Bootstrap3中文版

极客学院出版

前言

温馨提示：本站短域名 boot3.com

Bootstrap 是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目。

本教程根据官方最新版本（V3.3.4）改编，对原文做了精简，突出了学习的重点，帮助读者快速掌握 Bootstrap 3 的最新更新的内容。

0 基础前端服务群

>

前端公开课基础群　　Web 前端导学课

欢迎加 QQ 群： 185186017

说明：专为 **0** 基础用户提供全技术服务交流，定期提供一线大牛**免费**公开课福利。

>

条件：首批开放给：零基础\转行从业者

适用人群

只要您具备 HTML 和 CSS 的基础知识，您就可以阅读本教程，进而开发出自己的网站。在您学习完本教程后，您即可达到使用 Bootstrap 开发 Web 项目的中等水平。

学习前提

在您开始阅读本教程之前，您必须具备 HTML、CSS 和 JavaScript 的基础知识。

视频教程

极客学院为 Bootstrap 3 录制了一套完整的学习视频教程，希望能达到手把手教学的目的：
www.jikexueyuan.com/path/bootstrap/

最新版

Bootstrap v4.0.0-alpha 官方文档中文版全国首发，详细请访问：http://boot4.com

目录

1

起步

下载

Bootstrap （当前版本 v3.3.5）提供以下几种方式帮你快速上手，每一种方式针对具有不同技能等级的开发者和不同的使用场景。继续阅读下面的内容，看看哪种方式适合你的需求吧。

- 用于生产环境的 Bootstrap

编译并压缩后的 CSS、JavaScript 和字体文件。不包含文档和源码文件。

- Bootstrap 源码

Less、JavaScript 和 字体文件的源码，并且带有文档。需要 Less 编译器和一些设置工作。

- [Sass](这是 Bootstrap 从 Less 到 Sass 的源码移植项目，用于快速地在 Rails、Compass 或 只针对 Sass 的项目中引入。)

```
<!-- 新 Bootstrap 核心 CSS 文件 -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

<!-- 可选的Bootstrap主题文件（一般不用引入） -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
```

通过 Bower 进行安装

还可以通过 Bower 安装并管理 Bootstrap 的 Less、CSS、JavaScript 和字体文件。

```
$ bower install bootstrap
```

通过 npm 进行安装

你还可以利用 npm工具来安装 Bootstrap：

```
$ npm install bootstrap
```

`require('bootstrap')` 代码的作用是加载 Bootstrap 的所有 jQuery 插件。其中，`bootstrap` 模块自身并不导出任何内容。你可以通过加载安装包顶级目录下的 `/js/*.js` 文件的方式手动加载单个的 Bootstrap 插件。

Bootstrap 的 package.json 文件包含了一些额外的元数据：

- less - Bootstrap 源码的入口 Less 文件的路径
- style - Bootstrap 的未压缩 CSS 文件的路径

通过 Composer 进行安装

通过 Composer （中文官网：Composer 中文网）也可以安装 Bootstrap 安装包，其中包括 Less、CSS、JavaScript 和 fonts 文件：

```
$ composer require twbs/bootstrap
```

编译 Less/Sass 源码需要注意的事项

Bootstrap 利用 Autoprefixer 自动为 某些 CSS 属性添加针对特定厂商的前缀。如果你是从 Less/Sass 源码编译 Bootstrap 的，并且没有使用 Bootstrap 自带的 Gruntfile 文件，那你就必须将 Autoprefixer 集成到你的编译工具和编译过程中。如果你使用的是我们预先编译好的 Bootstrap 文件或者使用的是我们提供的 Gruntfile 文件，那就无需操心了，我们已经将这些工作替你搞定了。

包含的内容

Bootstrap 提供了两种形式的压缩包，在下载下来的压缩包内可以看到以下目录和文件，这些文件按照类别放到了不同的目录内，并且提供了压缩与未压缩两种版本。

`Bootstrap 插件全部依赖 jQuery` 请注意，**Bootstrap 的所有 JavaScript 插件都依赖 jQuery**，因此 jQuery 必须在 Bootstrap 之前引入，就像在基本模版中所展示的一样。在 `bower.json` 文件中 列出了 Bootstrap 所支持的 jQuery 版本。

预编译版

下载压缩包之后，将其解压缩到任意目录即可看到以下（压缩版的）目录结构：

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

上面展示的就是 Bootstrap 的基本文件结构：预编译文件可以直接使用到任何 web 项目中。我们提供了编译好的 CSS 和 JS（`bootstrap.*`）文件，还有经过压缩的 CSS 和 JS（`bootstrap.min.*`）文件。同时还提供了 CSS 源码映射表（`bootstrap.*.map`），可以在某些浏览器的开发工具中使用。同时还包含了来自 Glyphicons 的图标字体，在附带的 Bootstrap 主题中使用到了这些图标。

Bootstrap 源码

Bootstrap 源码包含了预先编译的 CSS、JavaScript 和图标字体文件，并且还有 LESS、JavaScript 和文档的源码。具体来说，主要文件组织结构如下：

```
bootstrap/
├── less/
├── js/
├── fonts/
├── dist/
│   ├── css/
│   ├── js/
│   └── fonts/
└── docs/
    └── examples/
```

`less/`、`js/` 和 `fonts/` 目录分别包含了 CSS、JS 和字体图标的源码。`dist/` 目录包含了上面所说的预编译 Bootstrap 包内的所有文件。`docs/` 包含了所有文档的源码文件，`examples/` 目录是 Bootstrap 官方提供的实例工程。除了这些，其他文件还包含 Bootstrap 安装包的定义文件、许可证文件和编译脚本等。

编译 CSS 和 JavaScript 文件

Bootstrap 使用 Grunt 作为编译系统，并且对外提供了一些方便的方法用于编译整个框架。下面讲解的就是如何编译源码、运行测试用例等内容。

安装 Grunt

安装 Grunt 前，你需要首先下载并安装 node.js （npm 已经包含在内）。npm 是 node packaged modules 的简称，它的作用是基于 node.js 管理扩展包之间的依赖关系。

然后在命令行中输入以下命令： 在全局环境中安装 `grunt-cli` ： `npm install -g grunt-cli` 。 进入 `/bootstrap/` 根目录，然后执行 `npm install` 命令。npm 将读取 `package.json` 文件并自动安装此文件中列出的所有被依赖的扩展包。 上述步骤完成后，你就可以运行 Bootstrap 所提供的各个 Grunt 命令了。

可用的 Grunt 命令

`grunt dist` （仅编译 CSS 和 JavaScript 文件）

重新生成 `/dist/` 目录，并将编译压缩后的 CSS 和 JavaScript 文件放入这个目录中。作为一名 Bootstrap 用户，大部分情况下你只需要执行这一个命令。

`grunt watch` （监测文件的改变，并运行指定的 Grunt 任务）

监测 Less 源码文件的改变，并自动重新将其编译为 CSS 文件。

`grunt test` （运行测试用例）

在 PhantomJS 环境中运行 JSHint 和 QUnit 自动化测试用例。

`grunt docs` （编译并测试文档中的资源文件）

编译并测试 CSS、JavaScript 和其他资源文件。在本地环境下通过 jekyll serve 运行 Bootstrap 文档时需要用到这些资源文件。

`grunt` （重新构建所有内容并运行测试用例）

编译并压缩 CSS 和 JavaScript 文件、构建文档站点、对文档做 HTML5 校验、重新生成定制工具所需的资源文件等，都需要 Jekyll 工具。这些只有在你对 Bootstrap 深度研究时才有用。

除错

如果你在安装依赖包或者运行 Grunt 命令时遇到了问题，请首先删除 npm 自动生成的 `/node_modules/` 目录，然后，再次运行 `npm install` 命令。

基本模板

使用以下给出的这份超级简单的 HTML 模版，或者修改这些实例。我们强烈建议你对这些实例按照自己的需求进行修改，而不要简单的复制、粘贴。

拷贝并粘贴下面给出的 HTML 代码，这就是一个最简单的 Bootstrap 页面了。

```
<!DOCTYPE html>
<html lang="zh-cn">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>你好，世界！</h1>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

工具

Bootlint

Bootlint 是 Bootstrap 官方所支持的 HTML 检测工具。在使用了 Bootstrap 的页面上（没有对 Bootstrap 做修改和扩展的情况下），它能自动检查某些常见的 HTML 错误。纯粹的 Bootstrap 组件需要固定的 DOM 结构。Bootlint 就能检测你的页面上的这些“纯粹”的 Bootstrap 组件是否符合 Bootstrap 的 HTML 结构规则。建议将 Bootlint 加入到你的开发工具中，这样就能帮你在项目开发中避免一些简单的错误影响你的开发进度。

社区

Bootstrap 有成熟、强大的社区。如果你希望随时获取 Bootstrap 的最新消息，请关注以下社区资源。

- 阅读并订阅 Bootstrap 官方博客。
- 通过 IRC （`irc.freenode.net` 服务器）与其他 Bootstrap 粉丝交流，我们在 ##bootstrap 频道。
- 如果使用 Bootstrap 过程中遇到问题，请在 StackOverflow 上交流。请搜索 `twitter-bootstrap-3` 关键词。
- 参观其他同学基于 Bootstrap 构建的网站，请进这里：Bootstrap 优站精选。

你还可以在 Twitter 上关注英文官方账号 @twbootstrap ，这里有最新的八卦和有趣的视频。（ :(原来和技术不沾边啊！）

禁止响应式布局

Bootstrap 会自动帮你针对不同的屏幕尺寸调整你的页面，使其在各个尺寸的屏幕上表现良好。下面我们列出了如何禁用这一特性，就像这个非响应式布局实例页面一样。

禁止响应式布局有如下几步：

- 移除 此 CSS 文档中提到的设置浏览器视口（viewport）的标签：`<meta>`。
- 通过为 `.container` 类设置一个 `width` 值从而覆盖框架的默认 width 设置，例如 `width: 970px !important;` 。请确保这些设置全部放在默认的 Bootstrap CSS 文件的后面。注意，如果你把它放到媒体查询中，也可以略去 `!important` 。
- 如果使用了导航条，需要移除所有导航条的折叠和展开行为。
- 对于栅格布局，额外增加 `.col-xs-*` 类或替换掉 `.col-md-*` 和 `.col-lg-*`。 不要担心，针对超小屏幕设备的栅格系统能够在所有分辨率的环境下展开。

针对 IE8 仍然需要额外引入 Respond.js 文件（由于仍然利用了浏览器对媒体查询（media query）的支持，因此还需要做处理）。这样就禁用了 Bootstrap 对移动设备的响应式支持。

禁止响应式特性的 Bootstrap 模版

我们已经按照上面的步骤制作了一份案例。仔细阅读其源码并对照上面的步骤实践一下吧。

从 v2.x 版本升级到 v3.x 版本

你在找从老版本升级到 Bootstrap v3.x 版本的方法吗？请查看我们整理的升级指南吧。

浏览器和设备的支持情况

Bootstrap 的目标是在最新的桌面和移动浏览器上有最佳的表现，也就是说，在较老旧的浏览器上可能会导致某些组件表现出的样式有些不同，但是功能是完整的。

被支持的浏览器

特别注意，我们坚决支持这些浏览器的**最新版本**。在 Windows 平台，我们支持 Internet Explorer 8-11。请看下面列出的详细信息。

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	支持	支持	N/A	不支持	N/A
iOS	支持	N/A		不支持	支持
Mac OS X	支持	支持		支持	支持
Windows	支持	支持	支持	支持	不支持

Bootstrap 在 Chromium 和 Linux 版 Chrome、Linux 版 Firefox 和 Internet Explorer 7 上的表现也是很不错的，虽然我们不对其进行官方支持。

浏览器 bug 列表中列出了影响 Bootstrap 正常功能的浏览器 bug。

Internet Explorer 8 和 9

Internet Explorer 8 和 9 是被支持的，然而，你要知道，很多 CSS3 属性和 HTML5 元素 -- 例如，圆角矩形和投影 -- 是肯定不被支持的。另外， **Internet Explorer 8 需要 Respond.js 配合才能实现对媒体查询（media query）的支持。**

Feature	Internet Explorer 8	Internet Explorer 9
`border-radius`	不支持	支持
`box-shadow`	不支持	支持
`transform`	不支持	支持, with `-ms` prefix
`transition`	不支持	
`placeholder`	不支持	

Internet Explorer 8 与 Respond.js

在开发环境和生产（线上）环境需要支持 Internet Explorer 8 时，请务必注意下面给出的警告。

Respond.js 与 跨域（cross-domain） CSS 的问题

如果 Respond.js 和 CSS 文件被放在不同的域名或子域名下面（例如，使用CDN）时需要一些额外的设置。请参考 R[Respond.jshref="https://github.com/scottjehl/Respond/blob/master/README.md") 文档 获取详细信息。

Respond.js 与 file:// 协议

由于浏览器的安全机制，Respond.js 不能在通过 `file://` 协议（打开本地HTML文件所用的协议）访问的页面上发挥正常的功能。如果需要测试 IE8 下面的响应式特性，务必通过 http 协议访问页面（例如搭建 apache、nginx 等）。请参考 R[Respond.jshref="https://github.com/scottjehl/Respond/blob/master/README.md") 文档获取更多信息。

Respond.js 与 `@import` 指令

Respond.js 不支持通过 `@import` 指令所引入的 CSS 文件。例如，Drupal 一般被配置为通过 `@import` 指令引入CSS文件，Respond.js 对其将无法起到作用。请参考 [Respond.jshref="https://github.com/scottjehl/Respond/blob/master/README.md") 文档获取更多信息。

Internet Explorer 8 与 box-sizing 属性

当 box-sizing: border-box; 与 min-width、max-width、min-height 或 max-height 一同使用时，IE8 不能完全支持 box-sizing 属相。由于此原因，从 Bootstrap v3.0.1 版本开始，我们不再为 .container 赋予 max-width 属性。

Internet Explorer 8 与 @font-face

当 `@font-face` 与 `:before` 在 IE8 下共同使用时会出现问题。由于 Bootstrap 对 Glyphicons 的样式设置使用了这一组合方式，如果某个页面被浏览器缓存了，并且此页面不是通过点击“刷新”按钮或通过 iframe 加载的，那么就会导致字体文件尚未加载的情况下就开始绘制此页面。当鼠标滑过页面（body）时，页面上的某些图

标就会显现，鼠标滑过其他没有显现的图标时，这些图标就能显示出来了。请参考 issue #13863 了解详细信息。

IE 兼容模式

Bootstrap 不支持 IE 古老的兼容模式。为了让 IE 浏览器运行最新的渲染模式下，建议将此 `<meta>` 标签加入到你的页面中：

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

按 F12 键打开 IE 的调试工具，就可以看到 IE 当前的渲染模式是什么。

此 meta 标签被包含在了所有 Bootstrap 文档和实例页面中，为的就是在每个被支持的 IE 版本中拥有最好的绘制效果。

请参考 这个发表在 StackOverflow 上的问题。

国产浏览器高速模式

国内浏览器厂商一般都支持兼容模式（即 IE 内核）和高速模式（即 webkit 内核），不幸的是，所有国产浏览器都是默认使用兼容模式，这就造成由于低版本 IE （IE8 及以下）内核让基于 Bootstrap 构建的网站展现效果很糟糕的情况。幸运的是，国内浏览器厂商逐渐意识到了这一点，某些厂商已经开始有所作为了！

将下面的 `<meta>` 标签加入到页面中，可以让部分国产浏览器默认采用高速模式渲染页面：

