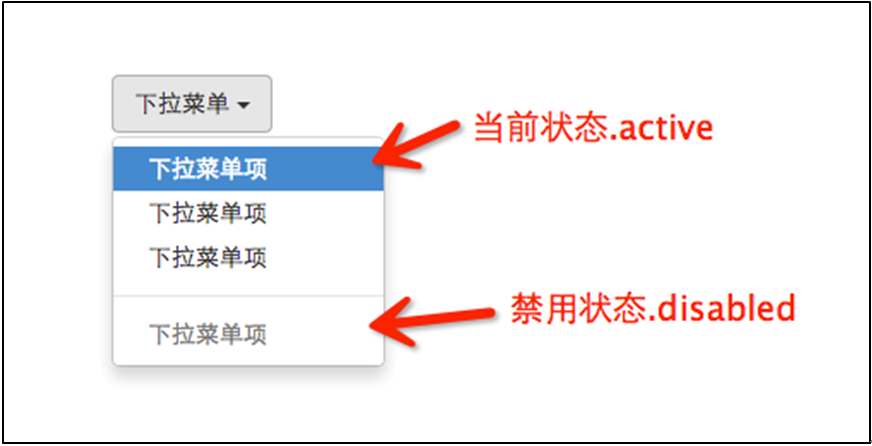
….

<li role="presentation" **class="disabled"**><a role="menuitem" tabindex="-1" href="#">下拉菜单项</a></li>

</ul>

</div>

运行效果如下：

[](http://img.mukewang.com/53e44d0d000131d208720446.jpg)

对应的样式代码也非常简单：

/\*请查看bootstrap.css文件第3055行～第3075行\*/

.dropdown-menu > .active > a,

.dropdown-menu > .active > a:hover,

.dropdown-menu > .active > a:focus {

color: #fff;

text-decoration: none;

background-color: #428bca;

outline: 0;

}

.dropdown-menu > .disabled > a,

.dropdown-menu > .disabled > a:hover,

.dropdown-menu > .disabled > a:focus {

color: #999;

}

.dropdown-menu > .disabled > a:hover,

.dropdown-menu > .disabled > a:focus {

text-decoration: none;

cursor: not-allowed;

background-color: transparent;

background-image: none;

filter: progid:DXImageTransform.Microsoft.gradient(enabled = false);

}

## 按钮（按钮组）

单个按钮在Web页面中的运用有时候并不能满足我们的业务需求，常常会看到将多个按钮组合在一起使用，比如富文本编辑器里的一组小图标按钮等。那么在这一节中，我们主要向大家介绍Bootstrap框架为大家提供的按钮组组件。

**源码查询：**

按钮组也是一个独立的组件，所以可以找到对应的源码文件：

  ☑  LESS版本：对应的源文件为buttons.less

  ☑  Sass版本：对应的源文件为\_buttons.scss

  ☑  CSS版本：对应bootstrap.css文件第3131行～第3291行

**使用方法：**

按钮组和下拉菜单组件一样，需要依赖于button.js插件才能正常运行。不过我们同样可以直接只调用bootstrap.js文件。因为这个文件已集成了button.js插件功能。

对于结构方面，非常的简单。使用一个名为“btn-group”的容器，把多个按钮放到这个容器中。如下所示：

<div **class="btn-group"**>

<button type="button" class="**btn** btn-default">

<span class="glyphicon glyphicon-step-backward"></span>

</button>

…

<button type="button" class="**btn** btn-default">

<span class="glyphicon glyphicon-step-forward"></span>

</button>

</div>

运行效果如下所示：

[](http://img.mukewang.com/53e458d10001af8808310131.jpg)

除了可以使用<button>元素之外，还可以使用其他标签元素，比如<a>标签。唯一要保证的是：不管使用什么标签，“.btn-group”容器里的标签元素需要带有类名“.btn”。

按钮组实现源码如下：

/\*查看bootstrap.css文件第3131行～第3161行\*/

.btn-group,

.btn-group-vertical {

position: relative;

display: inline-block;

vertical-align: middle;

}

.btn-group > .btn,

.btn-group-vertical > .btn {

position: relative;

float: left;

}

.btn-group > .btn:hover,

.btn-group-vertical > .btn:hover,

.btn-group > .btn:focus,

.btn-group-vertical > .btn:focus,

.btn-group > .btn:active,

.btn-group-vertical > .btn:active,

.btn-group > .btn.active,

.btn-group-vertical > .btn.active {

z-index: 2;

}

.btn-group > .btn:focus,

.btn-group-vertical > .btn:focus {

outline: none;

}

.btn-group .btn + .btn,

.btn-group .btn + .btn-group,

.btn-group .btn-group + .btn,

.btn-group .btn-group + .btn-group {

margin-left: -1px;

}

从效果图上我们可以看出，按钮组四个角都是圆角（支持CSS3的浏览器），但有的小伙伴会问，我们平常制作网页时每个按钮都是带有圆角，而在按钮组中的按钮，除了第一个和最后一个具有边上的圆角之外，其他的按钮没有圆角，它是怎么实现的呢？其实实现方法非常简单：

  1、默认所有按钮都有圆角

  2、除第一个按钮和最后一个按钮（下拉按钮除外），其他的按钮都取消圆角效果

  3、第一个按钮只留左上角和左下角是圆角

  4、最后一个按钮只留右上角和右下角是圆角

对应的源码如下：

/\*查看bootstrap.css文件第3174行～第3203行\*/

.btn-group > .btn:not(:first-child):not(:last-child):not(.dropdown-toggle) {

border-radius: 0;

}

.btn-group > .btn:first-child {

margin-left: 0;

}

.btn-group > .btn:first-child:not(:last-child):not(.dropdown-toggle) {

border-top-right-radius: 0;

border-bottom-right-radius: 0;

}

.btn-group > .btn:last-child:not(:first-child),

.btn-group > .dropdown-toggle:not(:first-child) {

border-top-left-radius: 0;

border-bottom-left-radius: 0;

}

.btn-group > .btn-group {

float: left;

}

.btn-group > .btn-group:not(:first-child):not(:last-child) > .btn {

border-radius: 0;

}

.btn-group > .btn-group:first-child> .btn:last-child,

.btn-group > .btn-group:first-child> .dropdown-toggle {

border-top-right-radius: 0;

border-bottom-right-radius: 0;

}

.btn-group > .btn-group:last-child> .btn:first-child {

border-top-left-radius: 0;

border-bottom-left-radius: 0;

}