

# 全局 CSS 样式

设置全局 CSS 样式；基本的 HTML 元素均可以通过 class 设置样式并得到增强效果；还有先进的栅格系统。

## 概览

深入了解 Bootstrap 底层结构的关键部分，包括我们让 web 开发变得更好、更快、更强壮的最佳实践。

## HTML5 文档类型

Bootstrap 使用到的某些 HTML 元素和 CSS 属性需要将页面设置为 HTML5 文档类型。在你项目中的每个页面都要参照下面的格式进行设置。

```
<!DOCTYPE html>
<html lang="zh-CN">
  ...
</html>
```

## 移动设备优先

在 Bootstrap 2 中，我们对框架中的某些关键部分增加了对移动设备友好的样式。而在 Bootstrap 3 中，我们重写了整个框架，使其一开始就是对移动设备友好的。这次不是简单的增加一些可选的针对移动设备的样式，而是直接融合进了框架的内核中。也就是说，**Bootstrap 是移动设备优先的**。针对移动设备的样式融合进了框架的每个角落，而不是增加一个额外的文件。

为了确保适当的绘制和触屏缩放，需要在 `<head>` 之中添加 **viewport** 元数据标签。

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

在移动设备浏览器上，通过为视口（viewport）设置 meta 属性为 `user-scalable=no` 可以禁用其缩放（zooming）功能。这样禁用缩放功能后，用户只能滚动屏幕，就能让你的网站看上去更像原生应用的感觉。注意，这种方式我们并不推荐所有网站使用，还是要看你自己的情况而定！

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

# 排版与链接

Bootstrap 排版、链接样式设置了基本的全局样式。分别是：

- 为 body 元素设置 background-color: #fff;
- 使用 @font-family-base 、 @font-size-base 和 @line-height-base 变量作为排版的基本参数
- 为所有链接设置了基本颜色 @link-color ，并且当链接处于 :hover 状态时才添加下划线

这些样式都能在 scaffolding.less 文件中找到对应的源码。

## Normalize.css

为了增强跨浏览器表现的一致性，我们使用了 Normalize.css，这是由 Nicolas Gallagher 和 Jonathan Neal 维护的一个CSS 重置样式库。

## 布局容器

Bootstrap 需要为页面内容和栅格系统包裹一个 .container 容器。我们提供了两个作此用途的类。注意，由于 padding 等属性的原因，这两种 容器类不能互相嵌套。

.container 类用于固定宽度并支持响应式布局的容器。

```
<div class="container">
  ...
</div>
```

.container-fluid 类用于 100% 宽度，占据全部视口（viewport）的容器。

```
<div class="container-fluid">
  ...
</div>
```

## 栅格系统

Bootstrap 提供了一套响应式、移动设备优先的流式栅格系统，随着屏幕或视口（viewport）尺寸的增加，系统会自动分为最多12列。它包含了易于使用的预定义类，还有强大的mixin 用于生成更具语义的布局。

# 简介

栅格系统用于通过一系列的行（row）与列（column）的组合来创建页面布局，你的内容就可以放入这些创建好的布局中。下面就介绍一下 Bootstrap 栅格系统的工作原理：

- “行（row）”必须包含在 `.container`（固定宽度）或 `.container-fluid`（100% 宽度）中，以便为其赋予合适的排列（alignment）和内补（padding）。
- 通过“行（row）”在水平方向创建一组“列（column）”。
- 你的内容应当放置于“列（column）”内，并且，只有“列（column）”可以作为行（row）”的直接子元素。
- 类似 `.row` 和 `.col-xs-4` 这种预定义的类，可以用来快速创建栅格布局。Bootstrap 源码中定义的 `mixin` 也可以用来创建语义化的布局。
- 通过为“列（column）”设置 `padding` 属性，从而创建列与列之间的间隔（gutter）。通过为 `.row` 元素设置负值 `margin` 从而抵消掉为 `.container` 元素设置的 `padding`，也就间接为“行（row）”所包含的“列（column）”抵消掉了 `padding`。
- 负值的 `margin`就是下面的示例为什么是向外突出的原因。在栅格列中的内容排成一行。
- 栅格系统中的列是通过指定1到12的值来表示其跨越的范围。例如，三个等宽的列可以使用三个 `.col-xs-4` 来创建。
- 如果一“行（row）”中包含了的“列（column）”大于 12，多余的“列（column）”所在的元素将被作为一个整体另起一行排列。
- 栅格类适用于与屏幕宽度大于或等于分界点大小的设备，并且针对小屏幕设备覆盖栅格类。因此，在元素上应用任何 `.col-md-*` 栅格类适用于与屏幕宽度大于或等于分界点大小的设备，并且针对小屏幕设备覆盖栅格类。因此，在元素上应用任何 `.col-lg-*` 不存在，也影响大屏幕设备。

通过研究后面的实例，可以将这些原理应用到你的代码中。

## 媒体查询

在栅格系统中，我们在 Less 文件中使用以下媒体查询（media query）来创建关键的分界点阈值。

```
/* 超小屏幕（手机，小于 768px） */
/* 没有任何媒体查询相关的代码，因为这在 Bootstrap 中是默认的（还记得 Bootstrap 是移动设备优先的吗？） */

/* 小屏幕（平板，大于等于 768px） */
@media (min-width: @screen-sm-min) { ... }

/* 中等屏幕（桌面显示器，大于等于 992px） */
@media (min-width: @screen-md-min) { ... }

/* 大屏幕（大桌面显示器，大于等于 1200px） */
@media (min-width: @screen-lg-min) { ... }
```

我们偶尔也会在媒体查询代码中包含 `max-width` 从而将 CSS 的影响限制在更小范围的屏幕大小之内。

```
@media (max-width: @screen-xs-max) { ... }
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }
@media (min-width: @screen-lg-min) { ... }
```

# 栅格参数

通过下表可以详细查看 Bootstrap 的栅格系统是如何在多种屏幕设备上工作的。

	超小屏幕 手机 (<768px)	小屏幕 平板 (≥768px)	中等屏幕 桌面显示器 (≥992px)	大屏幕 大桌面显示器 (≥1200px)
栅格系统行为	总是水平排列	开始是堆叠在一起的，当大于这些阈值时将变为水平排列C		
.container 最大宽度	None （自动）	750px	970px	1170px
类前缀	.col-xs-	.col-sm-	.col-md-	.col-lg-
列（column）数	12			
最大列（column）宽	自动	~62px	~81px	~97px
槽（gutter）宽	30px （每列左右均有 15px）			
可嵌套	是			
偏移（Offsets）	是			
列排序	是			

## 实例：从堆叠到水平排列

使用单一的一组 .col-md-\* 栅格类，就可以创建一个基本的栅格系统，在手机和平板设备上一开始是堆叠在一起的（超小屏幕到小屏幕这一范围），在桌面（中等）屏幕设备上变为水平排列。所有“列（column）必须放在 ” .row 内。

.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1

.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-1
.col-md-8
.col-md-4
.col-md-4
.col-md-4
.col-md-4
.col-md-6
.col-md-6

```

<div class="row">
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
</div>
<div class="row">
  <div class="col-md-8">.col-md-8</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-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>

```

## 实例：流式布局容器

将最外面的布局元素 `.container` 修改为 `.container-fluid`，就可以将固定宽度的栅格布局转换为 100% 宽度的布局。

```

<div class="container-fluid">
  <div class="row">
    ...
  </div>
</div>

```

## 实例：移动设备和桌面屏幕

是否不希望在小屏幕设备上所有列都堆叠在一起？那就使用针对超小屏幕和中等屏幕设备所定义的类吧，即 `.col-xs-*` 和 `.col-md-*`。请看下面的实例，研究一下这些是如何工作的。

```
.col-xs-12 .col-md-8
```

```
.col-xs-6 .col-md-4
```

.col-xs-6 .col-md-4	.col-xs-6 .col-md-4
.col-xs-6 .col-md-4	
.col-xs-6	.col-xs-6

```

<!-- Stack the columns on mobile by making one full-width and the other
half-width -->
<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on
desktop -->
<div class="row">
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop -->
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>

```

# 实例：手机、平板、桌面

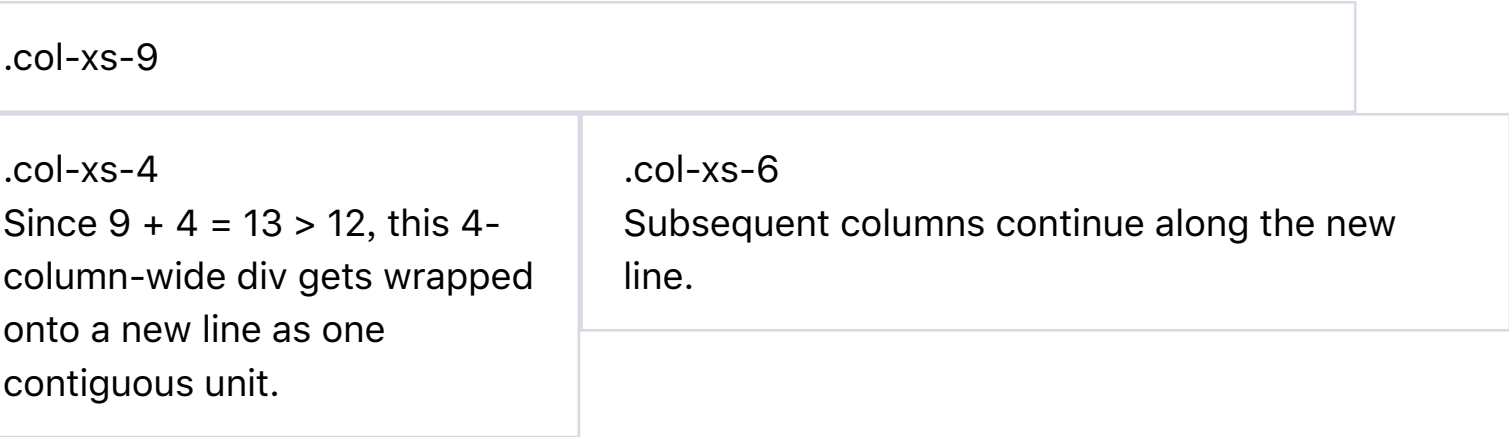
在上面案例的基础上，通过使用针对平板设备的 .col-sm-\* 类，我们来创建更加动态和强大的布局吧。

.col-xs-12 .col-sm-6 .col-md-8	
.col-xs-6 .col-md-4	
.col-xs-6 .col-sm-4	.col-xs-6 .col-sm-4
.col-xs-6 .col-sm-4	

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-8">.col-xs-12 .col-sm-6 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
<div class="row">
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
  <!-- Optional: clear the XS cols if their content doesn't match in height -->
  <div class="clearfix visible-xs-block"></div>
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
</div>
```

## 实例： 多余的列（column） 将另起一行排列

如果在一个 .row 内包含的列（column）大于12个，包含多余列（column）的元素将作为一个整体单元被另起一行排列。



```
<div class="row">
  <div class="col-xs-9">.col-xs-9</div>
  <div class="col-xs-4">.col-xs-4<br>Since 9 + 4 = 13 > 12, this 4-
column-wide div gets wrapped onto a new line as one contiguous unit.</div>
  <div class="col-xs-6">.col-xs-6<br>Subsequent columns continue along the
new line.</div>
</div>
```

## 响应式列重置

即便有上面给出的四组栅格class，你也不免会碰到一些问题，例如，在某些阈值时，某些列可能会出现比别的列高的情况。为了克服这一问题，建议联合使用 .clearfix 和 响应式工具类。

<div>.col-xs-6 .col-sm-3</div> <div>Resize your viewport or check it out on your phone for an example.</div>	<div>.col-xs-6 .col-sm-3</div>
<div>.col-xs-6 .col-sm-3</div>	<div>.col-xs-6 .col-sm-3</div>



```
<div class="row">
  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>

  <!-- Add the extra clearfix for only the required viewport -->
  <div class="clearfix visible-xs-block"></div>