```
    <meta name="renderer" content="webkit">
```

目前只有360浏览器支持此`<meta>`标签。希望更多国内浏览器尽快采取行动、尽快进入高速时代！

Windows 8 中的 Internet Explorer 10 和 Windows Phone 8

Internet Explorer 10 并没有对 **屏幕的宽度 和 视口（viewport）的宽度** 进行区分，这就导致 Bootstrap 中的媒体查询并不能很好的发挥作用。为了解决这个问题，你可以引入下面列出的这段 CSS 代码暂时修复此问题：

```
    @-ms-viewport       { width: device-width; }
```

然而，这样做并不能对 Windows Phone 8 Update 3 (a.k.a. GDR3) 版本之前的设备起作用，由于这样做将导致 Windows Phone 8 设备按照桌面浏览器的方式呈现页面，而不是较窄的“手机”呈现方式，为了解决这个问题，还需要加入以下 CSS 和 JavaScript 代码来化解此问题。

```
    @-webkit-viewport   { width: device-width; }
    @-moz-viewport      { width: device-width; }
    @-ms-viewport       { width: device-width; }
    @-o-viewport        { width: device-width; }
    @viewport           { width: device-width; }
```

```
if (navigator.userAgent.match(/IEMobile\/10\.0/)) {
  var msViewportStyle = document.createElement('style')
  msViewportStyle.appendChild(
    document.createTextNode(
      '@-ms-viewport{width:auto!important}'
    )
  )
  document.querySelector('head').appendChild(msViewportStyle)
    }
```

请查看 Windows Phone 8 and Device-Width 以了解更多信息。

作为提醒，我们将上面的代码加入到了所有 Bootstrap 文档和实例页面中。

Safari 对百分比数字凑整的问题

OS X 上搭载的 v7.1 以前 Safari 和 iOS v8.0 上搭载的 Safari 浏览器的绘制引擎对于处理 `.col-*-1` 类所对应的很长的百分比小数存在 bug。也就是说，如果你在一行（row）之中定义了12个单独的列（`.col-*-1`），你就会看到这一行比其他行要短一些。除了升级 Safari/iOS 外，有以下几种方式来应对此问题：

为最后一列添加 `.pull-right` 类，将其暴力向右对齐

手动调整百分比数字，让其针对 Safari 表现更好（这比第一种方式更困难）

模态框、导航条和虚拟键盘

Overflow and scrolling

`<body>` 元素在 iOS 和 Android 上对 `overflow: hidden` 的支持很有限。结果就是，在这两种设备上的浏览器中，当你滚动屏幕超过模态框的顶部或底部时，`<body>` 中的内容将开始随着滚动。

虚拟键盘

还有，如果你正在使用 fixed 定位的导航条或在模态框上面使用输入框，还会遇到 iOS 在页面绘制上的 bug，当触发虚拟键盘之后，其不会更新 fixed 定位的元素的位置。这里有几种解决方案，包括将 fixed 定位转变

为 `position: absolute` 定位，或者启动一个定时器手工修正组件的位置。这些没有加入 Bootstrap 中，因此，需要由你自己选择最好的解决方案并加入到你的应用中。

导航条上的下拉菜单

在 iOS 设备上，由于导航组件（nav）的复杂的 z-indexing 属性，.dropdown-backdrop 元素并未被使用。因此，为了关闭导航条上的下拉菜单，必须直接点击下拉菜单上的元素（或者任何其他能够触发 iOS 上 click 事件的元素）。

浏览器的缩放功能

页面缩放功能不可避免的会将某些组件搞得乱七八糟，不光是 Bootstrap ，整个互联网上的所有页面都是这样。针对具体问题，我们或许可以修复它（如果有必要的话，请先搜索一下你的问题，看看是否已有解决方案，然后在向我们提交 issue）。然而，我们更倾向于忽略这些问题，由于这些问题除了一些 hack 手段，一般没有直接的解决方案。

打印

即便是在某些很现代的浏览器中，打印页面功能也还是存在很多陷阱。

举个例子，从 Chrome v32 开始，打印一个支持媒体查询的页面时，不管如何设置留白，Chrome 总是使用一个远远小于实际页面尺寸的视口宽度的值作为页面宽度。这就导致被打印的页面总是被呈现为在超小屏幕（extra-small）上的效果（也就是激活了 Bootstrap 针对超小屏幕的栅格排布方式）。 参考 #12078 了解更多信息。 推荐解决方案：

- 让你的页面在超小（extra-small）屏幕上看起来不那么太差劲。
- 修改 `@screen-*` Less 变量的值，让你的页面总是大于 extra-small
- 添加额外的媒体查询代码，针对打印机修改栅格阈值。

另外，从 Safari v8.0 开始，固定宽度的 `.container` 会导致 Safari 使用非常小的字号来打印页面。参见 #14868了解跟多信息。下面这段 CSS 代码提供了一个临时解决方案：

```
@media print {
  .container {
    width: auto;
  }
}
```

Android 系统默认浏览器

Android 4.1 （甚至某些较新版本）系统的默认浏览器被设置为默认打开页面的应用程序（不同于 Chrome）。不幸的是， 一般情况下，这些浏览器有很多 bug 以及和 CSS 标准不一致的地方。

选项菜单

如果 `<select>` 元素应用了 border-radius 和 / 或 border 样式，Android 系统默认的浏览器将不会显示侧边栏控件。 使用下面的代码片段来删除有问题的 CSS 并且在Android 系统默认的浏览器上，`<select> as an` 呈现为无样式元素。可以通过检测用户代理（user agent）的特征串来避免干扰 Chrome、Safari 和 Mozilla 浏览器。

```
<script>
$(function () {
  var nua = navigator.userAgent
  var isAndroid = (nua.indexOf('Mozilla/5.0') > -1 && nua.indexOf('Android ') > -1 && nua.indexOf('AppleWebKit') > -1 &
  if (isAndroid) {
    $('select.form-control').removeClass('form-control').css('width', '100%')
  }
})
</script>
```

见 JS Bin 上的 demo。

W3C 页面源码校验

为了在老旧的浏览器上尽量提供最好的展现，Bootstrap 针对浏览器使用了一些 CSS hack 手段，为的是针对特定浏览器版本弥补浏览器自身的 bug。这些 CSS hack 手段在 CSS 校验器那里会被认为是无效代码。还有一些地方，我们使用了某些未被完全标准化的 CSS 特性，纯粹是为了实现渐进式增强的思路。

上面提到的这些校验器报告的警告信息并不会对实际使用造成影响，因为非 hack 部分的 CSS 是完全合格的，hack 部分不会对非 hack 部分的功能产生影响，这就是我们故意无视这些校验器警告的原因。

同样，我们的 HTML 文档中也有一些针对 Firefox bug 的 hack 代码，在 HTML 校验时也会被警告。

对第三方组件的支持

虽然我们并不官方支持任何第三方插件，我们还是提供一些建议，帮你避免可能在你的项目中会出现的问题。

box-sizing 属性

某些第三方软件，包括 Google 地图和 Google 定制搜索引擎都会由于 `* { box-sizing: border-box; }` 的设置而产生冲突，这一设置使 `padding` 不影响页面元素最终宽度的计算。更多信息请参考 盒模型与尺寸计算 - CSS Tricks。

根据不同情况，你可能需要根据情况覆盖（第1种选择）或为所有区域设置（第2种选择）。

```
/* Box-sizing resets
 *
 * 为了避免 Bootstrap 设置的全局盒模型所带来的影响，可以重置单个页面元素或覆盖整个区域的盒模型。
 * 你有两种选择：覆盖单个页面元素或重置整个区域。它们都可以通过纯 CSS 或 LESS 代码的形式实现。
 */

/* Option 1A: 通过 CSS 代码覆盖单个页面元素的盒模型 */
.element {
  -webkit-box-sizing: content-box;
     -moz-box-sizing: content-box;
          box-sizing: content-box;
}

/* Option 1B: 通过使用 Bootstrap 提供的 LESS mixin 覆盖单个页面元素的盒模型 */
.element {
  .box-sizing(content-box);
}

/* Option 2A: 通过 CSS 代码重置整个区域 */
.reset-box-sizing,
.reset-box-sizing *,
.reset-box-sizing *:before,
.reset-box-sizing *:after {
  -webkit-box-sizing: content-box;
     -moz-box-sizing: content-box;
          box-sizing: content-box;
}

/* Option 2B: 通过使用自定义的 LESS mixin 重置整个区域 */
```

```
.reset-box-sizing {
  &,
  *,
  *:before,
  *:after {
    .box-sizing(content-box);
  }
}
.element {
  .reset-box-sizing();
}
```

可访问性

Bootstrap 遵循统一的 web 标准，另外，通过做一些少量的修改，还可以让使用 辅助设备（AT - ASSISTIVE TECHNOLOGY）上网的人群访问你的站点。

跳过导航区

如果你的导航部分包含很多链接，并且在 DOM 结构上也是排在主内容之前，那么，建议在导航前面添加一个 Skip to main content（直接进入主内容区） 的链接（这篇文章 A11Y Project article on skip navigation links给了简要的解释）。 通过使用 `.sr-only` 类可以让 “Skip to main content（直接进入主内容区）” 链接在视觉上是不可见的，而 `.sr-only-focusable` 类可以让这个链接在获得焦点的时候变为可见（对于使用键盘导航的用户）。

```
<body>
  <a href="#content" class="sr-only sr-only-focusable">Skip to main content</a>
  ...
  <div class="container" id="content">
    <!-- The main page content -->
  </div>
</body>
```

标题嵌套

当标题嵌套时（`<h1>` - `<h6>`），你的文档的主标题应该是 `<h1>` 标签。随后的标题逻辑上就应该使用 `<h2>` - `<h6>` ，这样，屏幕阅读器就可以构造出页面的内容列表了。

更多信息请参考： HTML CodeSniffer and Penn State's AccessAbility.

色彩对比

当前，Bootstrap 使用的默认颜色组合的对比度（例如各种 styled button 类；一些 基本代码块高亮代码所用的颜色；`.bg-primary` 上下文背景色 辅助类；和白色背景下默认的链接颜色）都比较低（低于 推荐比例 4.5:1）。这会导致视力低下和色盲用户在使用时产生困难。这些默认颜色可能需要进行修改，以增强对比度和清晰度。

2

全局 CSS 样式

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` a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处的类。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（aligment）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 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。

.col-xs-9.col-xs-4 Since 9 + 4 = 13 > 12, this 4-column-wide div gets wrapped onto a new line as on e contiguous unit..col-xs-6 Subsequent columns continue along the new line.

```
<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 &gt; 12, this 4-column-wide div gets wrapped onto a new line
  <div class="col-xs-6">.col-xs-6<br>Subsequent columns continue along the new line.</div>
</div>
```

Responsive column resets

With the four tiers of grids available you're bound to run into issues where, at certain breakpoint s, your columns don't clear quite right as one is taller than the other. To fix that, use a combina tion of a `.clearfix` and our responsive utility classes.

.col-xs-6 .col-sm-3 Resize your viewport or check it out on your phone for an example. .col-xs-6 .c ol-sm-3.col-xs-6 .col-sm-3.col-xs-6 .col-sm-3

```
<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>
```

In addition to column clearing at responsive breakpoints, you may need to **reset offsets, pushes, or pulls.** See this in action in the grid example.

```
<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-offse
</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 .co
</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>
```

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;
  // 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。

```
<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。

```
<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）。

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

中心内容

通过添加 `.lead` 类可以让段落突出显示。

```
<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

标签定义带有记号的文本用 `<mark>` 标签。

```
    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。

```
    <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。

```
    <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。

```
    <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。

```
<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。

```
<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。

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

斜体

用斜体强调一段文本。

```
<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，不带有任何着重的意味；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。

```
<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。

```
<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

```
<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。

```
<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。

```
<address>
  <strong>Twitter, Inc.</strong><br>
  795 Folsom Ave, Suite 600<br>
  San Francisco, CA 94107<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 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。

```
<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。

```
<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

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

列表

无序列表

排列顺序无关紧要的一列元素。

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

有序列表

顺序至关重要的一组元素。

```
<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。

```
<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。

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

描述

带有描述的短语列表。

```
<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。

```
<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。

```
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。=

```
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。

```
<pre>&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。

```
<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。

```
<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。

```
<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。

```
<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。

```
<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。

```
<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。

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

状态类

```
<!-- 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的内容截断。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。除非使用我们下面提供的针对 Firefox 的 hack 代码，否则无解：

```
@-moz-document url-prefix() {
  fieldset { display: 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。

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" placeholder="Enter 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>
```

不要将表单组合输入框组混合使用

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。

```
<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>
```

```
<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="Enter 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>
```

```
<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。

```
<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。

```
<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。

```
<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。

设置了 `disabled` 属性的单选或多选框都能被赋予合适的样式。对于和多选或单选框联合使用的 `<label>` 标签，如果也希望将悬停于上方的鼠标设置为“禁止点击”的样式，请将 .disabled 类赋予 `.radio` 、 `.radio-inline` 、 `.checkbox` 、 `.checkbox-inline` 或 `<fieldset>` 。

默认外观（堆叠在一起）

```
    <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。

```
    <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）正是你说期望的。 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` ）。

```
<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。

```
<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。

```
<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。

```
<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>
```

```
<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。

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。

```
<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>`。

```
<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。

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

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。

```
<div class="form-group has-success">
  <label class="control-label" for="inputSuccess1">Input with success</label>
```

```
  <input type="text" class="form-control" id="inputSuccess1">
</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。

图标、 label 和输入控件组

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。

```
<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

```
<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>
```

```
<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。

```
<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。

```
    <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。

```
<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。

```
<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>
```

辅助文本

针对表单控件的“块（block）”级辅助文本。

与表单控件相关联的帮助文本

与表单控件相关联的帮助文本 `aria-describedby` 属性的表单控件关联，这将确保使用辅助技术- 如屏幕阅读器 - 的用户获取控件焦点或进入控制时显示这个帮助文本。

```
<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
```

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。

```
<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 `` ` `` 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。

```
<!-- 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>
```

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。

```
<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。

```
<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。

```
<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。

```
<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。

```
<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。

```
<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。此外，有的浏览器设置了 pointer-events: none，键盘导航仍然不受影响。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。

```
<img src="..." class="img-responsive" alt="Responsive image">
```

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

```
<img src="..." alt="..." class="img-rounded">
<img src="..." alt="..." class="img-circle">
<img src="..." alt="..." class="img-thumbnail">
```

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可以还可以加深，就像默认的链接一样。

```
<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>
```

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。

```
<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>
```

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。

```
<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。

```
<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，也就是所，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;
      visibility: hidden !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

类组	CSS `display`
`.visible-*-block`	`display: block;`
`.visible-*-inline`	`display: inline;`
`.visible-*-inline-block`	`display: inline-block;`

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
.visible-print-block .visible-print-inline .visible-print-inline-block	隐藏	可见
.hidden-print	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。

超小屏幕	小屏幕
中等屏幕	✓ 在大屏幕上可见
超小屏幕和小屏幕	✓ 在中等屏幕和大屏幕上可见
超小屏幕和中等屏幕	✓ 在小屏幕和大屏幕上可见
✓ 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。

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。

```
@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
```

```
@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 版本期就 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。

T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。

3

组件

Glyphicons 字体图标

如何使用

出于性能的考虑，所有图标都需要一个基类和对应每个图标的类。把下面的代码放在任何地方都可以正常使用。注意，为了设置正确的内补（padding），务必在图标和文本之间添加一个空格。

不要和其他组件混合使用

图标类不能和其它组件直接联合使用。它们不能在同一个元素上与其他类共同存在。应该创建一个嵌套的 `<span>` 标签，并将图标类应用到这个 `<span>` 标签上。

只对内容为空的元素起作用

图标类只能应用在不包含任何文本内容或子元素的元素上。

改变图标字体文件的位置

Bootstrap 假定所有的图标字体文件全部位于 `../fonts/` 目录内，相对于预编译版 CSS 文件的目录。如果你修改了图标字体文件的位置，那么，你需要通过下面列出的任何一种方式来更新 CSS 文件：

- 在 Less 源码文件中修改 `@icon-font-path` 和 `/` 或 `@icon-font-name` 变量。
- 利用 Less 编译器提供的 相对 URL 地址选项。
- 修改预编译 CSS 文件中的 url() 地址。

根据你自身的情况选择一种方式即可。

图标的可访问性

现代的辅助技术能够识别并朗读由 CSS 生成的内容和特定的 Unicode 字符。为了避免 屏幕识读设备抓取非故意的和可能产生混淆的输出内容（尤其是当图标纯粹作为装饰用途时），我们为这些图标设置了 `aria-hidden="true"` 属性。

如果你使用图标是为了表达某些含义（不仅仅是为了装饰用），请确保你所要表达的意思能够通过被辅助设备识别，例如，包含额外的内容并通过 `.sr-only` 类让其在视觉上表现出隐藏的效果。

如果你所创建的组件不包含任何文本内容（例如， `<button>` 内只包含了一个图标），你应当提供其他的内容来表示这个控件的意图，这样就能让使用辅助设备的用户知道其作用了。这种情况下，你可以为控件添加 aria-label 属相。

```
<span class="glyphicon glyphicon-search" aria-hidden="true"></span>
```

实例

可以把它们应用到按钮、工具条中的按钮组、导航或输入框等地方。

```
<button type="button" class="btn btn-default" aria-label="Left Align">
  <span class="glyphicon glyphicon-align-left" aria-hidden="true"></span>
</button>

<button type="button" class="btn btn-default btn-lg">
  <span class="glyphicon glyphicon-star" aria-hidden="true"></span> Star
</button>
```

alert 组件中所包含的图标是用来表示这是一条错误消息的，通过添加额外的 `.sr-only` 文本就可以让辅助设备知道这条提示所要表达的意思了。

下拉菜单

用于显示链接列表的可切换、有上下文的菜单。下拉菜单的 JavaScript 插件让它具有了交互性。

实例

将下拉菜单触发器和下拉菜单都包裹在 `.dropdown` 里，或者另一个声明了 `position: relative;` 的元素。然后加入组成菜单的 HTML 代码。

```
<div class="dropdown">
  <button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu1" data-toggle="dropdown" aria-haspopup
    Dropdown
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
    <li><a href="#">Action</a></li>
    <li><a href="#">Another action</a></li>
    <li><a href="#">Something else here</a></li>
    <li><a href="#">Separated link</a></li>
  </ul>
</div>
```

通过为下拉菜单的父元素设置 `.dropup` 类，可以让菜单向上弹出（默认是向下弹出的）。

```
<div class="dropup">
  <button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu2" data-toggle="dropdown" aria-haspopup
    Dropup
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenu2">
    <li><a href="#">Action</a></li>
    <li><a href="#">Another action</a></li>
    <li><a href="#">Something else here</a></li>
    <li><a href="#">Separated link</a></li>
  </ul>
</div>
```

对齐

B 默认情况下，下拉菜单自动沿着父元素的上沿和左侧被定位为 100% 宽度。 为 `.dropdown-menu` 添加 `.dropdown-menu-right` 类可以让菜单右对齐。

可能需要额外的定位May require additional positioning

在正常的文档流中，通过 CSS 为下拉菜单进行定位。这就意味着下拉菜单可能会由于设置了 overflow 属性的父元素而被部分遮挡或超出视口（viewport）的显示范围。如果出现这种问题，请自行解决。

不建议使用 `.pull-right`

从 v3.1.0 版本开始，我们不再建议对下拉菜单使用 `.pull-right` 类。如需将菜单右对齐，请使用 `.dropdown-menu-right` 类。导航条中如需添加右对齐的导航（nav）组件，请使用 `.pull-right` 的 mixin 版本，可以自动对齐菜单。如需左对齐，请使用 `.dropdown-menu-left` 类。

```
<ul class="dropdown-menu dropdown-menu-right" aria-labelledby="dLabel">
  ...
</ul>
```

标题

在任何下拉菜单中均可通过添加标题来标明一组动作。

```
<ul class="dropdown-menu" aria-labelledby="dropdownMenu3">
  ...
  <li class="dropdown-header">Dropdown header</li>
  ...
</ul>
```

分割线

为下拉菜单添加一条分割线，用于将多个链接分组。

```
<ul class="dropdown-menu" aria-labelledby="dropdownMenuDivider">
  ...
  <li role="separator" class="divider"></li>
```

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

禁用的菜单项

为下拉菜单中的 `<li>` 元素添加 `.disabled` 类，从而禁用相应的菜单项。

```
<ul class="dropdown-menu" aria-labelledby="dropdownMenu4">
  <li><a href="#">Regular link</a></li>
  <li class="disabled"><a href="#">Disabled link</a></li>
  <li><a href="#">Another link</a></li>
</ul>
```

按钮组

通过按钮组容器把一组按钮放在同一行里。通过与按钮插件联合使用，可以设置为单选框或多选框的样式和行为。

按钮组中的工具提示和弹出框需要特别的设置

当为 `.btn-group` 中的元素应用工具提示或弹出框时，必须指定 `container: 'body'` 选项，这样可以避免不必要的副作用（例如工具提示或弹出框触发时，会让页面元素变宽和/或失去圆角）。 确保设置正确的 role 属性并提供一个 label 标签 为了向使用辅助技术 – 如屏幕阅读器 – 的用户正确传达一正确的按钮分组，需要提供一个合适的 role 属性。对于按钮组合，应该是 `role="group"`，对于 toolbar（工具栏）应该是 `role="toolbar"` 。

一个例外是按钮组合只包含一个单一的控制元素或一个下拉菜单（比如实际情况，`<button>` 元素组成的两端对齐排列的按钮组 ）或下拉菜单。

此外，按钮组和工具栏应给定一个明确的 label 标签，尽管设置了正确的 role 属性，但是大多数辅助技术将不会正确的识读他们。在这里提供的实例中，我们使用 aria-label，但是， aria-labelledby 也可以使用。

按钮工具栏

把一组 `<div class="btn-group">` 组合进一个 `<div class="btn-toolbar">` 中就可以做成更复杂的组件。

```
<div class="btn-toolbar" role="toolbar" aria-label="...">
  <div class="btn-group" role="group" aria-label="...">...</div>
  <div class="btn-group" role="group" aria-label="...">...</div>
  <div class="btn-group" role="group" aria-label="...">...</div>
</div>
```

尺寸

只要给 `.btn-group` 加上 `.btn-group-*` 类，就省去为按钮组中的每个按钮都赋予尺寸类了，如果包含了多个按钮组时也适用。

```
<div class="btn-group btn-group-lg" role="group" aria-label="...">...</div>
<div class="btn-group" role="group" aria-label="...">...</div>
```

```
<div class="btn-group btn-group-sm" role="group" aria-label="...">...</div>
<div class="btn-group btn-group-xs" role="group" aria-label="...">...</div>
```

嵌套

想要把下拉菜单混合到一系列按钮中，只须把 `.btn-group` 放入另一个 `.btn-group` 中。

```
<div class="btn-group" role="group" aria-label="...">
  <button type="button" class="btn btn-default">1</button>
  <button type="button" class="btn btn-default">2</button>

  <div class="btn-group" role="group">
    <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expa
      Dropdown
      <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
      <li><a href="#">Dropdown link</a></li>
      <li><a href="#">Dropdown link</a></li>
    </ul>
  </div>
</div>
```

垂直排列

让一组按钮垂直堆叠排列显示而不是水平排列。分列式按钮下拉菜单不支持这种方式。

```
<div class="btn-group-vertical" role="group" aria-label="...">
  ...
</div>
```

两端对齐排列的按钮组

让一组按钮拉长为相同的尺寸，填满父元素的宽度。对于按钮组中的按钮式下拉菜单也同样适用。

关于边框的处理

由于对两端对齐的按钮组使用了特定的 HTML 和 CSS （即 `display: table-cell`），两个按钮之间的边框叠加在了一起。在普通的按钮组中，`margin-left: -1px` 用于将边框重叠，而没有删除任何一个按钮的边框。然而，mar

gin 属性不支持 `display: table-cell`。因此，根据你对 Bootstrap 的定制，你可以删除或重新为按钮的边框设置颜色。

IE8 和边框

Internet Explorer 8 不支持在两端对齐的按钮组中绘制边框，无论是 `<a>` 或 `<button>` 元素。为了照顾 IE 8，把每个按钮放入另一个 `.btn-group` 中即可。

参见 #12476 获取详细信息。

关于 `<a>` 元素

只须将一系列 `.btn` 元素包裹到 `.btn-group.btn-group-justified` 中即可。

```
<div class="btn-group btn-group-justified" role="group" aria-label="...">
  ...
</div>
```

关于 `<button>` 元素

为了将 `<button>` 元素用于两端对齐的按钮组中，必须将每个按钮包裹进一个按钮组中。大部分的浏览器不能将我们的 CSS 应用到对齐的 `<button>` 元素上，但是，由于我们支持按钮式下拉菜单，我们可以解决这个问题。

```
<div class="btn-group btn-group-justified" role="group" aria-label="...">
  <div class="btn-group" role="group">
    <button type="button" class="btn btn-default">Left</button>
  </div>
  <div class="btn-group" role="group">
    <button type="button" class="btn btn-default">Middle</button>
  </div>
  <div class="btn-group" role="group">
    <button type="button" class="btn btn-default">Right</button>
  </div>
</div>
```

按钮式下拉菜单

把任意一个按钮放入 `.btn-group` 中，然后加入适当的菜单标签，就可以让按钮作为菜单的触发器了。

插件依赖：按钮式下拉菜单依赖下拉菜单插件 ，因此需要将此插件包含在你所使用的 Bootstrap 版本中。

单按钮下拉菜单

只要改变一些基本的标记，就能把按钮变成下拉菜单的开关。

```
<!-- Single button -->
<div class="btn-group">
  <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expand
    Action <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    <li><a href="#">Action</a></li>
    <li><a href="#">Another action</a></li>
    <li><a href="#">Something else here</a></li>
    <li role="separator" class="divider"></li>
    <li><a href="#">Separated link</a></li>
  </ul>
</div>
```

分裂式按钮下拉菜单

相似地，分裂式按钮下拉菜单也需要同样的改变一些标记，但只是多一个分开的按钮。

```
<!-- Split button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger">Action</button>
  <button type="button" class="btn btn-danger dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expande
    <span class="caret"></span>
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <li><a href="#">Action</a></li>
    <li><a href="#">Another action</a></li>
    <li><a href="#">Something else here</a></li>
    <li role="separator" class="divider"></li>
```

```
    <li><a href="#">Separated link</a></li>
  </ul>
</div>
```