  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
</div>
```

除了列在分界点清除响应， 您可能需要 **重置偏移, 后推或前拉**某个列。请看此栅格实例。

```
<div class="row">
  <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
  <div class="col-sm-5 col-sm-offset-2 col-md-6 col-md-offset-0">.col-sm-5
.col-sm-offset-2 .col-md-6 .col-md-offset-0</div>
</div>

<div class="row">
  <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-
6</div>
  <div class="col-sm-6 col-md-5 col-md-offset-2 col-lg-6 col-lg-offset-
0">.col-sm-6 .col-md-5 .col-md-offset-2 .col-lg-6 .col-lg-offset-0</div>
</div>
```

# 列偏移

使用 `.col-md-offset-*` 类可以将列向右侧偏移。这些类实际是通过使用 `*` 选择器为当前元素增加了左侧的边距（margin）。例如，`.col-md-offset-4` 类将 `.col-md-4` 元素向右侧偏移了4个列（column）的宽度。

.col-md-4
.col-md-4 .col-md-offset-4
.col-md-3 .col-md-offset-3
.col-md-3 .col-md-offset-3
.col-md-6 .col-md-offset-3

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 col-md-offset-4">.col-md-4 .col-md-offset-4</div>
</div>
<div class="row">
  <div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>
  <div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>
</div>
<div class="row">
  <div class="col-md-6 col-md-offset-3">.col-md-6 .col-md-offset-3</div>
</div>
```

You can also override offsets from lower grid tiers with `.col-*-offset-0` classes.

```
<div class="row">
  <div class="col-xs-6 col-sm-4">
  </div>
  <div class="col-xs-6 col-sm-4">
  </div>
  <div class="col-xs-6 col-xs-offset-3 col-sm-4 col-sm-offset-0">
  </div>
</div>
```

## 嵌套列

为了使用内置的栅格系统将内容再次嵌套，可以通过添加一个新的 `.row` 元素和一系列 `.col-sm-*` 元素到已经存在的 `.col-sm-*` 元素内。被嵌套的行（row）所包含的列（column）的个数不能超过12（其实，没有要求你必须占满12列）。

Level 1: .col-sm-9	
Level 2: .col-xs-8 .col-sm-6	Level 2: .col-xs-4 .col-sm-6

```
<div class="row">
  <div class="col-sm-9">
    Level 1: .col-sm-9
    <div class="row">
      <div class="col-xs-8 col-sm-6">
        Level 2: .col-xs-8 .col-sm-6
      </div>
      <div class="col-xs-4 col-sm-6">
        Level 2: .col-xs-4 .col-sm-6
      </div>
    </div>
  </div>
</div>
```

# 列排序

通过使用 `.col-md-push-*` 和 `.col-md-pull-*` 类就可以很容易的改变列（column）的顺序。

```
.col-md-9 .col-md-push-3
```

```
.col-md-3 .col-md-pull-9
```

```
<div class="row">
  <div class="col-md-9 col-md-push-3">.col-md-9 .col-md-push-3</div>
  <div class="col-md-3 col-md-pull-9">.col-md-3 .col-md-pull-9</div>
</div>
```

## Less mixin 和变量

除了用于快速布局的预定义栅格类，Bootstrap 还包含了一组 Less 变量和 mixin 用于帮你生成简单、语义化的布局。

### 变量

通过变量来定义列数、槽（gutter）宽、媒体查询阈值（用于确定合适让列浮动）。我们使用这些变量生成预定义的栅格类，如上所示，还有如下所示的定制 mixin。

```
@grid-columns:          12;
@grid-gutter-width:      30px;
@grid-float-breakpoint:  768px;
```

### mixin

mixin 用来和栅格变量一同使用，为每个列（column）生成语义化的 CSS 代码。

```
// Creates a wrapper for a series of columns
.make-row(@gutter: @grid-gutter-width) {
  // Then clear the floated columns
  .clearfix();

  @media (min-width: @screen-sm-min) {
    margin-left: (@gutter / -2);
    margin-right: (@gutter / -2);
  }

  // Negative margin nested rows out to align the content of columns
  .row {
    margin-left: (@gutter / -2);
    margin-right: (@gutter / -2);
  }
}
```

```
// Generate the extra small columns
.make-xs-column(@columns; @gutter: @grid-gutter-width) {
    position: relative;
    // Prevent columns from collapsing when empty
    min-height: 1px;
    // Inner gutter via padding
    padding-left: (@gutter / 2);
    padding-right: (@gutter / 2);

    // Calculate width based on number of columns available
    @media (min-width: @grid-float-breakpoint) {
        float: left;
        width: percentage((@columns / @grid-columns));
    }
}
```

```
// Generate the small columns
.make-sm-column(@columns; @gutter: @grid-gutter-width) {
    position: relative;
    // Prevent columns from collapsing when empty
    min-height: 1px;
    // Inner gutter via padding
    padding-left: (@gutter / 2);
    padding-right: (@gutter / 2);

    // Calculate width based on number of columns available
    @media (min-width: @screen-sm-min) {
        float: left;
        width: percentage((@columns / @grid-columns));
    }
}
```

```
// Generate the small column offsets
.make-sm-column-offset(@columns) {
    @media (min-width: @screen-sm-min) {
        margin-left: percentage((@columns / @grid-columns));
    }
}

.make-sm-column-push(@columns) {
    @media (min-width: @screen-sm-min) {
        left: percentage((@columns / @grid-columns));
    }
}

.make-sm-column-pull(@columns) {
    @media (min-width: @screen-sm-min) {
        right: percentage((@columns / @grid-columns));
    }
}
```

```
// Generate the medium columns
.make-md-column(@columns; @gutter: @grid-gutter-width) {
    position: relative;
    // Prevent columns from collapsing when empty
    min-height: 1px;
```

```
min-height: 1px;
// Inner gutter via padding
padding-left: (@gutter / 2);
padding-right: (@gutter / 2);

// Calculate width based on number of columns available
@media (min-width: @screen-md-min) {
  float: left;
  width: percentage((@columns / @grid-columns));
}
}

// Generate the medium column offsets
.make-md-column-offset(@columns) {
  @media (min-width: @screen-md-min) {
    margin-left: percentage((@columns / @grid-columns));
  }
}

.make-md-column-push(@columns) {
  @media (min-width: @screen-md-min) {
    left: percentage((@columns / @grid-columns));
  }
}

.make-md-column-pull(@columns) {
  @media (min-width: @screen-md-min) {
    right: percentage((@columns / @grid-columns));
  }
}

// Generate the large columns
.make-lg-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Prevent columns from collapsing when empty
  min-height: 1px;
  // Inner gutter via padding
  padding-left: (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calculate width based on number of columns available
  @media (min-width: @screen-lg-min) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generate the large column offsets
.make-lg-column-offset(@columns) {
  @media (min-width: @screen-lg-min) {
    margin-left: percentage((@columns / @grid-columns));
  }
}

.make-lg-column-push(@columns) {
  @media (min-width: @screen-lg-min) {
    left: percentage((@columns / @grid-columns));
  }
}
```

```
}  
.make-lg-column-pull(@columns) {  
  @media (min-width: @screen-lg-min) {  
    right: percentage((@columns / @grid-columns));  
  }  
}
```

## 实例展示

你可以重新修改这些变量的值，或者用默认值调用这些 mixin。下面就是一个利用默认设置生成两列布局（列之间有间隔）的案例。

```
.wrapper {  
  .make-row();  
}  
.content-main {  
  .make-lg-column(8);  
}  
.content-secondary {  
  .make-lg-column(3);  
  .make-lg-column-offset(1);  
}
```

```
<div class="wrapper">  
  <div class="content-main">...</div>  
  <div class="content-secondary">...</div>  
</div>
```

# 排版

## 标题

HTML 中的所有标题标签，<h1> 到 <h6> 均可使用。另外，还提供了 .h1 到 .h6 类，为的是给内联（inline）属性的文本赋予标题的样式。

实例：

# h1. Bootstrap heading

Semibold 36px

## h2. Bootstrap heading

Semibold 30px

### h3. Bootstrap heading

Semibold 24px

#### h4. Bootstrap heading

Semibold 18px

##### h5. Bootstrap heading

Semibold 14px

###### h6. Bootstrap heading

Semibold 12px

```
<h1>h1. Bootstrap heading</h1>
<h2>h2. Bootstrap heading</h2>
<h3>h3. Bootstrap heading</h3>
<h4>h4. Bootstrap heading</h4>
<h5>h5. Bootstrap heading</h5>
<h6>h6. Bootstrap heading</h6>
```

在标题内还可以包含 `<small>` 标签或赋予 `.small` 类的元素，可以用来标记副标题。

实例：

# h1. Bootstrap heading

Secondary text

## h2. Bootstrap heading

Secondary text

### h3. Bootstrap heading

Secondary text

#### h4. Bootstrap heading

Secondary text

##### h5. Bootstrap heading

Secondary text

###### h6. Bootstrap heading

Secondary text

```
<h1>h1. Bootstrap heading <small>Secondary text</small></h1>
<h2>h2. Bootstrap heading <small>Secondary text</small></h2>
<h3>h3. Bootstrap heading <small>Secondary text</small></h3>
<h4>h4. Bootstrap heading <small>Secondary text</small></h4>
<h5>h5. Bootstrap heading <small>Secondary text</small></h5>
<h6>h6. Bootstrap heading <small>Secondary text</small></h6>
```

# 页面主体

Bootstrap 将全局 `font-size` 设置为 **14px**，`line-height` 设置为 **1.428**。这些属性直接赋予 `<body>` 元素和所有段落元素。另外，`<p>`（段落）元素还被设置了等于 1/2 行高（即 10px）的底部外边距（margin）。

实例：

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

Maecenas sed diam eget risus varius blandit sit amet non magna. Donec id elit non mi porta gravida at eget metus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit.

`<p>...</p>`

## 中心内容

通过添加 `.lead` 类可以让段落突出显示。

实例：

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus.

`<p class="lead">...</p>`

## 使用 Less 工具构建

**variables.less** 文件中定义的两个 Less 变量决定了排版尺寸：`@font-size-base` 和 `@line-height-base`。第一个变量定义了全局 `font-size` 基准，第二个变量是 `line-height` 基准。我们使用这些变量和一些简单的公式计算出其它所有页面元素的 `margin`、`padding` 和 `line-height`。自定义这些变量即可改变 Bootstrap 的默认样式。

## 内联文本元素

### Marked text

For highlighting a run of text due to its relevance in another context, use the `<mark>` tag.



实例：

You can use the mark tag to highlight text.

You can use the mark tag to `<mark>highlight</mark>` text.

## 被删除的文本

对于被删除的文本使用 `<del>` 标签。

实例：

~~This line of text is meant to be treated as deleted text.~~

`<del>`This line of text is meant to be treated as deleted text.`</del>`

## 无用文本

对于没用的文本使用 `<s>` 标签。

实例：

~~This line of text is meant to be treated as no longer accurate.~~

`<s>`This line of text is meant to be treated as no longer accurate.`</s>`

## 插入文本

额外插入的文本使用 `<ins>` 标签。

实例：

This line of text is meant to be treated as an addition to the document.

`<ins>`This line of text is meant to be treated as an addition to the document.`</ins>`

## 带下划线的文本

为文本添加下划线，使用 `<u>` 标签。

实例：

This line of text will render as underlined

`<u>`This line of text will render as underlined`</u>`

利用 HTML 自带的表示强调意味的标签来为文本增添少量样式。

## 小号文本

对于不需要强调的inline或block类型的文本，使用 `<small>` 标签包裹，其内的文本将被设置为父容器字体大小的 85%。标题元素中嵌套的 `<small>` 元素被设置不同的 `font-size` 。

你还可以为行内元素赋予 `.small` 类以代替任何 `<small>` 元素。

实例：

This line of text is meant to be treated as fine print.

```
<small>This line of text is meant to be treated as fine print.</small>
```

## 着重

通过增加 `font-weight` 值强调一段文本。

实例：

The following snippet of text is **rendered as bold text**.

```
<strong>rendered as bold text</strong>
```

## 斜体

用斜体强调一段文本。

实例：

The following snippet of text is *rendered as italicized text*.

```
<em>rendered as italicized text</em>
```

### Alternate elements

在 HTML5 中可以放心使用 `<b>` 和 `<i>` 标签。`<b>` 用于高亮单词或短语，不帶有任何着重的意味；而 `<i>` 标签主要用于发言、技术词汇等。

## 对齐

通过文本对齐类，可以简单方便的将文字重新对齐。

实例：

Left aligned text.