尺寸

按钮式下拉菜单适用所有尺寸的按钮。

```
<!-- Large button group -->
<div class="btn-group">
  <button class="btn btn-default btn-lg dropdown-toggle" type="button" data-toggle="dropdown" aria-haspopup="true" aria
    Large button <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>

<!-- Small button group -->
<div class="btn-group">
  <button class="btn btn-default btn-sm dropdown-toggle" type="button" data-toggle="dropdown" aria-haspopup="true" aria
    Small button <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>

<!-- Extra small button group -->
<div class="btn-group">
  <button class="btn btn-default btn-xs dropdown-toggle" type="button" data-toggle="dropdown" aria-haspopup="true" aria
    Extra small button <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

向上弹出式菜单

给父元素添加 `.dropup` 类就能使触发的下拉菜单朝上方打开。

```
<div class="btn-group dropup">
  <button type="button" class="btn btn-default">Dropup</button>
  <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expand
    <span class="caret"></span>
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

输入框组

通过在文本输入框 `<input>` 前面、后面或是两边加上文字或按钮，可以实现对表单控件的扩展。为 `.input-group` 赋予 `.input-group-addon` 类，可以给 `.form-control` 的前面或后面添加额外的元素。

只支持文本输入框 `<input>`

这里请避免使用 `<select>` 元素，因为 WebKit 浏览器不能完全绘制它的样式。

避免使用 `<textarea>` 元素，由于它们的 rows 属性在某些情况下不被支持。

输入框组中的工具提示和弹出框需要特别的设置

为 `.input-group` 中所包含的元素应用工具提示（tooltip）或popover（弹出框）时，必须设置 `container: 'body'` 参数，为的是避免意外的副作用（例如，工具提示或弹出框被激活后，可能会让当前元素变得更宽或/和变得失去其圆角）。

不要和其他组件混用

不要将表单组或栅格列（column）类直接和输入框组混合使用。而是将输入框组嵌套到表单组或栅格相关元素的内部。

基本实例

在输入框的任意一侧添加额外元素或按钮。你还可以在输入框的两侧同时添加额外元素。

我们不支持在输入框的单独一侧添加多个额外元素。

我们不支持在单个输入框组中添加多个表单控件。

```
<div class="input-group">
  <span class="input-group-addon" id="basic-addon1">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-describedby="basic-addon1">
</div>

<div class="input-group">
  <input type="text" class="form-control" placeholder="Recipient's username" aria-describedby="basic-addon2">
```

```
  <span class="input-group-addon" id="basic-addon2">@example.com</span>
</div>

<div class="input-group">
  <span class="input-group-addon">$</span>
  <input type="text" class="form-control" aria-label="Amount (to the nearest dollar)">
  <span class="input-group-addon">.00</span>
</div>
```

尺寸

为 `.input-group` 添加相应的尺寸类，其内部包含的元素将自动调整自身的尺寸。不需要为输入框组中的每个元素重复地添加控制尺寸的类。

```
<div class="input-group input-group-lg">
  <span class="input-group-addon" id="sizing-addon1">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-describedby="sizing-addon1">
</div>

<div class="input-group">
  <span class="input-group-addon" id="sizing-addon2">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-describedby="sizing-addon2">
</div>

<div class="input-group input-group-sm">
  <span class="input-group-addon" id="sizing-addon3">@</span>
  <input type="text" class="form-control" placeholder="Username" aria-describedby="sizing-addon3">
</div>
```

作为额外元素的多选框和单选框

可以将多选框或单选框作为额外元素添加到输入框组中。

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-addon">
        <input type="checkbox" aria-label="...">
      </span>
      <input type="text" class="form-control" aria-label="...">
    </div><!-- /input-group -->
  </div><!-- /.col-lg-6 -->
```

```
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-addon">
        <input type="radio" aria-label="...">
      </span>
      <input type="text" class="form-control" aria-label="...">
    </div><!-- /input-group -->
  </div><!-- /.col-lg-6 -->
</div><!-- /.row -->
```

作为额外元素的按钮

为输入框组添加按钮需要额外添加一层嵌套，不是 `.input-group-addon`，而是添加 `.input-group-btn` 来包裹按钮元素。由于不同浏览器的默认样式无法被统一的重新赋值，所以才需要这样做。

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-btn">
        <button class="btn btn-default" type="button">Go!</button>
      </span>
      <input type="text" class="form-control" placeholder="Search for...">
    </div><!-- /input-group -->
  </div><!-- /.col-lg-6 -->
  <div class="col-lg-6">
    <div class="input-group">
      <input type="text" class="form-control" placeholder="Search for...">
      <span class="input-group-btn">
        <button class="btn btn-default" type="button">Go!</button>
      </span>
    </div><!-- /input-group -->
  </div><!-- /.col-lg-6 -->
</div><!-- /.row -->
```

作为额外元素的按钮式下拉菜单

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <div class="input-group-btn">
        <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-
        <ul class="dropdown-menu">
```

```
          <li><a href="#">Action</a></li>
          <li><a href="#">Another action</a></li>
          <li><a href="#">Something else here</a></li>
          <li role="separator" class="divider"></li>
          <li><a href="#">Separated link</a></li>
        </ul>
      </div><!-- /btn-group -->
      <input type="text" class="form-control" aria-label="...">
    </div><!-- /input-group -->
  </div><!-- /.col-lg-6 -->
  <div class="col-lg-6">
    <div class="input-group">
      <input type="text" class="form-control" aria-label="...">
      <div class="input-group-btn">
        <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-
        <ul class="dropdown-menu dropdown-menu-right">
          <li><a href="#">Action</a></li>
          <li><a href="#">Another action</a></li>
          <li><a href="#">Something else here</a></li>
          <li role="separator" class="divider"></li>
          <li><a href="#">Separated link</a></li>
        </ul>
      </div><!-- /btn-group -->
    </div><!-- /input-group -->
  </div><!-- /.col-lg-6 -->
</div><!-- /.row -->
```

作为额外元素的分裂式按钮下拉菜单

```
<div class="input-group">
  <div class="input-group-btn">
    <!-- Button and dropdown menu -->
  </div>
  <input type="text" class="form-control" aria-label="...">
</div>

<div class="input-group">
  <input type="text" class="form-control" aria-label="...">
  <div class="input-group-btn">
    <!-- Button and dropdown menu -->
  </div>
</div>
```

导航

Bootstrap 中的导航组件都依赖同一个 `.nav` 类，状态类也是共用的。改变修饰类可以改变样式。

在标签页上使用导航需要依赖 JavaScript 标签页插件

由于标签页需要控制内容区的展示，因此，你必须使用 标签页组件的 JavaScript 插件。另外还要添加 role 和 ARIA 属性 - 详细信息请参考该插件的 实例。

确保导航组件的可访问性

如果你在使用导航组件实现导航条功能，务必在 `<ul>` 的最外侧的逻辑父元素上添加 `role="navigation"` 属性，或者用一个 `<nav>` 元素包裹整个导航组件。不要将 role 属性添加到 `<ul>` 上，因为这样可以被辅助设备（残疾人用的）上被识别为一个真正的列表。

标签页

注意 `.nav-tabs` 类依赖 `.nav` 基类。

```
<ul class="nav nav-tabs">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

胶囊式标签页

HTML 标记相同，但使用 `.nav-pills` 类：

```
<ul class="nav nav-pills">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

胶囊是标签页也是可以垂直方向堆叠排列的。只需添加 `.nav-stacked` 类。

```
<ul class="nav nav-pills nav-stacked">
  ...
</ul>
```

两端对齐的标签页

在大于 768px 的屏幕上，通过 `.nav-justified` 类可以很容易的让标签页或胶囊式标签呈现出同等宽度。在小屏幕上，导航链接呈现堆叠样式。

两端对齐的导航条导航链接已经被弃用了。

Safari 和响应式两端对齐导航

从 v8.0 版本开始，Safari 有一个 bug：对于两端对齐的导航，水平改变浏览器大小将引起绘制错误。此 bug 可以在两端对齐的导航实例中得到重现。

```
<ul class="nav nav-tabs nav-justified">
  ...
</ul>
<ul class="nav nav-pills nav-justified">
  ...
</ul>
```

禁用的链接

对任何导航组件（标签页、胶囊式标签页），都可以添加 `.disabled` 类，从而实现链接为灰色且没有鼠标悬停效果。

链接功能不受到影响

这个类只改变 `<a>` 的外观，不改变功能。可以自己写 JavaScript 禁用这里的链接。

```
<ul class="nav nav-pills">
  ...
  <li role="presentation" class="disabled"><a href="#">Disabled link</a></li>
  ...
</ul>
```

添加下拉菜单

用一点点额外 HTML 代码并加入下拉菜单插件的 JavaScript 插件即可。

带下拉菜单的标签页

```
<ul class="nav nav-tabs">
  ...
  <li role="presentation" class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false"
      Dropdown <span class="caret"></span>
    </a>
    <ul class="dropdown-menu">
      ...
    </ul>
  </li>
  ...
</ul>
```

带下拉菜单的胶囊式标签页

```
<ul class="nav nav-pills">
  ...
  <li role="presentation" class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false"
      Dropdown <span class="caret"></span>
    </a>
    <ul class="dropdown-menu">
      ...
    </ul>
  </li>
  ...
</ul>
```

导航条

默认样式的导航条

导航条是在您的应用或网站中作为导航页头的响应式基础组件。它们在移动设备上可以折叠（并且可开可关），且在视口（viewport）宽度增加时逐渐变为水平展开模式。

两端对齐的导航条导航链接已经被弃用了。

导航条内所包含元素溢出

由于 Bootstrap 并不知道你在导航条内放置的元素需要占据多宽的空间，你可能会遇到导航条中的内容折行的情况（也就是导航条占据两行）。解决办法如下：

- 减少导航条内所有元素所占据的宽度。
- 在某些尺寸的屏幕上（利用 响应式工具类）隐藏导航条内的一些元素。
- 修改导航条在水平排列和折叠排列互相转化时，触发这个转化的最小屏幕宽度值。可以通过修改 `@grid-float-breakpoint` 变量实现，或者自己重写相关的媒体查询代码，覆盖 Bootstrap 的默认值。

依赖 JavaScript 插件

如果 JavaScript 被禁用，并且视口（viewport）足够窄，致使导航条折叠起来，导航条将不能被打开，`.navbar-collapse` 内所包含的内容也将不可见。

响应式导航条依赖 collapse 插件，定制 Bootstrap 的话时候必将其包含。

修改视口的阈值，从而影响导航条的排列模式

当浏览器视口（viewport）的宽度小于 `@grid-float-breakpoint` 值时，导航条内部的元素变为折叠排列，也就是变现为移动设备展现模式；当浏览器视口（viewport）的宽度大于 `@grid-float-breakpoint` 值时，导航条内部的元素变为水平排列，也就是变现为非移动设备展现模式。通过调整源码中的这个值，就可以控制导航条何时堆叠排列，何时水平排列。默认值是 768px （小屏幕 -- 或者说是平板 --的最小值，或者说是平板）。

导航条的可访问性

务必使用 `<nav>` 元素，或者，如果使用的是通用的 `<div>` 元素的话，务必为导航条设置 `role="navigation"` 属性，这样能够让使用辅助设备的用户明确知道这是一个导航区域。

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-coll
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Brand</a>
    </div>

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
        <li><a href="#">Link</a></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="
          <ul class="dropdown-menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">Separated link</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">One more separated link</a></li>
          </ul>
        </li>
      </ul>
      <form class="navbar-form navbar-left" role="search">
        <div class="form-group">
          <input type="text" class="form-control" placeholder="Search">
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
      </form>
      <ul class="nav navbar-nav navbar-right">
```

```
        <li><a href="#">Link</a></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="
          <ul class="dropdown-menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">Separated link</a></li>
          </ul>
        </li>
      </ul>
    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>
```

品牌图标

将导航条内放置品牌标志的地方替换为 `<img>` 元素即可展示自己的品牌图标。由于 `.navbar-brand` 已经被设置了内补（padding）和高度（height），你需要根据自己的情况添加一些 CSS 代码从而覆盖默认设置。

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        <img alt="Brand" src="...">
      </a>
    </div>
  </div>
</nav>
```

表单

将表单放置于 `.navbar-form` 之内可以呈现很好的垂直对齐，并在较窄的视口（viewport）中呈现折叠状态。使用对齐选项可以规定其在导航条上出现的位置。

注意，`.navbar-form` 和 `.form-inline` 的大部分代码都一样，内部实现使用了 mixin。 某些表单组件，例如输入框组，可能需要设置一个固定宽度，从而在导航条内有合适的展现。

```
<form class="navbar-form navbar-left" role="search">
  <div class="form-group">
    <input type="text" class="form-control" placeholder="Search">
```

```
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

移动设备上的注意事项

在移动设备上，对于在 fixed 定位的元素内使用表单控件的情况有一些注意事项。请参考我们提供的浏览器支持情况相关的文档 。

为输入框添加 label 标签

如果你没有为输入框添加 label 标签，屏幕阅读器将会遇到问题。对于导航条内的表单，可以通过添加 `.sr-only` 类隐藏 label 标签。

按钮

对于不包含在 `<form>` 中的 `<button>` 元素，加上 `.navbar-btn` 后，可以让它在导航条里垂直居中。有一些对于为辅助设备提供可识别标签的方法，例如， aria-label、aria-labelledby 或者 title 属性。如果这些方法都没有，屏幕阅读器将使用 placeholder 属性（如果这个属性存在的话），但是请注意，使用 placeholder 代替其他识别标签的方式是不推荐的。

```
<button type="button" class="btn btn-default navbar-btn">Sign in</button>
```

基于情境的用法

就像标准的 按钮类 一样，`.navbar-btn` 可以被用在 `<a>` 和 `<input>` 元素上。然而，在 `.navbar-nav` 内，`.navbar-btn` 和标准的按钮类都不应该被用在 `<a>` 元素上。

文本

把文本包裹在 `.navbar-text` 中时，为了有正确的行距和颜色，通常使用 `<p>` 标签。

```
<p class="navbar-text">Signed in as Mark Otto</p>
```

非导航的链接

或许你希望在标准的导航组件之外添加标准链接，那么，使用 `.navbar-link` 类可以让链接有正确的默认颜色和反色设置。

```
<p class="navbar-text navbar-right">Signed in as <a href="#" class="navbar-link">Mark Otto</a></p>
```