Center aligned text.

Right aligned text.

Justified text.

No wrap text.

```
<p class="text-left">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-right">Right aligned text.</p>
<p class="text-justify">Justified text.</p>
<p class="text-nowrap">No wrap text.</p>
```

# 改变大小写

通过这几个类可以改变文本的大小写。

实例：

lowercased text.

UPPERCASED TEXT.

Capitalized Text.

```
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">Capitalized text.</p>
```

# 缩略语

当鼠标悬停在缩写和缩写词上时就会显示完整内容，Bootstrap 实现了对 HTML 的 <abbr> 元素的增强样式。缩略语元素带有 title 属性，外观表现为带有较浅的虚线框，鼠标移至上面时会变成带有“问号”的指针。如想看完整的内容可把鼠标悬停在缩略语上（对使用辅助技术的用户也可见），但需要包含 title 属性。

## 基本缩略语

实例：

An abbreviation of the word attribute is attr (attribute).

```
<abbr title="attribute">attr</abbr>
```

# 首字母缩略语

为缩略语添加 `.initialism` 类，可以让 font-size 变得稍微小些。

实例：

HTML (HYPERTEXT MARKUP LANGUAGE) is the best thing since sliced bread.

```
<abbr title="HyperText Markup Language" class="initialism">HTML</abbr>
```

# 地址

让联系信息以最接近日常使用的格式呈现。在每行结尾添加 `<br>` 可以保留需要的样式。

实例：

**Twitter, Inc.**  
1355 Market Street, Suite 900  
San Francisco, CA 94103  
P: (Phone) (123) 456-7890

**Full Name**  
first.last@example.com

```
<address>
  <strong>Twitter, Inc.</strong><br>
  1355 Market Street, Suite 900<br>
  San Francisco, CA 94103<br>
  <abbr title="Phone">P:</abbr> (123) 456-7890
</address>
```

```
<address>
  <strong>Full Name</strong><br>
  <a href="mailto:#">first.last@example.com</a>
</address>
```

# 引用

在你的文档中引用其他来源的内容。

## 默认样式的引用

将任何 HTML (HyperText Markup Language) 元素包裹在 `<blockquote>` 中即可表现为引用样式。对于直接引用，我们建议用 `<p>` 标签。

实例：

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

```
<blockquote>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
</blockquote>
```

## 多种引用样式

对于标准样式的 `<blockquote>`，可以通过几个简单的变体就能改变风格和内容。

### 命名来源

添加 `<footer>` 用于标明引用来源。来源的名称可以包裹进 `<cite>` 标签中。

实例：

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

— Someone famous in *Source Title*

```
<blockquote>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
  <footer>Someone famous in <cite title="Source Title">Source Title</cite>
</footer>
</blockquote>
```

### 另一种展示风格

通过赋予 `.blockquote-reverse` 类可以让引用呈现内容右对齐的效果。

实例：

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

Someone famous in *Source Title* —

```
<blockquote class="blockquote-reverse">
  ...
</blockquote>
```

# 列表

## 无序列表

排列顺序无关紧要的一系列元素。

实例：

- Lorem ipsum dolor sit amet
- Consectetur adipiscing elit
- Integer molestie lorem at massa
- Facilisis in pretium nisl aliquet
- Nulla volutpat aliquam velit
  - Phasellus iaculis neque
  - Purus sodales ultricies
  - Vestibulum laoreet porttitor sem
  - Ac tristique libero volutpat at
- Faucibus porta lacus fringilla vel
- Aenean sit amet erat nunc
- Eget porttitor lorem

```
<ul>
  <li>...</li>
</ul>
```

## 有序列表

顺序至关重要的一组元素。

实例：

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. Facilisis in pretium nisl aliquet
5. Nulla volutpat aliquam velit
6. Faucibus porta lacus fringilla vel
7. Aenean sit amet erat nunc
8. Eget porttitor lorem

```
<ol>
  <li>...</li>
</ol>
```

## 无样式列表

移除了默认的 `list-style` 样式和左侧外边距的一组元素（只针对直接子元素）。这是针对直接子元素的，也就是说，你需要对所有嵌套的列表都添加这个类才能具有同样的样式。

实例：

Lorem ipsum dolor sit amet  
Consectetur adipiscing elit  
Integer molestie lorem at massa  
Facilisis in pretium nisl aliquet  
Nulla volutpat aliquam velit

- Phasellus iaculis neque
- Purus sodales ultricies
- Vestibulum laoreet porttitor sem
- Ac tristique libero volutpat at

Faucibus porta lacus fringilla vel  
Aenean sit amet erat nunc  
Eget porttitor lorem

```
<ul class="list-unstyled">  
  <li>...</li>  
</ul>
```

## 内联列表

通过设置 `display: inline-block;` 并添加少量的内补（padding），将所有元素放置于同一行。

实例：

Lorem ipsum   Phasellus iaculis   Nulla volutpat

```
<ul class="list-inline">  
  <li>...</li>  
</ul>
```

## 描述

带有描述的短语列表。

实例：

**Description lists**  
A description list is perfect for defining terms.  
**Euismod**  
Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.  
Donec id elit non mi porta gravida at eget metus.  
**Malesuada porta**  
Etiam porta sem malesuada magna mollis euismod.

```
<dl>
  <dt>...</dt>
  <dd>...</dd>
</dl>
```

## 水平排列的描述

.dl-horizontal 可以让 <dl> 内的短语及其描述排在一行。开始是像 <dl> 的默认样式堆叠在一起，随着导航条逐渐展开而排列在一行。

实例：

### Description lists

A description list is perfect for defining terms.

### Euismod

Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.

Donec id elit non mi porta gravida at eget metus.

### Malesuada porta

Etiam porta sem malesuada magna mollis euismod.

### Felis euismod semper eget lacinia

Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

```
<dl class="dl-horizontal">
  <dt>...</dt>
  <dd>...</dd>
</dl>
```

### 自动截断

通过 text-overflow 属性，水平排列的描述列表将会截断左侧太长的短语。在较窄的视口（viewport）内，列表将变为默认堆叠排列的布局方式。

# 代码

## 内联代码

通过 <code> 标签包裹内联样式的代码片段。



实例：

For example, `<section>` should be wrapped as inline.

For example, `<code>&lt;section&gt;</code>` should be wrapped as inline.

# 用户输入

通过 `<kbd>` 标签标记用户通过键盘输入的内容。

实例：

To switch directories, type `cd` followed by the name of the directory.

To edit settings, press `ctrl + ,`

To switch directories, type `<kbd>cd</kbd>` followed by the name of the directory.  
<br>

To edit settings, press `<kbd><kbd>ctrl</kbd> + <kbd>,</kbd></kbd>`

# 代码块

多行代码可以使用 `<pre>` 标签。为了正确的展示代码，注意将尖括号做转义处理。

实例：

```
<p>Sample text here...</p>
```

```
&lt;p&gt;Sample text here...&lt;/p&gt;</pre>
```

还可以使用 `.pre-scrollable` 类，其作用是设置 `max-height` 为 `350px`，并在垂直方向展示滚动条。

# 变量

通过 `<var>` 标签标记变量。

实例：

$$y = mx + b$$

```
<var>y</var> = <var>m</var><var>x</var> + <var>b</var>
```

# 程序输出

通过 `<samp>` 标签来标记程序输出的内容。

实例：

This text is meant to be treated as sample output from a computer program.

```
<samp>This text is meant to be treated as sample output from a computer
program.</samp>
```

# 表格

## 基本实例

为任意 `<table>` 标签添加 `.table` 类可以为其赋予基本的样式 — 少量的内补（padding）和水平方向的分隔线。这种方式看起来很多余！？但是我们觉得，表格元素使用的很广泛，如果我们为其赋予默认样式可能会影响例如日历和日期选择之类的插件，所以我们选择将此样式独立出来。

实例：

Optional table caption.

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table">
  ...
</table>
```

## 条纹状表格

通过 `.table-striped` 类可以给 `<tbody>` 之内的每一行增加斑马条纹样式。

### 跨浏览器兼容性

条纹状表格是依赖 `:nth-child` CSS 选择器实现的，而这一功能不被 Internet Explorer 8 支持。

实例：

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table table-striped">
  ...
</table>
```

## 带边框的表格

添加 `.table-bordered` 类为表格和其中的每个单元格增加边框。

实例：

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table table-bordered">
  ...
</table>
```

## 鼠标悬停

通过添加 `.table-hover` 类可以让 `<tbody>` 中的每一行对鼠标悬停状态作出响应。

实例：

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

```
<table class="table table-hover">
  ...
</table>
```

## 紧缩表格

通过添加 `.table-condensed` 类可以让表格更加紧凑，单元格中的内补（padding）均会减半。

实例：

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-condensed">
  ...
</table>
```

## 状态类

通过这些状态类可以为行或单元格设置颜色。

Class	描述
.active	鼠标悬停在行或单元格上时所设置的颜色
.success	标识成功或积极的动作
.info	标识普通的提示信息或动作
.warning	标识警告或需要用户注意
.danger	标识危险或潜在的带来负面影响的动作

实例：

#	Column heading	Column heading	Column heading
1	Column content	Column content	Column content
2	Column content	Column content	Column content
3	Column content	Column content	Column content

4	Column content	Column content	Column content
5	Column content	Column content	Column content
6	Column content	Column content	Column content
7	Column content	Column content	Column content
8	Column content	Column content	Column content
9	Column content	Column content	Column content

```
<!-- On rows -->
<tr class="active">...</tr>
<tr class="success">...</tr>
<tr class="warning">...</tr>
<tr class="danger">...</tr>
<tr class="info">...</tr>

<!-- On cells (`td` or `th`) -->
<tr>
  <td class="active">...</td>
  <td class="success">...</td>
  <td class="warning">...</td>
  <td class="danger">...</td>
  <td class="info">...</td>
</tr>
```

### 向使用辅助技术的用户传达用意

通过为表格中的一行或一个单元格添加颜色而赋予不同的意义只是提供了一种视觉上的表现，并不能为使用辅助技术 -- 例如屏幕阅读器 -- 浏览网页的用户提供更多信息。因此，请确保通过颜色而赋予的不同意义可以通过内容本身来表达（即在相应行或单元格中的可见的文本内容）；或者通过包含额外的方式 -- 例如应用了 `.sr-only` 类而隐藏的文本 -- 来表达出来。

## 响应式表格

将任何 `.table` 元素包裹在 `.table-responsive` 元素内，即可创建响应式表格，其会在小屏幕设备上（小于768px）水平滚动。当屏幕大于 768px 宽度时，水平滚动条消失。

### 垂直方向的内容截断

响应式表格使用了 `overflow-y: hidden` 属性，这样就能将超出表格底部和顶部内容截断。特别是，也可以截断下拉菜单和其他第三方组件。

# Firefox 和 fieldset 元素

Firefox 浏览器对 fieldset 元素设置了一些影响 width 属性的样式，导致响应式表格出现问题。可以使用下面提供的针对 Firefox 的 hack 代码解决，但是以下代码并未集成在 Bootstrap 中：

```
@-moz-document url-prefix() {
  fieldset { display: table-cell; }
}
```

更多信息请参考 [this Stack Overflow answer](#).

实例：

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

```
<div class="table-responsive">
  <table class="table">
    ...
  </table>
</div>
```

# 表单

## 基本实例

单独的表单控件会被自动赋予一些全局样式。所有设置了 `.form-control` 类的 `<input>`、`<textarea>` 和 `<select>` 元素都将被默认设置宽度属性为 `width: 100%;`。将 `label` 元素和前面提到的控件包裹在 `.form-group` 中可以获得最好的排列。

实例：

Email address

Email

Password

Password

File input

选取文件 未选择文件

Example block-level help text here.

☐ Check me out

Submit

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
placeholder="Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1"
placeholder="Password">
  </div>
  <div class="form-group">
    <label for="exampleInputFile">File input</label>
    <input type="file" id="exampleInputFile">
    <p class="help-block">Example block-level help text here.</p>
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Check me out
    </label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

不要将表单组和输入框组混合使用

不要将表单组直接和输入框组混合使用。建议将输入框组嵌套到表单组中使用。

# 内联表单

为 `<form>` 元素添加 `.form-inline` 类可使其内容左对齐并且表现为 `inline-block` 级别的控件。只适用于视口（**viewport**）至少在 **768px** 宽度时（视口宽度再小的话就会使表单折叠）。

## 可能需要手动设置宽度

在 Bootstrap 中，输入框和单选/多选框控件默认被设置为 `width: 100%`；宽度。在内联表单，我们将这些元素的宽度设置为 `width: auto;`，因此，多个控件可以排列在同一行。根据你的布局需求，可能需要一些额外的定制化组件。

## 一定要添加 label 标签

如果你没有为每个输入控件设置 `label` 标签，屏幕阅读器将无法正确识别。对于这些内联表单，你可以通过为 `label` 设置 `.sr-only` 类将其隐藏。还有一些辅助技术提供`label`标签的替代方案，比如 `aria-label`、`aria-labelledby` 或 `title` 属性。如果这些都不存在，屏幕阅读器可能会采取使用 `placeholder` 属性，如果存在的话，使用占位符来替代其他的标记，但要注意，这种方法是不妥当的。

实例：

Name

Jane Doe

Email

jane.doe@example.com

Send invitation



```
<form class="form-inline">
  <div class="form-group">
    <label for="exampleInputName2">Name</label>
    <input type="text" class="form-control" id="exampleInputName2"
placeholder="Jane Doe">
  </div>
  <div class="form-group">
    <label for="exampleInputEmail2">Email</label>
    <input type="email" class="form-control" id="exampleInputEmail2"
placeholder="jane.doe@example.com">
  </div>
  <button type="submit" class="btn btn-default">Send invitation</button>
</form>
```

实例：

Email

Password

☐ Remember me

Sign in

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="exampleInputEmail3">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail3"
placeholder="Email">
  </div>
  <div class="form-group">
    <label class="sr-only" for="exampleInputPassword3">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword3"
placeholder="Password">
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Remember me
    </label>
  </div>
  <button type="submit" class="btn btn-default">Sign in</button>
</form>
```

实例：

\$	Amount	.00
----	--------	-----

Transfer cash

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="exampleInputAmount">Amount (in dollars)</label>
    <div class="input-group">
      <div class="input-group-addon">$</div>
      <input type="text" class="form-control" id="exampleInputAmount"
placeholder="Amount">
      <div class="input-group-addon">.00</div>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Transfer cash</button>
</form>
```

## 水平排列的表单

通过为表单添加 `.form-horizontal` 类，并联合使用 Bootstrap 预置的栅格类，可以将 `label` 标签和控件组水平并排布局。这样做将改变 `.form-group` 的行为，使其表现为栅格系统中的行（row），因此就无需再额外添加 `.row` 了。

实例：

Email

Email

Password

Password

☐ Remember me

Sign in

```
<form class="form-horizontal">
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3"
placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword3" class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3"
placeholder="Password">
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Remember me
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">Sign in</button>
    </div>
  </div>
</form>
```

# 被支持的控件

表单布局实例中展示了其所支持的标准表单控件。

## 输入框

包括大部分表单控件、文本输入域控件，还支持所有 HTML5 类型的输入控件：  
text、password、datetime、datetime-local、date、month、time、week、number、email、url、search、tel 和 color。

### 必须添加类型声明

只有正确设置了 type 属性的输入控件才能被赋予正确的样式。

实例：

Text input

```
<input type="text" class="form-control" placeholder="Text input">
```

## 输入控件组

如需在文本输入域 `<input>` 前面或后面添加文本内容或按钮控件，请参考输入控件组。

# 文本域

支持多行文本的表单控件。可根据需要改变 `rows` 属性。

实例：

Textarea

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

# 多选和单选框

多选框（checkbox）用于选择列表中的一个或多个选项，而单选框（radio）用于从多个选项中只选择一个。

Disabled checkboxes and radios are supported, but to provide a "not-allowed" cursor on hover of the parent `<label>`, you'll need to add the `.disabled` class to the parent `.radio`, `.radio-inline`, `.checkbox`, or `.checkbox-inline`.

## 默认外观（堆叠在一起）

实例：

- ☐ Option one is this and that—be sure to include why it's great
- ☐ Option two is disabled
- ☒ Option one is this and that—be sure to include why it's great
- ☐ Option two can be something else and selecting it will deselect option one
- ☐ Option three is disabled

```
<div class="checkbox">
  <label>
    <input type="checkbox" value="">
    Option one is this and that&mdash;be sure to include why it's great
  </label>
</div>
<div class="checkbox disabled">
  <label>
    <input type="checkbox" value="" disabled>
    Option two is disabled
  </label>
</div>

<div class="radio">
  <label>
    <input type="radio" name="optionsRadios" id="optionsRadios1"
value="option1" checked>
    Option one is this and that&mdash;be sure to include why it's great
  </label>
</div>
<div class="radio">
  <label>
    <input type="radio" name="optionsRadios" id="optionsRadios2"
value="option2">
    Option two can be something else and selecting it will deselect option one
  </label>
</div>
<div class="radio disabled">
  <label>
    <input type="radio" name="optionsRadios" id="optionsRadios3"
value="option3" disabled>
    Option three is disabled
  </label>
</div>
```

## 内联单选和多选框

通过将 `.checkbox-inline` 或 `.radio-inline` 类应用到一系列的多选框（checkbox）或单选框（radio）控件上，可以使这些控件排列在一行。

实例：

☐ 1 ☐ 2 ☐ 3

☐ 1 ☐ 2 ☐ 3

```
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox1" value="option1"> 1
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox2" value="option2"> 2
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="inlineCheckbox3" value="option3"> 3
</label>

<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio1"
value="option1"> 1
</label>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio2"
value="option2"> 2
</label>
<label class="radio-inline">
  <input type="radio" name="inlineRadioOptions" id="inlineRadio3"
value="option3"> 3
</label>
```

## 不带label文本的Checkbox 和 radio

如果需要 <label> 内没有文字，输入框（input）正是你所期望的。目前只适用于非内联的 **checkbox** 和 **radio**。请记住，仍然需要为使用辅助技术的用户提供某种形式的 label（例如，使用 aria-label）。

实例：

☐

☐

```
<div class="checkbox">
  <label>
    <input type="checkbox" id="blankCheckbox" value="option1" aria-label="...">
  </label>
</div>
<div class="radio">
  <label>
    <input type="radio" name="blankRadio" id="blankRadio1" value="option1"
aria-label="...">
  </label>
</div>
```

## 下拉列表（select）

注意，很多原生选择菜单 - 即在 Safari 和 Chrome 中 - 的圆角是无法通过修改 `border-radius` 属性来改变的。

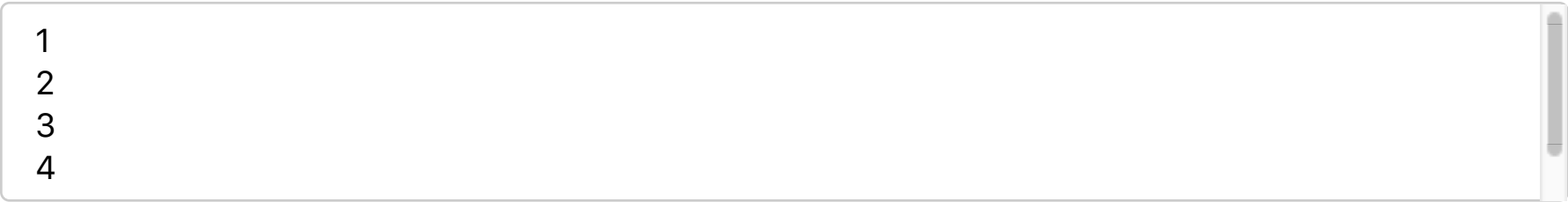
实例：

A single-select dropdown menu with a light gray border and rounded corners. The number '1' is displayed inside the menu, and a small downward-pointing arrow is visible on the right side.

```
<select class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

对于标记了 `multiple` 属性的 `<select>` 控件来说，默认显示多选项。

实例：

A multiple-select dropdown menu with a light gray border and rounded corners. It displays a list of four options: '1', '2', '3', and '4'. A vertical scrollbar is visible on the right side of the menu.

```
<select multiple class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

## 静态控件

如果需要在表单中将一行纯文本和 `label` 元素放置于同一行，为 `<p>` 元素添加 `.form-control-static` 类即可。

实例：

**Email**

email@example.com

**Password**

A password input field with a light gray border and rounded corners. The placeholder text 'Password' is displayed inside the field.

```
<form class="form-horizontal">
  <div class="form-group">
    <label class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
      <p class="form-control-static">email@example.com</p>
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword" class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword"
placeholder="Password">
    </div>
  </div>
</form>
```

实例：

email@example.com

Password

Confirm identity

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only">Email</label>
    <p class="form-control-static">email@example.com</p>
  </div>
  <div class="form-group">
    <label for="inputPassword2" class="sr-only">Password</label>
    <input type="password" class="form-control" id="inputPassword2"
placeholder="Password">
  </div>
  <button type="submit" class="btn btn-default">Confirm identity</button>
</form>
```

## 焦点状态

我们将某些表单控件的默认 outline 样式移除，然后对 :focus 状态赋予 box-shadow 属性。

实例：

Demonstrative focus state



## 演示 :focus 状态

在本文档中，我们为上面实例中的输入框赋予了自定义的样式，用于演示 `.form-control` 元素的 `:focus` 状态。

# 禁用状态

为输入框设置 `disabled` 属性可以禁止其与用户有任何交互（焦点、输入等）。被禁用的输入框颜色更浅，并且还添加了 `not-allowed` 鼠标状态。

实例：

Disabled input here...

```
<input class="form-control" id="disabledInput" type="text"
placeholder="Disabled input here..." disabled>
```

## 被禁用的 fieldset

为 `<fieldset>` 设置 `disabled` 属性,可以禁用 `<fieldset>` 中包含的所有控件。

### <a> 标签的链接功能不受影响

默认情况下，浏览器会将 `<fieldset disabled>` 内所有的原生的表单控件（`<input>`、`<select>` 和 `<button>` 元素）设置为禁用状态，防止键盘和鼠标与他们交互。然而，如果表单中还包含 `<a ... class="btn btn-*">` 元素，这些元素将只被赋予 `pointer-events: none` 属性。正如在关于 禁用状态的按钮 章节中（尤其是关于锚点元素的子章节中）所描述的那样，该 CSS 属性尚不规范，并且在 Opera 18 及更低版本的浏览器或 Internet Explorer 11 总没有得到全面支持，并且不会阻止键盘用户能够获取焦点或激活这些链接。所以为了安全起见，建议使用自定义 JavaScript 来禁用这些链接。

### 跨浏览器兼容性

虽然 Bootstrap 会将这些样式应用到所有浏览器上，Internet Explorer 11 及以下浏览器中的 `<fieldset>` 元素并不完全支持 `disabled` 属性。因此建议在这些浏览器上通过 JavaScript 代码来禁用 `<fieldset>`。

实例：

Disabled input

Disabled input

Disabled input

## Disabled select menu

Disabled select

☐ Can't check this

Submit

```
<form>
  <fieldset disabled>
    <div class="form-group">
      <label for="disabledTextInput">Disabled input</label>
      <input type="text" id="disabledTextInput" class="form-control"
placeholder="Disabled input">
    </div>
    <div class="form-group">
      <label for="disabledSelect">Disabled select menu</label>
      <select id="disabledSelect" class="form-control">
        <option>Disabled select</option>
      </select>
    </div>
    <div class="checkbox">
      <label>
        <input type="checkbox"> Can't check this
      </label>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </fieldset>
</form>
```

## 只读状态

为输入框设置 `readonly` 属性可以禁止用户修改输入框中的内容。处于只读状态的输入框颜色更浅（就像被禁用的输入框一样），但是仍然保留标准的鼠标状态。

实例：

Readonly input here...