组件排列

通过添加 `.navbar-left` 和 `.navbar-right` 工具类让导航链接、表单、按钮或文本对齐。两个类都会通过 CSS 设置特定方向的浮动样式。例如，要对齐导航链接，就要把它们放在个分开的、应用了工具类的 `<ul>` 标签里。

这些类是 `.pull-left` 和 `.pull-right` 的 mixin 版本，但是他们被限定在了媒体查询（media query）中，这样可以更容易的在各种尺寸的屏幕上处理导航条组件。

向右侧对齐多个组件

导航条目前不支持多个 `.navbar-right` 类。为了让内容之间有合适的空隙，我们为最后一个 `.navbar-right` 元素使用负边距（margin）。如果有多个元素使用这个类，它们的边距（margin）将不能按照你的预期正常展现。

我们将在 v4 版本中重写这个组件时重新审视这个功能。

固定在顶部

添加 `.navbar-fixed-top` 类可以让导航条固定在顶部，还可包含一个 `.container` 或 `.container-fluid` 容器，从而让导航条居中，并在两侧添加内补（padding）。

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    ...
  </div>
</nav>
```

需要为 body 元素设置内补（padding）

这个固定的导航条会遮住页面上的其它内容，除非你给 元素底部设置了 padding。用你自己的值，或用下面给出的代码都可以。提示：导航条的默认高度是 50px。

```
body { padding-top: 70px; }
```

固定在底部

添加 `.navbar-fixed-bottom` 类可以让导航条固定在底部，并且还可以包含一个 `.container` 或 `.container-fluid` 容器，从而让导航条居中，并在两侧添加内补（padding）。

```
<nav class="navbar navbar-default navbar-fixed-bottom">
  <div class="container">
    ...
  </div>
</nav>
```

需要为 body 元素设置内补（padding）

这个固定的导航条会遮住页面上的其它内容，除非你给 `<body>` 元素底部设置了 `padding`。用你自己的值，或用下面给出的代码都可以。提示：导航条的默认高度是 50px。

```
body { padding-bottom: 70px; }
```

静止在顶部

通过添加 `.navbar-static-top` 类即可创建一个与页面等宽度的导航条，它会随着页面向下滚动而消失。还可以包含一个 `.container` 或 `.container-fluid` 容器，用于将导航条居中对齐并在两侧添加内补（padding）。

通过添加 `.navbar-static-top` 类即可创建一个与页面等宽度的导航条，它会随着页面向下滚动而消失。还可以包含一个 `.container` 或 `.container-fluid` 容器，用于将导航条居中对齐并在两侧添加内补（padding）。

与 `.navbar-fixed-*` 类不同的是，你不用给 body 添加任何内补（padding）。

```
<nav class="navbar navbar-default navbar-static-top">
  <div class="container">
    ...
  </div>
</nav>
```

反色的导航条

通过添加 `.navbar-inverse` 类可以改变导航条的外观。

```
<nav class="navbar navbar-inverse">
  ...
</nav>
```

路径导航

在一个带有层次的导航结构中标明当前页面的位置。

各路径间的分隔符已经自动通过 CSS 的 `:before` 和 `content` 属性添加了。

```
<ol class="breadcrumb">
  <li><a href="#">Home</a></li>
  <li><a href="#">Library</a></li>
  <li class="active">Data</li>
</ol>
```

分页

为您的网站或应用提供带有展示页码的分页组件，或者可以使用简单的翻页组件。

默认分页

受 Rdio 的启发，我们提供了这个简单的分页组件，用在应用或搜索结果中超级棒。组件中的每个部分都很大，有点事容易点击、易缩放、点击区域大。

```
<nav>
  <ul class="pagination">
    <li>
      <a href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
      </a>
    </li>
    <li><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
    <li>
      <a href="#" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
      </a>
    </li>
  </ul>
</nav>
```

禁用和激活状态

链接在不同情况下可以定制。你可以给不能点击的链接添加 `.disabled` 类、给当前页添加 `.active` 类。

```
<nav>
  <ul class="pagination">
    <li class="disabled"><a href="#" aria-label="Previous"><span aria-hidden="true">&laquo;</span></a></li>
    <li class="active"><a href="#">1 <span class="sr-only">(current)</span></a></li>
    ...
  </ul>
</nav>
```

你还可以将 active 或 disabled 状态应用于 `<span>` 标签，或者在向前 / 向后的箭头处省略 `<a>` 标记，即替换 `<a>` 标签，这样就可以让其保持需要的样式而不能被点击。

```
<nav>
  <ul class="pagination">
    <li class="disabled">
      <span>
        <span aria-hidden="true">&laquo;</span>
      </span>
    </li>
    <li class="active">
      <span>1 <span class="sr-only">(current)</span></span>
    </li>
    ...
  </ul>
</nav>
```

尺寸

想要更小或更大的分页？`.pagination-lg` 或 `.pagination-sm` 类提供了额外可供选择的尺寸。

```
<nav><ul class="pagination pagination-lg">...</ul></nav>
<nav><ul class="pagination">...</ul></nav>
<nav><ul class="pagination pagination-sm">...</ul></nav>
```

翻页

用简单的标记和样式，就能做个上一页和下一页的简单翻页。用在像博客和杂志这样的简单站点上棒极了。

默认实例

在默认的翻页中，链接居中对齐。

```
<nav>
  <ul class="pager">
    <li><a href="#">Previous</a></li>
    <li><a href="#">Next</a></li>
  </ul>
</nav>
```

对齐链接

你还可以把链接向两端对齐：

```
<nav>
  <ul class="pager">
    <li class="previous"><a href="#"><span aria-hidden="true">&larr;</span> Older</a></li>
    <li class="next"><a href="#">Newer <span aria-hidden="true">&rarr;</span></a></li>
  </ul>
</nav>
```

可选的禁用状态

`.disabled` 类也可用于翻页中的链接。

```
<nav>
  <ul class="pager">
    <li class="previous disabled"><a href="#"><span aria-hidden="true">&larr;</span> Older</a></li>
    <li class="next"><a href="#">Newer <span aria-hidden="true">&rarr;</span></a></li>
  </ul>
</nav>
```

标签

实例

```
<h3>Example heading <span class="label label-default">New</span></h3>
```

可用的变体

用下面的任何一个类即可改变标签的外观。

```
<span class="label label-default">Default</span>
<span class="label label-primary">Primary</span>
<span class="label label-success">Success</span>
<span class="label label-info">Info</span>
<span class="label label-warning">Warning</span>
<span class="label label-danger">Danger</span>
```

如果标签数量很多怎么办？

如果你有大量的设置为 inline 属性的标签全部放在一个较窄的容器元素内，在页面上展示这些标签就会出现问题，每个标签就会有自己的一个 `inline-block` 元素（就像图标一样）。解决的办法是为每个标签都设置为 `display: inline-block;` 属性。关于这个问题以及实例，请参考 #13219 。

徽章

给链接、导航等元素嵌套 `<span class="badge">` 元素，可以很醒目的展示新的或未读的信息条目。

```
<a href="#">Inbox <span class="badge">42</span></a>

<button class="btn btn-primary" type="button">
  Messages <span class="badge">4</span>
</button>
```

如果没有新的或未读的信息条目，也就是说不包含任何内容，徽章组件能够自动隐藏（通过 CSS 的 :empty 选择符实现）。

跨浏览器兼容性

徽章组件在 Internet Explorer 8 浏览器中不会自动消失，因为 IE8 不支持 `:empty` 选择符。

适配导航元素的激活状态

Bootstrap 提供了内置的样式，让胶囊式导航内处于激活状态的元素所包含的徽章展示相匹配的样式。

```
<ul class="nav nav-pills" role="tablist">
  <li role="presentation" class="active"><a href="#">Home <span class="badge">42</span></a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages <span class="badge">3</span></a></li>
</ul>
```

巨幕

这是一个轻量、灵活的组件，它能延伸至整个浏览器视口来展示网站上的关键内容。

```
<div class="jumbotron">
  <h1>Hello, world!</h1>
  <p>...</p>
  <p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>
</div>
```

如果需要让巨幕组件的宽度与浏览器宽度一致并且没有圆角，请把此组件放在所有 `.container` 元素的外面，并在组件内部添加一个 `.container` 元素。

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

页头

页头组件能够为 h1 标签增加适当的空间，并且与页面的其他部分形成一定的分隔。它支持 h1 标签内内嵌 small 元素的默认效果，还支持大部分其他组件（需要增加一些额外的样式）。

```
<div class="page-header">
  <h1>Example page header <small>Subtext for header</small></h1>
</div>
```

缩略图

通过缩略图组件扩展 Bootstrap 的 栅格系统，可以很容易地展示栅格样式的图像、视频、文本等内容。

如果你想实现一个类似 Pinterest 的页面效果（不同高度和/宽度的缩略图顺序排列）的话，你需要使用一个第三方插件，比如 Masonry、Isotope 或 Salvattore。

默认样式的实例

Boostrap 缩略图的默认设计仅需最少的标签就能展示带链接的图片。

```
<div class="row">
  <div class="col-xs-6 col-md-3">
    <a href="#" class="thumbnail">
      <img src="..." alt="...">
    </a>
  </div>
  ...
</div>
```

自定义内容

添加一点点额外的标签，就可以把任何类型的 HTML 内容，例如标题、段落或按钮，加入缩略图组件内。

```
<div class="row">
  <div class="col-sm-6 col-md-4">
    <div class="thumbnail">
      <img src="..." alt="...">
      <div class="caption">
        <h3>Thumbnail label</h3>
        <p>...</p>
        <p><a href="#" class="btn btn-primary" role="button">Button</a> <a href="#" class="btn btn-default" role="butto
      </div>
    </div>
  </div>
</div>
```

警告框

警告框组件通过提供一些灵活的预定义消息，为常见的用户动作提供反馈消息。

实例

将任意文本和一个可选的关闭按钮组合在一起就能组成一个警告框，`.alert` 类是必须要设置的，另外我们还提供了有特殊意义的 4 个类（例如，`.alert-success`），代表不同的警告信息。

没有默认类

警告框没有默认类，只有基类和修饰类。默认的灰色警告框并没有多少意义。所以您要使用一种有意义的警告类。目前提供了成功、消息、警告或危险。

```
<div class="alert alert-success" role="alert">...</div>
<div class="alert alert-info" role="alert">...</div>
<div class="alert alert-warning" role="alert">...</div>
<div class="alert alert-danger" role="alert">...</div>
```

可关闭的警告框

为警告框添加一个可选的 `.alert-dismissible` 类和一个关闭按钮。

依赖警告框 JavaScript 插件

如果需要为警告框组件提供关闭功能，请使用 jQuery 警告框插件。

```
<div class="alert alert-warning alert-dismissible" role="alert">
  <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></
  <strong>Warning!</strong> Better check yourself, you're not looking too good.
</div>
```

保在所有设备上的正确行为

务必给 `<button>` 元素添加 `data-dismiss="alert"` 属性。

警告框中的链接

用 `.alert-link` 工具类，可以为链接设置与当前警告框相符的颜色。

```
<div class="alert alert-success" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-info" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-warning" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-danger" role="alert">
  <a href="#" class="alert-link">...</a>
</div>
```

进度条

通过这些简单、灵活的进度条，为当前工作流程或动作提供实时反馈。

跨浏览器兼容性

进度条组件使用了 CSS3 的 transition 和 animation 属性来完成一些特效。这些特性在 Internet Explorer 9 或以下版本中、Firefox 的老版本中没有被支持。Opera 12 不支持 animation 属性。

基本实例

默认样式的进度条

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 6
    <span class="sr-only">60% Complete</span>
  </div>
</div>
```

带有提示标签的进度条

将设置了 `.sr-only` 类的 `<span>` 标签从进度条组件中移除类，从而让当前进度显示出来。

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 6
    60%
  </div>
</div>
```

在展示很低的百分比时，如果需要让文本提示能够清晰可见，可以为进度条设置 `min-width` 属性。

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100" style="min-width
    0%
  </div>
</div>
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="2" aria-valuemin="0" aria-valuemax="100" style="min-width
    2%
```

```
  </div>
</div>
```

根据情境变化效果

进度条组件使用与按钮和警告框相同的类，根据不同情境展现相应的效果。

```
<div class="progress">
  <div class="progress-bar progress-bar-success" role="progressbar" aria-valuenow="40" aria-valuemin="0" aria-valuemax=
    <span class="sr-only">40% Complete (success)</span>
  </div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-info" role="progressbar" aria-valuenow="20" aria-valuemin="0" aria-valuemax="10
    <span class="sr-only">20% Complete</span>
  </div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-warning" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax=
    <span class="sr-only">60% Complete (warning)</span>
  </div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-danger" role="progressbar" aria-valuenow="80" aria-valuemin="0" aria-valuemax="
    <span class="sr-only">80% Complete (danger)</span>
  </div>
</div>
```

条纹效果

通过渐变可以为进度条创建条纹效果，IE9 及更低版本不支持。

```
<div class="progress">
  <div class="progress-bar progress-bar-success progress-bar-striped" role="progressbar" aria-valuenow="40" aria-valuem
    <span class="sr-only">40% Complete (success)</span>
  </div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-info progress-bar-striped" role="progressbar" aria-valuenow="20" aria-valuemin=
    <span class="sr-only">20% Complete</span>
  </div>
</div>
<div class="progress">
```

```
  <div class="progress-bar progress-bar-warning progress-bar-striped" role="progressbar" aria-valuenow="60" aria-valuen
    <span class="sr-only">60% Complete (warning)</span>
  </div>
</div>
<div class="progress">
  <div class="progress-bar progress-bar-danger progress-bar-striped" role="progressbar" aria-valuenow="80" aria-valuem
    <span class="sr-only">80% Complete (danger)</span>
  </div>
</div>
```

动画效果

为 `.progress-bar-striped` 添加 `.active` 类，使其呈现出由右向左运动的动画效果。IE9 及更低版本的浏览器不支持。

```
<div class="progress">
  <div class="progress-bar progress-bar-striped active" role="progressbar" aria-valuenow="45" aria-valuemin="0" aria-va
    <span class="sr-only">45% Complete</span>
  </div>
</div>
```

堆叠效果

把多个进度条放入同一个 `.progress` 中，使它们呈现堆叠的效果。

```
<div class="progress">
  <div class="progress-bar progress-bar-success" style="width: 35%">
    <span class="sr-only">35% Complete (success)</span>
  </div>
  <div class="progress-bar progress-bar-warning progress-bar-striped" style="width: 20%">
    <span class="sr-only">20% Complete (warning)</span>
  </div>
  <div class="progress-bar progress-bar-danger" style="width: 10%">
    <span class="sr-only">10% Complete (danger)</span>
  </div>
</div>
```

媒体对象

这是一个抽象的样式，用以构建不同类型的组件，这些组件都具有在文本内容的左或右侧对齐的图片（就像博客评论或 Twitter 消息等）。

默认样式

默认样式的媒体对象组件允许在一个内容块的左边或右边展示一个多媒体内容（图像、视频、音频）。

```
<div class="media">
  <div class="media-left">
    <a href="#">
      <img class="media-object" src="..." alt="...">
    </a>
  </div>
  <div class="media-body">
    <h4 class="media-heading">Media heading</h4>
    ...
  </div>
</div>
```

`.pull-left` 和 `.pull-right` 这两个类以前也曾经被用在了媒体组件上，但是，从 v3.3.0 版本开始，他们就不再被建议使用了。`.media-left` 和 `.media-right` 替代了他们，不同之处是，在 html 结构中， `.media-right` 应当放在 `.media-body` 的后面。

对齐

图片或其他媒体类型可以顶部、中部或底部对齐。默认是顶部对齐。

```
<div class="media">
  <div class="media-left media-middle">
    <a href="#">
      <img class="media-object" src="..." alt="...">
    </a>
  </div>
  <div class="media-body">
    <h4 class="media-heading">Middle aligned media</h4>
    ...
```

```
  </div>
</div>
```

媒体对象列表

用一点点额外的标记，就能在列表内使用媒体对象组件（对评论或文章列表很有用）。

```
<ul class="media-list">
  <li class="media">
    <div class="media-left">
      <a href="#">
        <img class="media-object" src="..." alt="...">
      </a>
    </div>
    <div class="media-body">
      <h4 class="media-heading">Media heading</h4>
      ...
    </div>
  </li>
</ul>
```

列表组

列表组是灵活又强大的组件，不仅能用于显示一组简单的元素，还能用于复杂的定制的内容。

基本实例

最简单的列表组仅仅是一个带有多个列表条目的无序列表，另外还需要设置适当的类。我们提供了一些预定义的样式，你可以根据自身的需求通过 CSS 自己定制。

```
<ul class="list-group">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

徽章

给列表组加入徽章组件，它会自动被放在右边。

```
<ul class="list-group">
  <li class="list-group-item">
    <span class="badge">14</span>
    Cras justo odio
  </li>
</ul>
```

链接

用 `<a>` 标签代替 `<li>` 标签可以组成一个全部是链接的列表组（还要注意的是，我们需要将 `<ul>` 标签替换为 `<div>` 标签）。没必要给列表组中的每个元素都加一个父元素。

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    Cras justo odio
  </a>
  <a href="#" class="list-group-item">Dapibus ac facilisis in</a>
```

```
  <a href="#" class="list-group-item">Morbi leo risus</a>
  <a href="#" class="list-group-item">Porta ac consectetur ac</a>
  <a href="#" class="list-group-item">Vestibulum at eros</a>
</div>
```

按钮

列表组中的元素也可以直接就是按钮（也同时意味着父元素必须是 `<div>` 而不能用 `<ul>` 了），并且无需为每个按钮单独包裹一个父元素。注意不要使用标准的 .btn 类！

```
<div class="list-group">
  <button type="button" class="list-group-item">Cras justo odio</button>
  <button type="button" class="list-group-item">Dapibus ac facilisis in</button>
  <button type="button" class="list-group-item">Morbi leo risus</button>
  <button type="button" class="list-group-item">Porta ac consectetur ac</button>
  <button type="button" class="list-group-item">Vestibulum at eros</button>
</div>
```

被禁用的条目

为 `.list-group-item` 添加 `.disabled` 类可以让单个条目显示为灰色，表现出被禁用的效果。

```
<div class="list-group">
  <a href="#" class="list-group-item disabled">
    Cras justo odio
  </a>
  <a href="#" class="list-group-item">Dapibus ac facilisis in</a>
  <a href="#" class="list-group-item">Morbi leo risus</a>
  <a href="#" class="list-group-item">Porta ac consectetur ac</a>
  <a href="#" class="list-group-item">Vestibulum at eros</a>
</div>
```

情境类

为列表中的条目添加情境类，默认样式或链接列表都可以。还可以为列表中的条目设置 `.active` 状态。

```
<ul class="list-group">
  <li class="list-group-item list-group-item-success">Dapibus ac facilisis in</li>
  <li class="list-group-item list-group-item-info">Cras sit amet nibh libero</li>
  <li class="list-group-item list-group-item-warning">Porta ac consectetur ac</li>
  <li class="list-group-item list-group-item-danger">Vestibulum at eros</li>
```

```
</ul>
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-success">Dapibus ac facilisis in</a>
  <a href="#" class="list-group-item list-group-item-info">Cras sit amet nibh libero</a>
  <a href="#" class="list-group-item list-group-item-warning">Porta ac consectetur ac</a>
  <a href="#" class="list-group-item list-group-item-danger">Vestibulum at eros</a>
</div>
```

定制内容

列表组中的每个元素都可以是任何 HTML 内容，甚至是像下面的带链接的列表组。

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    <h4 class="list-group-item-heading">List group item heading</h4>
    <p class="list-group-item-text">...</p>
  </a>
</div>
```

面版

虽然不总是必须，但是某些时候你可能需要将某些 DOM 内容放到一个盒子里。对于这种情况，可以试试面板组件。

基本实例

默认的 `.panel` 组件所做的只是设置基本的边框（border）和内补（padding）来包含内容。

```
<div class="panel panel-default">
  <div class="panel-body">
    Basic panel example
  </div>
</div>
```

带标题的面版

通过 `.panel-heading` 可以很简单地为面板加入一个标题容器。你也可以通过添加设置了 `.panel-title` 类的 `<h1>-<h6>` 标签，添加一个预定义样式的标题。

为了给链接设置合适的颜色，务必将链接放到带有 `.panel-title` 类的标题标签内。

```
<div class="panel panel-default">
  <div class="panel-heading">Panel heading without title</div>
  <div class="panel-body">
    Panel content
  </div>
</div>

<div class="panel panel-default">
  <div class="panel-heading">
    <h3 class="panel-title">Panel title</h3>
  </div>
  <div class="panel-body">
    Panel content
  </div>
</div>
```

带脚注的面版

把按钮或次要的文本放入 `.panel-footer` 容器内。注意面版的脚注不会从情境效果中继承颜色，因为他们并不是主要内容。

```
<div class="panel panel-default">
  <div class="panel-body">
    Panel content
  </div>
  <div class="panel-footer">Panel footer</div>
</div>
```

情境效果

像其他组件一样，可以简单地通过加入有情境效果的状态类，给特定的内容使用更针对特定情境的面版。

```
<div class="panel panel-primary">...</div>
<div class="panel panel-success">...</div>
<div class="panel panel-info">...</div>
<div class="panel panel-warning">...</div>
<div class="panel panel-danger">...</div>
```

带表格的面版

为面板中不需要边框的表格添加 `.table` 类，是整个面板看上去更像是一个整体设计。如果是带有 `.panel-body` 的面板，我们为表格的上方添加一个边框，看上去有分隔效果。

```
<div class="panel panel-default">
  <!-- Default panel contents -->
  <div class="panel-heading">Panel heading</div>
  <div class="panel-body">
    <p>...</p>
  </div>

  <!-- Table -->
  <table class="table">
    ...
  </table>
</div>
```

如果没有 `.panel-body` ，面版标题会和表格连接起来，没有空隙。

```
<div class="panel panel-default">
  <!-- Default panel contents -->
  <div class="panel-heading">Panel heading</div>

  <!-- Table -->
  <table class="table">
    ...
  </table>
</div>
```

带列表组的面版

可以简单地在任何面版中加入具有最大宽度的列表组。

```
<div class="panel panel-default">
  <!-- Default panel contents -->
  <div class="panel-heading">Panel heading</div>
  <div class="panel-body">
    <p>...</p>
  </div>

  <!-- List group -->
  <ul class="list-group">
    <li class="list-group-item">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Morbi leo risus</li>
    <li class="list-group-item">Porta ac consectetur ac</li>
    <li class="list-group-item">Vestibulum at eros</li>
  </ul>
</div>
```

具有响应式特性的嵌入内容

根据被嵌入内容的外部容器的宽度，自动创建一个固定的比例，从而让浏览器自动确定视频或 slideshow 的尺寸，能够在各种设备上缩放。

这些规则被直接应用在 `<iframe>`、`<embed>`、`<video>` 和 `<object>` 元素上。如果你希望让最终样式与其他属性相匹配，还可以明确地使用一个派生出来的 `.embed-responsive-item` 类。

超级提示： 不需要为 `<iframe>` 元素设置 `frameborder="0"` 属性，因为我们已经替你这样做了！

```
<!-- 16:9 aspect ratio -->
<div class="embed-responsive embed-responsive-16by9">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>

<!-- 4:3 aspect ratio -->
<div class="embed-responsive embed-responsive-4by3">
  <iframe class="embed-responsive-item" src="..."></iframe>
</div>
```

Well

默认效果

把 Well 用在元素上，能有嵌入（inset）的的简单效果。

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

可选类/样式

通过这两种可选修饰类，可以控制此组件的内补（padding）和圆角的设置。

```
<div class="well well-lg">...</div>
```

4

JavaScript 插件

概览

单个还是全部引入

JavaScript 插件可以单个引入（使用 Bootstrap 提供的单个 *.js 文件），或者一次性全部引入（使用 bootstrap.js 或压缩版的 bootstrap.min.js）。

建议使用压缩版的 JavaScript 文件

bootstrap.js 和 bootstrap.min.js 都包含了所有插件，你在使用时，只需选择一个引入页面就可以了。

插件之间的依赖关系

某些插件和 CSS 组件依赖于其它插件。如果你是单个引入每个插件的，请确保在文档中检查插件之间的依赖关系。注意，所有插件都依赖 jQuery （也就是说，jQuery必须在所有插件之前引入页面）。 bower.json 文件中列出了 Bootstrap 所支持的 jQuery 版本。

data 属性

你可以仅仅通过 data 属性 API 就能使用所有的 Bootstrap 插件，无需写一行 JavaScript 代码。这是 Bootstrap 中的一等 API，也应该是你的首选方式。

话又说回来，在某些情况下可能需要将此功能关闭。因此，我们还提供了关闭 data 属性 API 的方法，即解除以 data-api 为命名空间并绑定在文档上的事件。就像下面这样：

```
$(document).off('.data-api')
```

另外，如果是针对某个特定的插件，只需在 data-api 前面添加那个插件的名称作为命名空间，如下：

```
$(document).off('.alert.data-api')
```

编程方式的 API

我们为所有 Bootstrap 插件提供了纯 JavaScript 方式的 API。所有公开的 API 都是支持单独或链式调用方式，并且返回其所操作的元素集合（注：和 jQuery 的调用形式一致）。

```
$('.btn.danger').button('toggle').addClass('fat')
```

所有方法都可以接受一个可选的 option 对象作为参数，或者一个代表特定方法的字符串，或者什么也不提供（在这种情况下，插件将会以默认值初始化）：

```
$('#myModal').modal()                       // 以默认值初始化
$('#myModal').modal({ keyboard: false })    // initialized with no keyboard
$('#myModal').modal('show')                 // 初始化后立即调用 show 方法
```

每个插件还通过 `Constructor` 属性暴露了其原始的构造函数：`$.fn.popover.Constructor`。如果你想获取某个插件的实例，可以直接通过页面元素获取：`$('[rel="popover"]').data('popover')`。

默认设置

每个插件都可以通过修改其自身的 `Constructor.DEFAULTS` 对象从而改变插件的默认设置：

```
$.fn.modal.Constructor.DEFAULTS.keyboard = false // 将模态框插件的 `keyboard` 默认选参数置为 false
```

避免命名空间冲突

某些时候可能需要将 Bootstrap 插件与其他 UI 框架共同使用。在这种情况下，命名空间冲突随时可能发生。如果不幸发生了这种情况，你可以通过调用插件的 `.noConflict` 方法恢复其原始值。

```
var bootstrapButton = $.fn.button.noConflict() // return $.fn.button to previously assigned value
$.fn.bootstrapBtn = bootstrapButton            // give $().bootstrapBtn the Bootstrap functionality
```

事件

Bootstrap 为大部分插件所具有的动作提供了自定义事件。一般来说，这些事件都有不定式和过去式两种动词的命名形式，例如，不定式形式的动词（例如 show）表示其在事件开始时被触发；而过去式动词（例如 shown）表示在动作执行完毕之后被触发。

从 3.0.0 版本开始，所有 Bootstrap 事件的名称都采用命名空间方式。

所有以不定式形式的动词命名的事件都提供了 preventDefault 功能。这就赋予你在动作开始执行前将其停止的能力。

```
$('#myModal').on('show.bs.modal', function (e) {
  if (!data) return e.preventDefault() // 阻止模态框的展示
})
```

未对禁用 JavaScript 的浏览器提供补救措施

Bootstrap 插件未对禁用 JavaScript 的浏览器提供补救措施。如果你对这种情况下的用户体验很关心的话，请添加 `<noscript>` 标签向你的用户进行解释（并告诉他们如何启用 JavaScript），或者按照你自己的方式提供补救措施。

第三方工具库

Bootstrap 官方不提供对第三方 JavaScript 工具库的支持，例如 Prototype 或 jQuery UI。除了 `.noConflict` 和为事件名称添加命名空间，还可能会有兼容性方面的问题，这就需要你自己来处理了。

过渡效果

关于过渡效果

对于简单的过渡效果，只需将 transition.js 和其它 JS 文件一起引入即可。如果你使用的是编译（或压缩）版的 bootstrap.js 文件，就无需再单独将其引入了。

包含的内容

Transition.js 是针对 transitionEnd 事件的一个基本辅助工具，也是对 CSS 过渡效果的模拟。它被其它插件用来检测当前浏览器对是否支持 CSS 的过渡效果。

禁用过度效果

通过下面的 JavaScript 代码可以在全局范围禁用过渡效果，并且必须将此代码放在 transition.js （或 bootstrap.js 或 bootstrap.min.js）后面，确保在 js 文件加载完毕后再执行下面的代码：