```
<input class="form-control" type="text" placeholder="Readonly input here..."
readonly>
```

## Help text

Block level help text for form controls.

## Associating help text with form controls

Help text should be explicitly associated with the form control it relates to using the `aria-describedby` attribute. This will ensure that assistive technologies – such as screen readers – will announce this help text when the user focuses or enters the control.

实例：

### Input with help text

A block of help text that breaks onto a new line and may extend beyond one line.

```
<label class="sr-only" for="inputHelpBlock">Input with help text</label>
<input type="text" id="inputHelpBlock" class="form-control" aria-
describedby="helpBlock">
...
<span id="helpBlock" class="help-block">A block of help text that breaks onto a
new line and may extend beyond one line.</span>
```

## 校验状态

Bootstrap 对表单控件的校验状态，如 `error`、`warning` 和 `success` 状态，都定义了样式。使用时，添加 `.has-warning`、`.has-error` 或 `.has-success` 类到这些控件的父元素即可。任何包含在此元素之内的 `.control-label`、`.form-control` 和 `.help-block` 元素都将接受这些校验状态的样式。

### 将验证状态传达给辅助设备和盲人用户

使用这些校验样式只是为表单控件提供一个可视的、基于色彩的提示，但是并不能将这种提示信息传达给使用辅助设备的用户 - 例如屏幕阅读器 - 或者色盲用户。

为了确保所有用户都能获取正确信息，Bootstrap 还提供了另一种提示方式。例如，你可以在表单控件的 `<label>` 标签上以文本的形式显示提示信息（就像下面代码中所展示的）；包含一个 Glyphicon 字体图标（还有赋予 `.sr-only` 类的文本信息 - 参考Glyphicon 字体图标实例）；或者提供一个额外的 辅助信息 块。另外，对于使用辅助设备的用户，无效的表单控件还可以赋予一个 `aria-invalid="true"` 属性。

实例：

### Input with success

A block of help text that breaks onto a new line and may extend beyond one line.

**Input with warning**

**Input with error**

☐ Checkbox with success

☐ Checkbox with warning

☐ Checkbox with error

---

```
<div class="form-group has-success">
  <label class="control-label" for="inputSuccess1">Input with success</label>
  <input type="text" class="form-control" id="inputSuccess1" aria-
describedby="helpBlock2">
  <span id="helpBlock2" class="help-block">A block of help text that breaks
onto a new line and may extend beyond one line.</span>
</div>
<div class="form-group has-warning">
  <label class="control-label" for="inputWarning1">Input with warning</label>
  <input type="text" class="form-control" id="inputWarning1">
</div>
<div class="form-group has-error">
  <label class="control-label" for="inputError1">Input with error</label>
  <input type="text" class="form-control" id="inputError1">
</div>
<div class="has-success">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxSuccess" value="option1">
      Checkbox with success
    </label>
  </div>
</div>
<div class="has-warning">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxWarning" value="option1">
      Checkbox with warning
    </label>
  </div>
</div>
<div class="has-error">
  <div class="checkbox">
    <label>
      <input type="checkbox" id="checkboxError" value="option1">
      Checkbox with error
    </label>
  </div>
</div>
```

## 添加额外的图标

你还可以针对校验状态为输入框添加额外的图标。只需设置相应的 `.has-feedback` 类并添加正确的图标即可。

反馈图标（**feedback icon**）只能使用在文本输入框 `<input class="form-control">` 元素上。

对于不带有 `label` 标签的输入框以及右侧带有附加组件的输入框组，需要手动为其图标定位。为了让所有用户都能访问你的网站，我们强烈建议为所有输入框添加 `label` 标签。如果你不希望将 `label` 标签展示出来，可以通过添加 `.sr-only` 类来实现。如果的确不能添加 `label` 标签，请调整图标的 `top` 值。对于输入框组，请根据你的实际情况调整 `right` 值。

## 向辅助技术设备传递图标的含义

为了确保辅助技术- 如屏幕阅读器 - 正确传达一个图标的含义，额外的隐藏的文本应包含在 `.sr-only` 类中，并明确关联使用了 `aria-describedby` 的表单控件。或者，以某些其他形式（例如，文本输入字段有一个特定的警告信息）传达含义，例如改变与表单控件实际相关联的 `<label>` 的文本。

虽然下面的例子已经提到各自表单控件本身的 `<label>` 文本的验证状态，上述技术（使用 `.sr-only` 文本 和 `aria-describedby`）已经包括了需要说明的目的。

实例：

### Input with success

### Input with warning

### Input with error

### Input group with success

```
<div class="form-group has-success has-feedback">
  <label class="control-label" for="inputSuccess2">Input with success</label>
  <input type="text" class="form-control" id="inputSuccess2" aria-
describedby="inputSuccess2Status">
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
  <span id="inputSuccess2Status" class="sr-only">(success)</span>
</div>

<div class="form-group has-warning has-feedback">
  <label class="control-label" for="inputWarning2">Input with warning</label>
  <input type="text" class="form-control" id="inputWarning2" aria-
describedby="inputWarning2Status">
  <span class="glyphicon glyphicon-warning-sign form-control-feedback" aria-
hidden="true"></span>
  <span id="inputWarning2Status" class="sr-only">(warning)</span>
</div>

<div class="form-group has-error has-feedback">
  <label class="control-label" for="inputError2">Input with error</label>
  <input type="text" class="form-control" id="inputError2" aria-
describedby="inputError2Status">
  <span class="glyphicon glyphicon-remove form-control-feedback" aria-
hidden="true"></span>
  <span id="inputError2Status" class="sr-only">(error)</span>
</div>

<div class="form-group has-success has-feedback">
  <label class="control-label" for="inputGroupSuccess1">Input group with
success</label>
  <div class="input-group">
    <span class="input-group-addon">@</span>
    <input type="text" class="form-control" id="inputGroupSuccess1" aria-
describedby="inputGroupSuccess1Status">
  </div>
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
  <span id="inputGroupSuccess1Status" class="sr-only">(success)</span>
</div>
```

## 为水平排列的表单和内联表单设置可选的图标

实例：

Input with success

✓

Input group with success

@

✓

```
<form class="form-horizontal">
  <div class="form-group has-success has-feedback">
    <label class="control-label col-sm-3" for="inputSuccess3">Input with
success</label>
    <div class="col-sm-9">
      <input type="text" class="form-control" id="inputSuccess3" aria-
describedby="inputSuccess3Status">
      <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
      <span id="inputSuccess3Status" class="sr-only">(success)</span>
    </div>
  </div>
  <div class="form-group has-success has-feedback">
    <label class="control-label col-sm-3" for="inputGroupSuccess2">Input group
with success</label>
    <div class="col-sm-9">
      <div class="input-group">
        <span class="input-group-addon">@</span>
        <input type="text" class="form-control" id="inputGroupSuccess2" aria-
describedby="inputGroupSuccess2Status">
      </div>
      <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
      <span id="inputGroupSuccess2Status" class="sr-only">(success)</span>
    </div>
  </div>
</form>
```

实例：

Input with success

✓

Input group with success

@

✓



```
<form class="form-inline">
  <div class="form-group has-success has-feedback">
    <label class="control-label" for="inputSuccess4">Input with success</label>
    <input type="text" class="form-control" id="inputSuccess4" aria-
describedby="inputSuccess4Status">
    <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
    <span id="inputSuccess4Status" class="sr-only">(success)</span>
  </div>
</form>

<form class="form-inline">
  <div class="form-group has-success has-feedback">
    <label class="control-label" for="inputGroupSuccess3">Input group with
success</label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input type="text" class="form-control" id="inputGroupSuccess3" aria-
describedby="inputGroupSuccess3Status">
    </div>
    <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
    <span id="inputGroupSuccess3Status" class="sr-only">(success)</span>
  </div>
</form>
```

## 可选的图标与设置 .sr-only 类的 label

如果你使用 .sr-only 类来隐藏表单控件的 <label> （而不是使用其它标签选项，如 aria-label 属性）， 一旦它被添加，Bootstrap 会自动调整图标的位置。

实例：

	✓
@	✓

```
<div class="form-group has-success has-feedback">
  <label class="control-label sr-only" for="inputSuccess5">Hidden label</label>
  <input type="text" class="form-control" id="inputSuccess5" aria-
describedby="inputSuccess5Status">
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
  <span id="inputSuccess5Status" class="sr-only">(success)</span>
</div>

<div class="form-group has-success has-feedback">
  <label class="control-label sr-only" for="inputGroupSuccess4">Input group
with success</label>
  <div class="input-group">
    <span class="input-group-addon">@</span>
    <input type="text" class="form-control" id="inputGroupSuccess4" aria-
describedby="inputGroupSuccess4Status">
  </div>
  <span class="glyphicon glyphicon-ok form-control-feedback" aria-
hidden="true"></span>
  <span id="inputGroupSuccess4Status" class="sr-only">(success)</span>
</div>
```

# 控件尺寸

通过 `.input-lg` 类似的类可以为控件设置高度，通过 `.col-lg-*` 类似的类可以为控件设置宽度。

## 高度尺寸

创建大一些或小一些的表单控件以匹配按钮尺寸。

实例：

.input-lg

Default input

.input-sm

.input-lg

Default select

.input-sm

```
<input class="form-control input-lg" type="text" placeholder=".input-lg">
<input class="form-control" type="text" placeholder="Default input">
<input class="form-control input-sm" type="text" placeholder=".input-sm">

<select class="form-control input-lg">...</select>
<select class="form-control">...</select>
<select class="form-control input-sm">...</select>
```

## 水平排列的表单组的尺寸

通过添加 `.form-group-lg` 或 `.form-group-sm` 类，为 `.form-horizontal` 包裹的 `label` 元素和表单控件快速设置尺寸。

实例：

### Large label

Large input

### Small label

Small input

```
<form class="form-horizontal">
  <div class="form-group form-group-lg">
    <label class="col-sm-2 control-label" for="formGroupInputLarge">Large
label</label>
    <div class="col-sm-10">
      <input class="form-control" type="text" id="formGroupInputLarge"
placeholder="Large input">
    </div>
  </div>
  <div class="form-group form-group-sm">
    <label class="col-sm-2 control-label" for="formGroupInputSmall">Small
label</label>
    <div class="col-sm-10">
      <input class="form-control" type="text" id="formGroupInputSmall"
placeholder="Small input">
    </div>
  </div>
</form>
```

## 调整列（column）尺寸

用栅格系统中的列（column）包裹输入框或其任何父元素，都可很容易的为其设置宽度。

实例：

.col-xs-2

.col-xs-3

.col-xs-4

```
<div class="row">
  <div class="col-xs-2">
    <input type="text" class="form-control" placeholder=".col-xs-2">
  </div>
  <div class="col-xs-3">
    <input type="text" class="form-control" placeholder=".col-xs-3">
  </div>
  <div class="col-xs-4">
    <input type="text" class="form-control" placeholder=".col-xs-4">
  </div>
</div>
```

# 按钮

## 可作为按钮使用的标签或元素

为 `<a>`、`<button>` 或 `<input>` 元素添加按钮类（button class）即可使用 Bootstrap 提供的样式。

实例：