```
$.support.transition = false
```

模态框

模态框经过了优化，更加灵活，以弹出对话框的形式出现，具有最小和最实用的功能集。

不支持同时打开多个模态框

千万不要在一个模态框上重叠另一个模态框。要想同时支持多个模态框，需要自己写额外的代码来实现。

模态框的 HTML 代码放置的位置

务必将模态框的 HTML 代码放在文档的最高层级内（也就是说，尽量作为 body 标签的直接子元素），以避免其他组件影响模态框的展现和/或功能。

对于移动设备的附加说明

这里提供了在移动设备上使用模态框有一些附加说明。请参考浏览器支持章节。

实例

静态实例

以下模态框包含了模态框的头、体和一组放置于底部的按钮。

```
<div class="modal fade">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</s
        <h4 class="modal-title">Modal title</h4>
      </div>
      <div class="modal-body">
        <p>One fine body&hellip;</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
```

```
    </div><!-- /.modal-content -->
  </div><!-- /.modal-dialog -->
</div><!-- /.modal -->
```

动态实例

点击下面的按钮即可通过 JavaScript 启动一个模态框。此模态框将从上到下、逐渐浮现到页面前。

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</s
        <h4 class="modal-title" id="myModalLabel">Modal title</h4>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

增强模态框的可访问性

务必为 `.modal` 添加 `role="dialog"` 和 `aria-labelledby="..."` 属性，用于指向模态框的标题栏；为 `.moda l-dialog` 添加 `aria-hidden="true"` 属性。

另外，你还应该通过 aria-describedby 属性为模态框 `.modal` 添加描述性信息。

可选尺寸

模态框提供了两个可选尺寸，通过为 `.modal-dialog` 增加一个样式调整类实现。

```
<!-- Large modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target=".bs-example-modal-lg">Large modal</butto

<div class="modal fade bs-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>

<!-- Small modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target=".bs-example-modal-sm">Small modal</butto

<div class="modal fade bs-example-modal-sm" tabindex="-1" role="dialog" aria-labelledby="mySmallModalLabel">
  <div class="modal-dialog modal-sm">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>
```

禁止动画效果

如果你不需要模态框弹出时的动画效果（淡入淡出效果），删掉 `.fade` 类即可。

```
<div class="modal" tabindex="-1" role="dialog" aria-labelledby="...">
  ...
</div>
```

用法

通过 data 属性或 JavaScript 调用模态框插件，可以根据需要动态展示隐藏的内容。模态框弹出时还会为 `<body>` 元素添加 `.modal-open` 类，从而覆盖页面默认的滚动行为，并且还会自动生成一个 `.modal-backdrop` 元素用于提供一个可点击的区域，点击此区域就即可关闭模态框。

通过 data 属性

不需写 JavaScript 代码也可激活模态框。通过在一个起控制器作用的元素（例如：按钮）上添加 `data-toggle="modal"` 属性，或者 `data-target="#foo"` 属性，再或者 `href="#foo"` 属性，用于指向被控制的模态框。

```
<button type="button" data-toggle="modal" data-target="#myModal">Launch modal</button>
```

通过 JavaScript 调用

只需一行 JavaScript 代码，即可通过元素的 id myModal 调用模态框：

```
$('#myModal').modal(options)
```

参数

可以将选项通过 data 属性或 JavaScript 代码传递。对于 data 属性，需要将参数名称放到 `data-` 之后，例如 `data-backdrop=""`。

名称	类型	默认值	描述
backdrop	boolean	true	为模态对话框添加一个背景元素。另外，为背景指定 `static` 时，点击模态对话框的外部区域不会将其关闭。
keyboard	boolean	true	按下esc键时关闭模态对话框
show	boolean	true	初始化时即显示模态对话框
remote	path	false	如果提供了远程url地址，就会通过 jQuery的 `load` 方法加载内容并注入到 `.modal-body` 中。如果你使用的是data属性api，你还可以使用 `href` 标签指定远程数据源。案例如下： `<a data-toggle="modal" href="remote.html" data-target="#modal">click me</a>`

方法

`.modal(options)`

将页面中的某块内容作为模态框激活。接受可选参数 object。

```
$('#myModal').modal({
  keyboard: false
})
```

`.modal('toggle')`

手动打开或关闭模态框。在模态框显示或隐藏之前返回到主调函数中（也就是，在触发 `shown.bs.modal` 或 `hidden.bs.modal` 事件之前）。

```
$('#myModal').modal('toggle')
```

`.modal('hide')`

手动隐藏模态框。在模态框隐藏之前返回到主调函数中 （也就是，在触发 `hidden.bs.modal` 事件之前）。

```
$('#myModal').modal('hide')
```

事件

Bootstrap 的模态框类提供了一些事件用于监听并执行你自己的代码。

事件类型	描述
show.bs.modal	`show` 方法调用之后立即触发该事件。如果是通过点击某个作为触发器的元素，则此元素可以通过事件的 `relatedTarget` 属性进行访问。
shown.bs.modal	此事件在模态框已经显示出来（并且同时在 CSS 过渡效果完成）之后被触发。如果是通过点击某个作为触发器的元素，则此元素可以通过事件的 `relatedTarget` 属性进行访问。
hide.bs.modal	`hide` 方法调用之后立即触发该事件。
hidden.bs.modal	此事件在模态框被隐藏（并且同时在 CSS 过渡效果完成）之后被触发。
loaded.bs.modal	从 `远端的数据源` 加载完数据之后触发该事件。

下拉菜单

调用方式

通过 data 属性

在链接或按钮上添加 data-toggle="dropdown" 即可切换下拉菜单。

```
<div class="dropdown">
  <a class="dropdown-toggle" data-toggle="dropdown" href="#">Dropdown trigger</a>
  <ul class="dropdown-menu" role="menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

为了保证 URL 符合规范，请使用 data-target 属性来代替 href="#"。

```
<div class="dropdown">
  <a class="dropdown-toggle" id="dLabel" role="button" data-toggle="dropdown" data-target="#" href="/page.html">
    Dropdown
    <b class="caret"></b>
  </a>
  <ul class="dropdown-menu" role="menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

通过 JavaScript

通过 JavaScript 调用下拉菜单：

```
$('.dropdown-toggle').dropdown()
```

滚动监听

案例

滚动监听插件可以根据滚动条的位置自动更新所对应的导航标记。你可以试试滚动这个页面，看看左侧导航的变化。

先把实现的代码上了，你可以通过测试代码先来看看效果。

```
<!DOCTYPE html>
 <html>
 <head>
 <title>Bootstrap</title>
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <!-- Bootstrap -->
 <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">
 <!--[if lt IE 9]>
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
 <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>
 <![endif]-->
     <style type="text/css"> .scrollspy-example
        {
            height: 200px;
            overflow: auto;
            position: relative;
            border:1px solid red;
        } </style>
 </head>
 <body>
<div class="container" >
      <nav id="navbar-example" class="navbar navbar-default navbar-static" role="navigation">
        <div class="navbar-header">
          <button class="navbar-toggle" type="button" data-toggle="collapse" data-target=".bs-js-navbar-scrollspy">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Project Name</a>
        </div>
        <div class="collapse navbar-collapse bs-js-navbar-scrollspy">
          <ul class="nav navbar-nav">
```

```
            <li class="active"><a href="#fat">@fat</a></li>
            <li><a href="#mdo">@mdo</a></li>
            <li class="dropdown">
              <a href="#" id="navbarDrop1" class="dropdown-toggle" data-toggle="dropdown">Dropdown <b class="caret"></b
              <ul class="dropdown-menu" role="menu" aria-labelledby="navbarDrop1">
                <li><a href="#one" tabindex="-1">one</a></li>
                <li><a href="#two" tabindex="-1">two</a></li>
                <li class="divider"></li>
                <li><a href="#three" tabindex="-1">three</a></li>
              </ul>
            </li>
          </ul>
        </div>
      </nav>
      <div data-offset="0" class="scrollspy-example" data-spy="scroll" data-target="#navbar-example">
        <h4 id="fat">@fat</h4>
        <p>Ad leggings keytar, brunch id art party dolor labore. Pitchfork yr enim lo-fi before they sold out qui. Tumb
        <h4 id="mdo">@mdo</h4>
        <p>Veniam marfa mustache skateboard, adipisicing fugiat velit pitchfork beard. Freegan beard aliqua cupidatat n
        <h4 id="one">one</h4>
        <p>Occaecat commodo aliqua delectus. Fap craft beer deserunt skateboard ea. Lomo bicycle rights adipisicing bar
        <h4 id="two">two</h4>
        <p>In incididunt echo park, officia deserunt mcsweeney's proident master cleanse thundercats sapiente veniam. M
        <h4 id="three">three</h4>
        <p>Ad leggings keytar, brunch id art party dolor labore. Pitchfork yr enim lo-fi before they sold out qui. Tumb
        <p>Keytar twee blog, culpa messenger bag marfa whatever delectus food truck. Sapiente synth id assumenda. Locav
      </div>
</div>
<script src="js/jquery-2.0.3.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

然后运行后，在内容下，也就是有滚动条哪里滚动鼠标齿轮，即可看到效果。

用法 1--通过 data 属性

通过为需要监听的页面元素（一般是 `<body>`）不过在上面添加在了 div 上面，你可以自己看看代码就明白了。然后给 div 添加属性 `data-spy="scroll"` 就可很轻松的为顶部导航条添加滚动监听功能。然后为其添加 data-target 属性，此属性的值为任何 Bootstrap 中 `.nav` 组件的父元素的 ID 或 class。

```
<div data-offset="0" class="scrollspy-example" data-spy="scroll" data-target="#navbar-example"> ..... </div>
```

导航链接地址必须有对应的目标

导航条内的链接地址必须有对应的页面元素具有同样的 ID 值。例如，`<a href="#home">home</a>` 必须对应 DOM 中例如 `<div id="home"></div>`。

用法 2--通过 JavaScript

通过 JavaScript 启动滚动监听：

```
<script type="text/javascript"> $(function () {
        $('.scrollspy-example').scrollspy({ target:'#navbar-example' });
      }) </script>
```

通过将样式类为 scrollspy-example 的 div，去掉它的 data-target 属性。这样同样可以进行鼠标滚轮的切换。

方法

`.scrollspy('refresh')`

使用滚动监听插件时，每当页面中从 DOM 中增加或删除页面元素时，都需要调用此方法以，如下：

```
$('[data-spy="scroll"]').each(function () { var $spy = $(this).scrollspy('refresh')
})
```

选项

可以将选项通过 data 属性或 JavaScript 传递。对于 data 属性，需要将选项名称放到 data- 之后，例如 `data-offset=""`。

名称	类型	默认值	描述
offset	number	10	Pixels to offset from top when calculating position of scroll.

事件

事件类型	描述
activate.bs.scrollspy	当滚动监听插件将某个元素置为active时，此事件被触发。

```
<script type="text/javascript"> $('#navbar-example').on('activate.bs.scrollspy', function () {
          alert(1);
    })
</script>
```

最后注意：针对滚动监听的内容当然要添加滚动条，也就是要预先添加样式。

```
<style type="text/css"> .scrollspy-example
        {
            height: 200px;
            overflow: auto;
            position: relative;
            border:1px solid red;
        } </style>
```

给与 Div 内容一定的高度。

标签页

之前通过组件只是简单的学习过这样的。

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#">Home</a></li>
  <li><a href="#">Profile</a></li>
  <li><a href="#">Messages</a></li>
</ul>
```

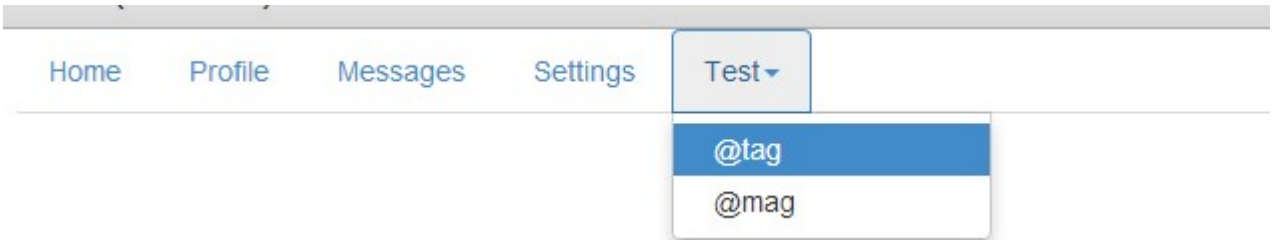
当然效果就是这样，只是默认的激活了第一个标签 Home，然后不能点击。

现在我们来优化一下。

我们给上面的先预定义一些 href 的标签 ID。