```
<a class="btn btn-default" href="#" role="button">Link</a>
<button class="btn btn-default" type="submit">Button</button>
<input class="btn btn-default" type="button" value="Input">
<input class="btn btn-default" type="submit" value="Submit">
```

### 针对组件的注意事项

虽然按钮类可以应用到 `<a>` 和 `<button>` 元素上，但是，导航和导航条组件只支持 `<button>` 元素。

### 链接被作为按钮使用时的注意事项

如果 `<a>` 元素被作为按钮使用 -- 并用于在当前页面触发某些功能 -- 而不是用于链接其他页面或链接当前页面中的其他部分，那么，务必为其设置 `role="button"` 属性。

### 跨浏览器展现

我们总结的最佳实践是：强烈建议尽可能使用 `<button>` 元素来获得在各个浏览器上获得相匹配的绘制效果。

另外，我们还发现了 Firefox <30 版本的浏览器上出现的一个 bug，其表现是：阻止我们为基于 `<input>` 元素所创建的按钮设置 `line-height` 属性，这就导致在 Firefox 浏览器上不能完全和其他按钮保持一致的高度。

# 预定义样式

使用下面列出的类可以快速创建一个带有预定义样式的按钮。

实例：

(默认样式) Default

(首选项) Primary

(成功) Success

(一般信息) Info

(警告) Warning

(危险) Danger

(链接) Link

```
<!-- Standard button -->
<button type="button" class="btn btn-default"> (默认样式) Default</button>

<!-- Provides extra visual weight and identifies the primary action in a set of
buttons -->
<button type="button" class="btn btn-primary"> (首选项) Primary</button>

<!-- Indicates a successful or positive action -->
<button type="button" class="btn btn-success"> (成功) Success</button>

<!-- Contextual button for informational alert messages -->
<button type="button" class="btn btn-info"> (一般信息) Info</button>

<!-- Indicates caution should be taken with this action -->
<button type="button" class="btn btn-warning"> (警告) Warning</button>

<!-- Indicates a dangerous or potentially negative action -->
<button type="button" class="btn btn-danger"> (危险) Danger</button>

<!-- Deemphasize a button by making it look like a link while maintaining
button behavior -->
<button type="button" class="btn btn-link"> (链接) Link</button>
```

## Conveying meaning to assistive technologies

为按钮添加不同的颜色只是一种视觉上的信息表达方式，但是，对于使用辅助技术 -- 例如屏幕阅读器 -- 的用户来说，颜色是不可见的。建议，确保通过颜色表达的信息或者通过内容自身表达出来（按钮上的文字），或者通过其他方式 -- 例如通过 `.sr-only` 类隐藏的额外文本 -- 表达出来。

# 尺寸

需要让按钮具有不同尺寸吗？使用 `.btn-lg`、`.btn-sm` 或 `.btn-xs` 就可以获得不同尺寸的按钮。

实例：

(大按钮) Large button

(大按钮) Large button

(默认尺寸) Default button

(默认尺寸) Default button

(小按钮) Small button

(小按钮) Small button

(超小尺寸) Extra small button

(超小尺寸) Extra small button

```
<p>
  <button type="button" class="btn btn-primary btn-lg"> (大按钮) Large
button</button>
  <button type="button" class="btn btn-default btn-lg"> (大按钮) Large
button</button>
</p>
<p>
  <button type="button" class="btn btn-primary"> (默认尺寸) Default
button</button>
  <button type="button" class="btn btn-default"> (默认尺寸) Default
button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm"> (小按钮) Small
button</button>
  <button type="button" class="btn btn-default btn-sm"> (小按钮) Small
button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-xs"> (超小尺寸) Extra small
button</button>
  <button type="button" class="btn btn-default btn-xs"> (超小尺寸) Extra small
button</button>
</p>
```

通过给按钮添加 `.btn-block` 类可以将其拉伸至父元素100%的宽度，而且按钮也变为了块级（block）元素。

实例：

(块级元素) Block level button

(块级元素) Block level button

```
<button type="button" class="btn btn-primary btn-lg btn-block"> (块级元素) Block level button</button>
<button type="button" class="btn btn-default btn-lg btn-block"> (块级元素) Block level button</button>
```

## 激活状态

当按钮处于激活状态时，其表现为被按压下去（底色更深、边框夜色更深、向内投射阴影）。对于 `<button>` 元素，是通过 `:active` 状态实现的。对于 `<a>` 元素，是通过 `.active` 类实现的。然而，你还可以将 `.active` 应用到 `<button>` 上（包含 `aria-pressed="true"` 属性），并通过编程的方式使其处于激活状态。

## button 元素

由于 `:active` 是伪状态，因此无需额外添加，但是在需要让其表现出同样外观的时候可以添加 `.active` 类。

实例：

Primary button

Button

```
<button type="button" class="btn btn-primary btn-lg active">Primary button</button>
<button type="button" class="btn btn-default btn-lg active">Button</button>
```

## 链接（<a>）元素

可以为基于 `<a>` 元素创建的按钮添加 `.active` 类。

实例：

Primary link

Link

```
<a href="#" class="btn btn-primary btn-lg active" role="button">Primary link</a>
<a href="#" class="btn btn-default btn-lg active" role="button">Link</a>
```

# 禁用状态

通过为按钮的背景设置 `opacity` 属性就可以呈现出无法点击的效果。

## button 元素

为 `<button>` 元素添加 `disabled` 属性，使其表现出禁用状态。

实例：



```
<button type="button" class="btn btn-lg btn-primary" disabled="disabled">Primary button</button>
<button type="button" class="btn btn-default btn-lg" disabled="disabled">Button</button>
```

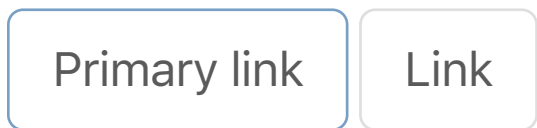
### 跨浏览器兼容性

如果为 `<button>` 元素添加 `disabled` 属性，Internet Explorer 9 及更低版本的浏览器将会把按钮中的文本绘制为灰色，并带有恶心的阴影，目前我们还没有解决办法。

## 链接（`<a>`）元素

为基于 `<a>` 元素创建的按钮添加 `.disabled` 类。

实例：



```
<a href="#" class="btn btn-primary btn-lg disabled" role="button">Primary link</a>
<a href="#" class="btn btn-default btn-lg disabled" role="button">Link</a>
```

我们把 `.disabled` 作为工具类使用，就像 `.active` 类一样，因此不需要增加前缀。

### 链接的原始功能不受影响

上面提到的类只是通过设置 `pointer-events: none` 来禁止 `<a>` 元素作为链接的原始功能，但是，这一 CSS 属性并没有被标准化，并且 Opera 18 及更低版本的浏览器并没有完全支持这一属性，同样，Internet Explorer 11 也不支持。In addition, even in browsers that do support



pointer-events: none , keyboard navigation remains unaffected, meaning that sighted keyboard users and users of assistive technologies will still be able to activate these links. 因此，为了安全起见，建议通过 JavaScript 代码来禁止链接的原始功能。

# 图片

## 响应式图片

在 Bootstrap 版本 3 中，通过为图片添加 `.img-responsive` 类可以让图片支持响应式布局。其实质是为图片设置了 `max-width: 100%;`、`height: auto;` 和 `display: block;` 属性，从而让图片在其父元素中更好的缩放。

如果需要让使用了 `.img-responsive` 类的图片水平居中，请使用 `.center-block` 类，不要用 `.text-center`。请参考助手类章节 了解更多关于 `.center-block` 的用法。

### SVG 图像和 IE 8-10

在 Internet Explorer 8-10 中，设置为 `.img-responsive` 的 SVG 图像显示出的尺寸不匀称。为了解决这个问题，在出问题的地方添加 `width: 100% \9;` 即可。Bootstrap 并没有自动为所有图像元素设置这一属性，因为这会导致其他图像格式出现错乱。

```

```

## 图片形状

通过为 `<img>` 元素添加以下相应的类，可以让图片呈现不同的形状。

### 跨浏览器兼容性

请时刻牢记：Internet Explorer 8 不支持 CSS3 中的圆角属性。

实例：



```



```

# 辅助类

## 情境文本颜色

通过颜色来展示意图，Bootstrap 提供了一组工具类。这些类可以应用于链接，并且在鼠标经过时颜色还可以加深，就像默认的连接一样。

实例：

Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.

Nullam id dolor id nibh ultricies vehicula ut id elit.

Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Maecenas sed diam eget risus varius blandit sit amet non magna.

Etiam porta sem malesuada magna mollis euismod.

Donec ullamcorper nulla non metus auctor fringilla.

```
<p class="text-muted">...</p>
<p class="text-primary">...</p>
<p class="text-success">...</p>
<p class="text-info">...</p>
<p class="text-warning">...</p>
<p class="text-danger">...</p>
```

Sometimes emphasis classes cannot be applied due to the specificity of another selector. In most cases, a sufficient workaround is to wrap your text in a `<span>` with the class.

## Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (the contextual colors are only used to reinforce meaning that is already present in the text/markup), or is included through alternative means, such as additional text hidden with the `.sr-only` class.

## 情境背景色

和情境文本颜色类一样，使用任意情境背景色类就可以设置元素的背景。链接组件在鼠标经过时颜色会加深，就像上面所讲的情境文本颜色类一样。

实例：

Nullam id dolor id nibh ultricies vehicula ut id elit.

Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Maecenas sed diam eget risus varius blandit sit amet non magna.

Etiam porta sem malesuada magna mollis euismod.

Donec ullamcorper nulla non metus auctor fringilla.

```
<p class="bg-primary">...</p>
<p class="bg-success">...</p>
<p class="bg-info">...</p>
<p class="bg-warning">...</p>
<p class="bg-danger">...</p>
```

## 处理差异

Sometimes contextual background classes cannot be applied due to the specificity of another selector. In some cases, a sufficient workaround is to wrap your element's content in a `<div>` with the class.

## Conveying meaning to assistive technologies

As with contextual colors, ensure that any meaning conveyed through color is also conveyed in a format that is not purely presentational.

## 关闭按钮

通过使用一个象征关闭的图标，可以让模态框和警告框消失。

实例：



```
<button type="button" class="close" aria-label="Close"><span aria-hidden="true">&times;</span></button>
```

## 三角符号

通过使用三角符号可以指示某个元素具有下拉菜单的功能。注意，向上弹出式菜单中的三角符号是反方向的。

实例：



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

## 快速浮动

通过添加一个类，可以将任意元素向左或向右浮动。 `!important` 被用来明确 CSS 样式的优先级。这些类还可以作为 `mixin`（参见 `less` 文档）使用。

```
<div class="pull-left">...</div>
<div class="pull-right">...</div>
```

```
// Classes
.pull-left {
  float: left !important;
}
.pull-right {
  float: right !important;
}

// Usage as mixins
.element {
  .pull-left();
}
.another-element {
  .pull-right();
}
```

## 不能用于导航条组件中

排列导航条中的组件时可以使用这些工具类： `.navbar-left` 或 `.navbar-right` 。参见导航条文档以获取更多信息。

## 让内容块居中

为任意元素设置 `display: block` 属性并通过 `margin` 属性让其中的内容居中。下面列出的类还可以作为 `mixin` 使用。

```
<div class="center-block">...</div>
```

```
// Class
.center-block {
  display: block;
  margin-left: auto;
  margin-right: auto;
}

// Usage as a mixin
.element {
  .center-block();
}
```

## 清除浮动

通过为父元素添加 `.clearfix` 类可以很容易地清除浮动（ `float` ）。这里所使用的是 Nicolas Gallagher 创造的 `micro clearfix` 方式。此类还可以作为 `mixin` 使用。

```
<!-- Usage as a class -->
<div class="clearfix">...</div>
```

```
// Mixin itself
.clearfix() {
  &:before,
  &:after {
    content: " ";
    display: table;
  }
  &:after {
    clear: both;
  }
}

// Usage as a mixin
.element {
  .clearfix();
}
```

## 显示或隐藏内容

`.show` 和 `.hidden` 类可以强制任意元素显示或隐藏(对于屏幕阅读器也能起效)。这些类通过 `!important` 来避免 CSS 样式优先级问题，就像 quick floats 一样的做法。注意，这些类只对块级元素起作用，另外，还可以作为 mixin 使用。

`.hide` 类仍然可用，但是它不能对屏幕阅读器起作用，并且从 v3.0.1 版本开始就不建议使用了。请使用 `.hidden` 或 `.sr-only`。

另外，`.invisible` 类可以被用来仅仅影响元素的可见性，也就是说，元素的 `display` 属性不被改变，并且这个元素仍然能够影响文档流的排布。

```
<div class="show">...</div>
<div class="hidden">...</div>
```

```
// Classes
.show {
  display: block !important;
}
.hidden {
  display: none !important;
}
.invisible {
  visibility: hidden;
}

// Usage as mixins
.element {
  .show();
}
.another-element {
  .hidden();
}
```

## 屏幕阅读器和键盘导航

`.sr-only` 类可以对屏幕阅读器以外的设备隐藏内容。`.sr-only` 和 `.sr-only-focusable` 联合使用的话可以在元素有焦点的时候再次显示出来（例如，使用键盘导航的用户）。对于遵循可访问性的最佳实践很有必要。这个类也可以作为 mixin 使用。

```
<a class="sr-only sr-only-focusable" href="#content">Skip to main
content</a>
```

```
// Usage as a mixin
.skip-navigation {
  .sr-only();
  .sr-only-focusable();
}
```

## 图片替换

使用 `.text-hide` 类或对应的 mixin 可以用来将元素的文本内容替换为一张背景图。

```
<h1 class="text-hide">Custom heading</h1>
```

```
// Usage as a mixin
.heading {
  .text-hide();
}
```

# 响应式工具

为了加快对移动设备友好的页面开发工作，利用媒体查询功能并使用这些工具类可以方便的针对不同设备展示或隐藏页面内容。另外还包含了针对打印机显示或隐藏内容的工具类。

有针对性的使用这类工具类，从而避免为同一个网站创建完全不同的版本。相反，通过使用这些工具类可以在不同设备上提供不同的展现形式。

## 可用的类

通过单独或联合使用以下列出的类，可以针对不同屏幕尺寸隐藏或显示页面内容。

	超小屏幕 手机 (<768px)	小屏幕 平板 (≥768px)	中等屏幕 桌面 (≥992px)	大屏幕 桌面 (≥1200px)
.visible-xs-*	可见	隐藏	隐藏	隐藏
.visible-sm-*	隐藏	可见	隐藏	隐藏
.visible-md-*	隐藏	隐藏	可见	隐藏
.visible-lg-*	隐藏	隐藏	隐藏	可见
.hidden-xs	隐藏	可见	可见	可见
.hidden-sm	可见	隐藏	可见	可见
.hidden-md	可见	可见	隐藏	可见
.hidden-lg	可见	可见	可见	隐藏

从 v3.2.0 版本起，形如 `.visible-*-*` 的类针对每种屏幕大小都有了三种变体，每个针对 CSS 中不同的 `display` 属性，列表如下：

类组	CSS display
.visible-*-block	<code>display: block;</code>
.visible-*-inline	<code>display: inline;</code>
.visible-*-inline-block	<code>display: inline-block;</code>

因此，以超小屏幕（xs）为例，可用的 `.visible-*-*` 类是：`.visible-xs-block`、`.visible-xs-inline` 和 `.visible-xs-inline-block`。



`.visible-xs`、`.visible-sm`、`.visible-md` 和 `.visible-lg` 类也同时存在。但是从 **v3.2.0** 版本开始不再建议使用。除了 `<table>` 相关的元素的特殊情况外，它们与 `.visible-*-block` 大体相同。

# 打印类

和常规的响应式类一样，使用下面的类可以针对打印机隐藏或显示某些内容。

class	浏览器	打印机
<code>.visible-print-block</code> <code>.visible-print-inline</code> <code>.visible-print-inline-block</code>	隐藏	可见
<code>.hidden-print</code>	可见	隐藏

`.visible-print` 类也是存在的，但是从 v3.2.0 版本开始不建议使用。它与 `.visible-print-block` 类大致相同，除了 `<table>` 相关元素的特殊情况外。

# 测试用例

调整你的浏览器大小，或者用其他设备打开页面，都可以测试这些响应式工具类。

## 在...上可见

带有绿色标记的元素表示其在当前浏览器视口（viewport）中是可见的。

✓ 在超小屏幕上可见	小屏幕
中等屏幕	大屏幕
✓ 在超小屏幕和小屏幕上可见	中等屏幕和大屏幕
✓ 在超小屏幕和中等屏幕上可见	小屏幕和大屏幕
✓ 在超小屏幕和大屏幕上可见	小屏幕和中等屏幕

## 在...上隐藏

带有绿色标记的元素表示其在当前浏览器视口（viewport）中是隐藏的。

✓ 在超小屏幕上隐藏	小屏幕
------------	-----

中等屏幕	大屏幕
✓ 在超小屏幕和小屏幕上隐藏	中等屏幕和大屏幕
✓ 在超小屏幕和中等屏幕上隐藏	小屏幕和大屏幕
✓ 在超小屏幕和大屏幕上隐藏	小屏幕和中等屏幕

# 使用 Less

Bootstrap 的 CSS 文件是通过 Less 源码编译而来的。Less 是一门预处理语言，支持变量、mixin、函数等额外功能。对于希望使用 Less 源码而非编译而来的 CSS 文件的用户，Bootstrap 框架中包含的大量变量、mixin 将非常有价值。

针对栅格系统的变量和 mixin 包含在栅格系统章节。

## 编译 Bootstrap

可以通过两种方式使用 Bootstrap：使用编译后的 CSS 文件或者使用 Less 源码文件。若要编译 Less 文件，请参考“起步”章节的内容以了解如何设置开发环境并运行必须的编译指令。

## 变量

整个 Bootstrap 项目中使用了大量的变量，这些变量被用来代表颜色、空白（内部、边距）、字体等。详细内容请参考定制工具。

## 颜色

Bootstrap 使用了两种颜色模式：灰度颜色和语义颜色。灰度颜色用于快速获取常用的黑色色调；语义颜色包含了各种赋予语义的颜色值。

实例：

```
@gray-darker: lighten(#000, 13.5%); // #222
@gray-dark:   lighten(#000, 20%);   // #333
@gray:        lighten(#000, 33.5%);  // #555
@gray-light:  lighten(#000, 46.7%);  // #777
@gray-lighter: lighten(#000, 93.5%); // #eee
```

实例：

```
@brand-primary: darken(#428bca, 6.5%); // #337ab7
@brand-success: #5cb85c;
@brand-info:     #5bc0de;
@brand-warning:  #f0ad4e;
@brand-danger:   #d9534f;
```

你在项目中可以使用这些预定义的颜色变量，或者重新为其赋予别名，使其更有语义。

```
// Use as-is
.masthead {
  background-color: @brand-primary;
}

// Reassigned variables in Less
@alert-message-background: @brand-info;
.alert {
  background-color: @alert-message-background;
}
```

## Scaffolding

某几个变量是改变网站外观的关键要素。

```
// Scaffolding
@body-bg: #fff;
@text-color: @black-50;
```

## 链接

仅仅通过改变一个变量，可以很容易地为链接赋予正确的颜色。

```
// Variables
@link-color:          @brand-primary;
@link-hover-color:    darken(@link-color, 15%);

// Usage
a {
  color: @link-color;
  text-decoration: none;

  &:hover {
    color: @link-hover-color;
    text-decoration: underline;
  }
}
```

注意：@link-hover-color 使用了 Less 提供的一个内置函数，用于自动为鼠标悬停设置合适的颜色。你还可以使用 darken、lighten、saturate 和 desaturate 等 Less 内置的函数。

## 排版

通过几个变量就能轻松的设置字体、字号、行距等。Bootstrap 利用这些变量提供了简单地定制排版的功能。

```
@font-family-sans-serif: "Helvetica Neue", Helvetica, Arial, sans-serif;
@font-family-serif:      Georgia, "Times New Roman", Times, serif;
@font-family-monospace:  Menlo, Monaco, Consolas, "Courier New",
monospace;
@font-family-base:       @font-family-sans-serif;

@font-size-base:         14px;
@font-size-large:        ceil((@font-size-base * 1.25)); // ~18px
@font-size-small:        ceil((@font-size-base * 0.85)); // ~12px

@font-size-h1:           floor((@font-size-base * 2.6)); // ~36px
@font-size-h2:           floor((@font-size-base * 2.15)); // ~30px
@font-size-h3:           ceil((@font-size-base * 1.7)); // ~24px
@font-size-h4:           ceil((@font-size-base * 1.25)); // ~18px
@font-size-h5:           @font-size-base;
@font-size-h6:           ceil((@font-size-base * 0.85)); // ~12px

@line-height-base:       1.428571429; // 20/14
@line-height-computed:   floor((@font-size-base * @line-height-base)); //
~20px

@headings-font-family:   inherit;
@headings-font-weight:   500;
@headings-line-height:   1.1;
@headings-color:         inherit;
```

## 图标

以下两个变量用于设置图标文件的位置和文件名。

```
@icon-font-path:          "../fonts/";  
@icon-font-name:          "glyphicons-halflings-regular";
```

## 组件

组件贯穿整个 Bootstrap 框架，他们通过一些变量来设置默认值。下面列出的是常用的几个。

```
@padding-base-vertical:    6px;  
@padding-base-horizontal:   12px;  
  
@padding-large-vertical:    10px;  
@padding-large-horizontal:   16px;  
  
@padding-small-vertical:    5px;  
@padding-small-horizontal:   10px;  
  
@padding-xs-vertical:       1px;  
@padding-xs-horizontal:     5px;  
  
@line-height-large:         1.33;  
@line-height-small:         1.5;  
  
@border-radius-base:        4px;  
@border-radius-large:       6px;  
@border-radius-small:       3px;  
  
@component-active-color:    #fff;  
@component-active-bg:       @brand-primary;  
  
@caret-width-base:          4px;  
@caret-width-large:         5px;
```

## 特定浏览器厂商的 mixin

特定浏览器厂商的 mixin 用于为不同厂商的浏览器使用相应的 CSS 属性前缀来支持各厂商的浏览器。

## Box-sizing

通过这一个 mixin 来为所有组件设置盒模型。请参考这篇 来自 Mozilla 的文章。

此 mixin 从 v3.2.0 版本开始就被列为 **不建议使用** 了，取而代之的是使用 Autoprefixer。为了保持向后兼容，在 v4 版本之前，Bootstrap 将在内部继续使用这些 mixin。

```
.box-sizing(@box-model) {
  -webkit-box-sizing: @box-model; // Safari <= 5
  -moz-box-sizing: @box-model; // Firefox <= 19
  box-sizing: @box-model;
}
```

## 圆角

现在，所有现代浏览器都支持不带厂商前缀的 `border-radius` 属性了。有鉴于此，我们没有提供 `.border-radius()` mixin，但是，Bootstrap does 提供了用于快速设置同一侧圆角的 mixin。

```
.border-top-radius(@radius) {
  border-top-right-radius: @radius;
  border-top-left-radius: @radius;
}
.border-right-radius(@radius) {
  border-bottom-right-radius: @radius;
  border-top-right-radius: @radius;
}
.border-bottom-radius(@radius) {
  border-bottom-right-radius: @radius;
  border-bottom-left-radius: @radius;
}
.border-left-radius(@radius) {
  border-bottom-left-radius: @radius;
  border-top-left-radius: @radius;
}
```

## Box (Drop) 隐形

如果你的目标用户使用的是最新版本和更高级的浏览器和设备，只需单独使用 `box-shadow` 属性即可。如果你需要兼容较老的 Android (低于 v4) 和 iOS 设备 (低于 iOS 5)，可以使用下面这个 **不建议使用** 的 mixin，便于帮你添加 `-webkit` 前缀。

由于 Bootstrap 并未官方提供对过时（不支持标准属性）平台的支持，此 mixin 从 v3.1.0 版本起就 **不建议使用** 了。为了保持向后兼容，Bootstrap 将继续在内部使用此 mixin，直到 Bootstrap v4。

在设置 box 阴影时务必使用 `rgba()` 颜色，这样可以使他们尽可能地与背景无缝融入。

```
.box-shadow(@shadow: 0 1px 3px rgba(0,0,0,.25)) {
  -webkit-box-shadow: @shadow; // iOS <4.3 & Android <4.1
  box-shadow: @shadow;
}
```

## 过渡效果

有多个 `mixin` 供你灵活使用。可以一次性设置所有的过渡效果的属性，或者根据需要只是指定延时和持续时间。

此 `mixin` 从 v3.2.0 版本开始就被列为 **不建议使用** 了，取而代之的是使用 `Autoprefixer`。为了保持向后兼容，在 v4 版本之前，Bootstrap 将在内部继续使用这些 `mixin`。

```
.transition(@transition) {
  -webkit-transition: @transition;
  transition: @transition;
}
.transition-property(@transition-property) {
  -webkit-transition-property: @transition-property;
  transition-property: @transition-property;
}
.transition-delay(@transition-delay) {
  -webkit-transition-delay: @transition-delay;
  transition-delay: @transition-delay;
}
.transition-duration(@transition-duration) {
  -webkit-transition-duration: @transition-duration;
  transition-duration: @transition-duration;
}
.transition-timing-function(@timing-function) {
  -webkit-transition-timing-function: @timing-function;
  transition-timing-function: @timing-function;
}
.transition-transform(@transition) {
  -webkit-transition: -webkit-transform @transition;
  -moz-transition: -moz-transform @transition;
  -o-transition: -o-transform @transition;
  transition: transform @transition;
}
```

## 变形

旋转、缩放、平移（移动）或倾斜任何对象。

此 `mixin` 从 v3.2.0 版本开始就被列为 **不建议使用** 了，取而代之的是使用 `Autoprefixer`。为了保持向后兼容，在 v4 版本之前，Bootstrap 将在内部继续使用这些 `mixin`。

```
.rotate(@degrees) {
  -webkit-transform: rotate(@degrees);
  -ms-transform: rotate(@degrees); // IE9 only
  transform: rotate(@degrees);
}

.scale(@ratio; @ratio-y...) {
  -webkit-transform: scale(@ratio, @ratio-y);
  -ms-transform: scale(@ratio, @ratio-y); // IE9 only
  transform: scale(@ratio, @ratio-y);
}

.translate(@x; @y) {
  -webkit-transform: translate(@x, @y);
  -ms-transform: translate(@x, @y); // IE9 only
  transform: translate(@x, @y);
}

.skew(@x; @y) {
  -webkit-transform: skew(@x, @y);
  -ms-transform: skewX(@x) skewY(@y); // See
https://github.com/twbs/bootstrap/issues/4885; IE9+
  transform: skew(@x, @y);
}

.translate3d(@x; @y; @z) {
  -webkit-transform: translate3d(@x, @y, @z);
  transform: translate3d(@x, @y, @z);
}

.rotateX(@degrees) {
  -webkit-transform: rotateX(@degrees);
  -ms-transform: rotateX(@degrees); // IE9 only
  transform: rotateX(@degrees);
}

.rotateY(@degrees) {
  -webkit-transform: rotateY(@degrees);
  -ms-transform: rotateY(@degrees); // IE9 only
  transform: rotateY(@degrees);
}

.perspective(@perspective) {
  -webkit-perspective: @perspective;
  -moz-perspective: @perspective;
  perspective: @perspective;
}

.perspective-origin(@perspective) {
  -webkit-perspective-origin: @perspective;
  -moz-perspective-origin: @perspective;
  perspective-origin: @perspective;
}

.transform-origin(@origin) {
  -webkit-transform-origin: @origin;
  -moz-transform-origin: @origin;
  -ms-transform-origin: @origin; // IE9 only
  transform-origin: @origin;
}
```



# 动画

仅适用一个 mixin 就可以在一个声明中使用所有 CSS3 所提供的动画属性，其他 mixin 用于设置单个属性。

此 mixin 从 v3.2.0 版本开始就 **不建议使用** 了，取而代之的是使用 Autoprefixer。为了保持向后兼容，在 v4 版本之前，Bootstrap 将在内部继续使用这些 mixin。

```
.animation(@animation) {
  -webkit-animation: @animation;
  animation: @animation;
}
.animation-name(@name) {
  -webkit-animation-name: @name;
  animation-name: @name;
}
.animation-duration(@duration) {
  -webkit-animation-duration: @duration;
  animation-duration: @duration;
}
.animation-timing-function(@timing-function) {
  -webkit-animation-timing-function: @timing-function;
  animation-timing-function: @timing-function;
}
.animation-delay(@delay) {
  -webkit-animation-delay: @delay;
  animation-delay: @delay;
}
.animation-iteration-count(@iteration-count) {
  -webkit-animation-iteration-count: @iteration-count;
  animation-iteration-count: @iteration-count;
}
.animation-direction(@direction) {
  -webkit-animation-direction: @direction;
  animation-direction: @direction;
}
```

# 透明度

为所有浏览器设置透明度，并为IE8提供 filter 备用滤镜。

```
.opacity(@opacity) {
  opacity: @opacity;
  // IE8 filter
  @opacity-ie: (@opacity * 100);
  filter: ~"alpha(opacity=@{opacity-ie})";
}
```

# 占位符文本

为表单控件中每个文本域提供占位符（Placeholder）文本的颜色。

```
.placeholder(@color: @input-color-placeholder) {  
  &::-moz-placeholder { color: @color; } // Firefox  
  &:-ms-input-placeholder { color: @color; } // Internet Explorer 10+  
  &::-webkit-input-placeholder { color: @color; } // Safari and Chrome  
}
```

## 列

通过CSS在一个单独的元素中生成列。

```
.content-columns(@width; @count; @gap) {  
  -webkit-column-width: @width;  
  -moz-column-width: @width;  
  column-width: @width;  
  -webkit-column-count: @count;  
  -moz-column-count: @count;  
  column-count: @count;  
  -webkit-column-gap: @gap;  
  -moz-column-gap: @gap;  
  column-gap: @gap;  
}
```

## 渐变

便于把任何两种颜色变成背景渐变色。想要使他更高级些，可以设置一个direction（方向），使用三种颜色，也可以使用径向（radial）渐变。使用一个mixin（混入），你就可以得到所有需要的前缀语法。

```
#gradient > .vertical(#333; #000);  
#gradient > .horizontal(#333; #000);  
#gradient > .radial(#333; #000);
```

你也可以为标准的里两颜色线性渐变指定角度：

```
#gradient > .directional(#333; #000; 45deg);
```

如果你需要一个条纹风格的渐变，这也很容易。只要指定一个颜色，我们将该颜色半透明的条纹覆盖其上。

```
#gradient > .striped(#333; 45deg);
```

再来试试三种颜色。利用此 mixin，并为其设置第一种颜色、第二种颜色、第二种颜色的色标（例如25%），还有第三种颜色：

```
#gradient > .vertical-three-colors(#777; #333; 25%; #000);  
#gradient > .horizontal-three-colors(#777; #333; 25%; #000);
```

当心！如果你想删除某个渐变，确保将你所添加的针对 IE 的 `filter` 一并删除。你可以通过使用 `.reset-filter()` mixin 和 `background-image: none;` 达到目的。

## 实用工具 mixin

实用工具 mixin 用于与不相关的 CSS 结合以达到特定目的或任务。

### Clearfix -- 清除浮动

建议为需要清除浮动的元素使用 `.clearfix()` mixin，尽量不要直接添加 `class="clearfix"` 类。基于 Nicolas Gallagher 的 micro clearfix 代码。

```
// Mixin  
.clearfix() {  
  &:before,  
  &:after {  
    content: " ";  
    display: table;  
  }  
  &:after {  
    clear: both;  
  }  
}  
  
// Usage  
.container {  
  .clearfix();  
}
```

### 水平居中

让元素在其父元素中水平居中。需要设置 `width` 或 `max-width` 属性。

```
// Mixin  
.center-block() {  
  display: block;  
  margin-left: auto;  
  margin-right: auto;  
}  
  
// Usage  
.container {  
  width: 940px;  
  .center-block();  
}
```

# 尺寸助手 mixin

用于方便的指定对象的尺寸。

```
// Mixins
.size(@width; @height) {
  width: @width;
  height: @height;
}

.square(@size) {
  .size(@size; @size);
}

// Usage
.image { .size(400px; 300px); }
.avatar { .square(48px); }
```

## 可调整大小的文本域

方便设置任何文本域或其他元素的尺寸可调整。默认依循浏览器默认行为（`both`），即垂直、水平都可以调整。

```
.resizable(@direction: both) {
  // Options: horizontal, vertical, both
  resize: @direction;
  // Safari fix
  overflow: auto;
}
```

## 截断文本

此 mixin 用来以省略号代替被截断的文本。元素必须是 **block** 或 **inline-block** 级。

```
// Mixin
.text-overflow() {
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}

// Usage
.branch-name {
  display: inline-block;
  max-width: 200px;
  .text-overflow();
}
```

## 视网膜屏幕（Retina）下的图片

通过指定两个图片路径和 @1x 图片尺寸，Bootstrap 还提供了对 @2x 媒体查询的支持。如果你的页面上有很多图片，建议在一个单独的媒体查询中手工编写针对视网膜屏幕的 **CSS** 代码。

```
.img-retina(@file-1x; @file-2x; @width-1x; @height-1x) {
  background-image: url("@{file-1x}");

  @media
  only screen and (-webkit-min-device-pixel-ratio: 2),
  only screen and (   min--moz-device-pixel-ratio: 2),
  only screen and (     -o-min-device-pixel-ratio: 2/1),
  only screen and (       min-device-pixel-ratio: 2),
  only screen and (         min-resolution: 192dpi),
  only screen and (         min-resolution: 2dppx) {
    background-image: url("@{file-2x}");
    background-size: @width-1x @height-1x;
  }
}

// Usage
.jumbotron {
  .img-retina("/img/bg-1x.png", "/img/bg-2x.png", 100px, 100px);
}
```

# 使用 Sass

虽然 Bootstrap 是基于 Less 构建的，我们还提供了一套官方支持的 Sass 移植版代码。我们将这个版本放在单独的 GitHub 仓库中进行维护，并通过脚本处理源码更新。

## 包含的内容

由于 Sass 移植版存放于单独的仓库，并针对不同的使用群体，这个项目中的内容与 Bootstrap 主项目有很大不同。这也是为了保证 Sass 移植版与更多基于 Sass 的系统相兼容。

路径	描述
lib/	Ruby gem code (Sass configuration, Rails and Compass integrations)
tasks/	Converter scripts (turning upstream Less to Sass)
test/	Compilation tests
templates/	Compass package manifest
vendor/assets/	Sass, JavaScript, and font files

Rakefile	Internal tasks, such as rake and convert
----------	--

请访问 Sass 移植版在 GitHub 上的仓库 来了解这些文件。

# 安装

关于如何安装并使用 Bootstrap 的 Sass 移植版，请参考GitHub 仓库中的 readme 文件。 此仓库中包含了最新的源码以及如何与 Rails、Compass 以及标准 Sass 项目一同使用的详细信息。

Bootstrap for Sass

[GitHub 仓库](#) [实例](#) [优站精选](#) [关于](#)

Designed and built with all the love in the world by @mdo and @fat. Maintained by the core team with the help of our contributors.

本项目源码受 MIT开源协议保护，文档受 CC BY 3.0 开源协议保护。