```
<ul class="nav nav-tabs">
     <li><a href="#home" >Home</a></li>
     <li><a href="#profile" >Profile</a></li>
     <li><a href="#messages" >Messages</a></li>
     <li><a href="#settings" >Settings</a></li>
     <li class="dropdown">
         <a href="#" data-toggle="dropdown" class="dropdown-toggle">Test<b class="caret"></b></a>
         <ul class="dropdown-menu">
           <li ><a href="#AAA">@tag</a></li>
           <li ><a href="#BBB">@mag</a></li>
       </ul>
     </li>
   </ul>
```

并且添加了一个下拉菜单。



然后现在我们继续的修正代码

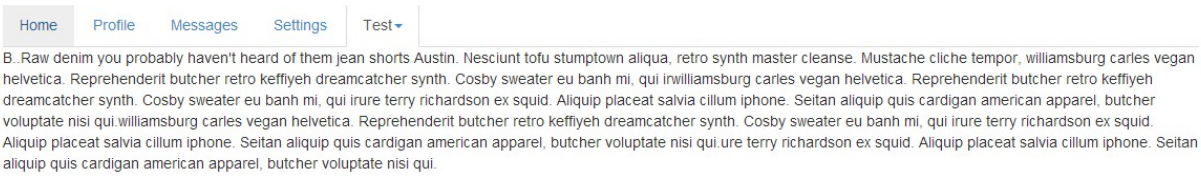
```
<ul class="nav nav-tabs">
      <li><a href="#home" data-toggle="tab">Home</a></li>
      <li><a href="#profile" data-toggle="tab">Profile</a></li>
      <li><a href="#messages" data-toggle="tab">Messages</a></li>
      <li><a href="#settings" data-toggle="tab">Settings</a></li>
      <li class="dropdown">
          <a href="#" data-toggle="dropdown" class="dropdown-toggle">Test<b class="caret"></b></a>
          <ul class="dropdown-menu">
            <li ><a data-toggle="tab" href="#AAA">@tag</a></li>
            <li ><a data-toggle="tab" href="#BBB">@mag</a></li>
        </ul>
      </li>
    </ul>

    <div class="tab-content">
      <div class="tab-pane active" id="home">1..Raw denim you probably haven't heard of them jean shorts Austin. Nesciu
    </div>
```

将标签页中的 a 标签都添加了一个属性 `data-toggle="tab"` 。

然后在下面添加一个 div 的容器，并给与 `tab-content` 的样式类。

容器里面定义 div，然后在 div 上添加 id 属性，和上面的 href 的标签 ID 对应，并添加 tab-pane 的样式类，其中一个如下，当然这个里面还添加了一个 active 的样式类，目的就是默认激活。

```
<div class="tab-pane active" id="home">
```



最终现在每个标签页都可以进行点击，并且下拉菜单的菜单想也是可以点击，对应着我们为 tab-content 中定义的标签内容页。

可以看出上面的操作我们都是通过 data 属性就可以实现标签切换和内容展示的。

下面我们就通过 JavaScript 来进行实现

用法

通过 JavaScript 启动可切换标签页（每个标签页单独被激活）：

```
$('#myTab a').click(function (e) {
  e.preventDefault()
  $(this).tab('show')
})
```

将所有代码贴上

```
<!DOCTYPE html>
 <html>
 <head>
 <title>Bootstrap</title>
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <!-- Bootstrap -->
 <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">
 <!--[if lt IE 9]>
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
 <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>
 <![endif]-->
 </head>
 <body>
 <div class="container">
    <ul class="nav nav-tabs" id="myTab">
      <li><a href="#home" >Home</a></li>
      <li><a href="#profile" >Profile</a></li>
      <li><a href="#messages" >Messages</a></li>
      <li><a href="#settings" >Settings</a></li>
      <li class="dropdown">
          <a href="#" data-toggle="dropdown" class="dropdown-toggle">Test<b class="caret"></b></a>
          <ul class="dropdown-menu">
            <li ><a href="#AAA">@tag</a></li>
            <li ><a href="#BBB">@mag</a></li>
        </ul>
      </li>
    </ul>

    <div class="tab-content">
      <div class="tab-pane active" id="home">1..Raw denim you probably haven't heard of them jean shorts Austin. Nesciu
    </div>
 <script src="js/jquery-2.0.3.min.js"></script>
 <script src="js/bootstrap.min.js"></script>
 <script type="text/javascript"> $('#myTab a').click(function (e) {
      e.preventDefault()
      $(this).tab('show')
    }) </script>
 </body>
 </html>
```

View Code

就是将前面的代码 a 标签中的 data-toggle 属性去掉，这样应该就找不到下面的 tab 内容了，所以内容无法进行切换。

不过我们可以通过上面的 JavaScript 进行点击切换实现。

可以有以下几种方式单独激活标签页：

```
$('#myTab a[href="#profile"]').tab('show')
$('#myTab a:first').tab('show')
$('#myTab a:last').tab('show')
$('#myTab li:eq(2) a').tab('show')
```

只需要添加相应的事件进行调用就可以了。

只需为页面元素简单的添加 `data-toggle="tab"` 或 `data-toggle="pill"` 就可以无需写任何 JavaScript 代码也能激活标签页或胶囊式导航。为 ul 添加 `.nav` 和 `.nav-tabs classe` 即可为其赋予 Bootstrap 标签页样式；而添加 nav 和 nav-pills class 可以为其赋予胶囊标签页。

可以通过 jQuery 来执行首次的加载

```
<script> $(function () {
    $('#myTab a:first').tab('show')
  }) </script>
```

事件

事件类型	描述
show.bs.tab	This event fires on tab show, but before the new tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.
shown.bs.tab	This event fires on tab show after a tab has been shown. Use `event.target` and `event.relatedTarget` to target the active tab and the previous active tab (if available) respectively.

```
$('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {
  e.target //通过此参数可以获得激活的tab信息  e.relatedTarget // 激活之前的那一个tab的信息 })
```

工具提示

受到 Jason Frame 开发的 jQuery.tipsy 插件的启发，Bootstrap 才把这个工具提示插件做的更好，而且此插件不依赖图片，只是使用 CSS3 来实现动画效果，并使用 data 属性存储标题。

将鼠标悬停在按钮、文本框、链接等等一些基本控件上就可以看到提示了，先上一个效果图

主要实现了按钮的上下左右的 ToolTip，然后是文本框和链接的 ToolTip。

```
<div class="container" style="margin-top:40px;"> <div class="bs-example tooltip-demo"> <div class="bs-example-tooltips
```

在按钮上添加 `data-toggle="tooltip"` 的属性来开启它的工具提示功能，然后指定 `data-placement="left"` 工具提示的位置，可以是上下左右（top、bottom、left、right）。 选择性加入的功能

出于性能方面的考虑，工具提示和弹框组件的 data 属性 api 是选择性加入的，也就是说你必须自己初始化他们。

因此针对上述六个控件，我们需要初始化它们，我们是通过 jQuery 的初始化事件进行的

```
 <script type="text/javascript"> $(function() {
     $('.tooltip-demo').tooltip({
      selector: "[data-toggle=tooltip]",
      container: "body" })
    $("#testt").tooltip({})
    $('a').tooltip()
}) </script>
```

按钮、文本框、链接三种不同的控件的实现初始化稍微有点不同。

工具提示与按钮组和输入框组联合使用时需要一些特殊设置

在 `.btn-group` 或 `.input-group` 内的元素上使用工具提示时，你需要指定 `container: 'body'` 选项以避免不需要的副作用（例如，当工具提示显示之后，与其合作的页面元素可能变得更宽或是去圆角）。

在禁止使用的页面元素上使用工具提示时需要额外增加一个元素将其包裹起来

为了给 disabled 或 `.disabled` 元素添加工具提示，将需要增加工具提示的页面元素包裹在一个 `<div>` 中，然后对这个 `<div>` 元素应用工具提示。

用法

通过 JavaScript 激活工具提示：上面也已经使用过了。

```
$('#example').tooltip(options)
```

选项

名称	类型	默认值	描述
animation	boolean	true	apply a CSS fade transition to the tooltip
html	boolean	false	Insert HTML into the tooltip. If false, jQuery's `text` method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
placement	string | function	'top'	how to position the tooltip - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the tooltip. For example, if placement is "auto left", the tooltip will display to the left when possible, otherwise it will display right.
selector	string	false	If a selector is provided, tooltip objects will be delegated to the specified targets.
title	string | function	''	default title value if `title` attribute isn't present
trigger	string	'hover focus'	how tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space.
delay	number | object	0	delay showing and hiding the tooltip (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: `delay: { show: 500, hide: 100 }`
container	string | false	false	Appends the tooltip to a specific element. Example: `container: 'body'`

对单个工具提示使用 data 属性。

使用 data 属性可以为单个工具提示指定额外选项，如下所示。

标记

```
<a href="#" data-toggle="tooltip" title="first tooltip">Hover over me</a>
```

方法

`$().tooltip(options)` ——为一组元素应用工具提示。

`.tooltip('show')` ——展示工具提示。

```
$('#element').tooltip('show')
```

`.tooltip('hide')` ——隐藏工具提示。

```
$('#element').tooltip('hide')
```

`.tooltip('toggle')` ——展示或隐藏工具提示。

```
$('#element').tooltip('toggle')
```

`.tooltip('destroy')` ——隐藏并销毁工具提示。

```
$('#element').tooltip('destroy')
```

事件

事件类型	描述
show.bs.tooltip	当 `show` 方法被调用之后，此事件将被立即触发。
shown.bs.tooltip	当工具提示展示到用户面前之后（同时CSS过渡效果执行完之后）此事件被触发。
hide.bs.tooltip	当 `hide` 方法被调用之后，此事件被触发。
hidden.bs.tooltip	当工具提示被隐藏之后（同时CSS过渡效果执行完之后），此事件被触发。

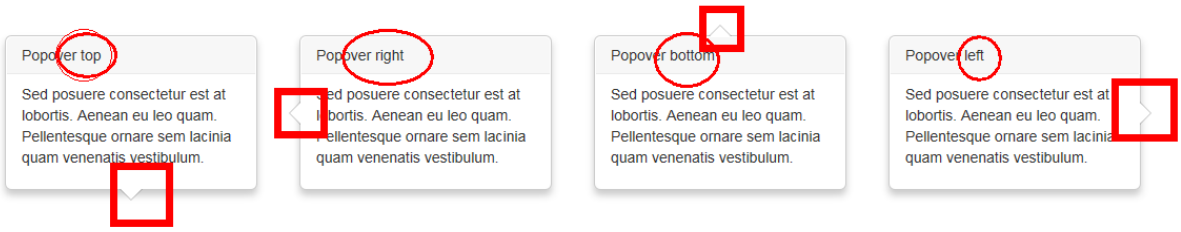
```
$('.tooltip-demo').on('hidden.bs.tooltip', function () {
        alert(1);
    })
```

弹出框

案例

为页面内容添加一个小的覆盖层，就像 iPad 上的效果一样，为页面元素增加额外的信息。

先来看几个简单的静态案例效果图



效果比较简单主要就是静态的弹出的小窗体，分为窗体标题和窗体内容。

```
<div class="bs-example bs-example-popover">
      <div class="popover top">
        <div class="arrow"></div>
        <h3 class="popover-title">Popover top</h3>
        <div class="popover-content">
          <p>Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenat
        </div>
      </div>

      <div class="popover right">
        <div class="arrow"></div>
        <h3 class="popover-title">Popover right</h3>
        <div class="popover-content">
          <p>Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenat
        </div>
      </div>

      <div class="popover bottom">
        <div class="arrow"></div>
        <h3 class="popover-title">Popover bottom</h3>

        <div class="popover-content">
          <p>Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenat
        </div>
```

```
      </div>

      <div class="popover left">
        <div class="arrow"></div>
        <h3 class="popover-title">Popover left</h3>
        <div class="popover-content">
          <p>Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenati
        </div>
      </div>

      <div class="clearfix"></div>
    </div>
```

但是我们还是需要给元素设置简单的基本布局

```
<style type="text/css"> .bs-example-popover .popover {
      position: relative;
      display: block; float: left;
      width: 240px;
      margin: 20px;
} </style>
```

动态演示

先来看效果图

一个按钮，点击按钮的时候就会弹出右侧的小窗体。

看代码，其实也很简单。

```
<a id="a2" class="btn btn-lg btn-danger" data-placement="right" data-content="text" title="" href="#" data-original-t
```

就一个 a 标签，但是赋予了按钮的样式类，然后给与几个属性，主要用于展示弹出框的：

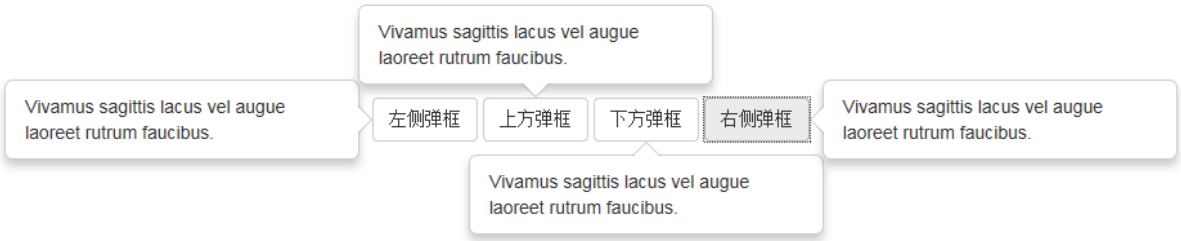
- 第一个：`data-original-title` ——标题
- 第二个：`data-content` ——内容
- 第三个：`data-placement` ——位置（上下左右top、bottom、left、right）

不过现在如果你来运行，按钮是有了，你点击按钮弹出框被不会出现，原来很简单，就是我们还没有给它初始化，就像上一节中的 tooltip 一样的。

只需要添加简单的 JavaScript 代码就可以了。

```
<script type="text/javascript"> $("#a1").popover(); </script>
```

四个方向




```
<div  style="margin-left:200px;margin-top:100px;margin-bottom:200px;" class="bs-example tooltip-demo">
      <div class="bs-example-tooltips">
        <button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="left"
        <button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="top"
        <button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="botto
        <button type="button" class="btn btn-default" data-container="body" data-toggle="popover" data-placement="right
      </div>
    </div>
```

然后用 JavaScript 来激活

```
<script type="text/javascript"> $("#a1").popover();
      $("[data-toggle=popover]").popover(); </script>
```

选择性加入的功能

出于性能方面的考虑，工具提示和弹框组件的 data 属性 api 是选择性加入的，也就是说你必须自己初始化他们。

弹出框在按钮组和输入框组中使用时，需要额外的设置

当提示框与 `.btn-group` 或 `.input-group` 联合使用时，你需要指定 `container: 'body'` 选项（见下面的文档）以避免不需要的副作用（例如，当弹出框显示之后，与其合作的页面元素可能变得更宽或是去圆角）。

在禁止使用的页面元素上使用弹出框时需要额外增加一个元素将其包裹起来

为了给 disabled 或 `.disabled` 元素添加弹出框时，将需要增加弹出框的页面元素包裹在一个 `<div>` 中，然后对这个 `<div>` 元素应用弹出框。

用法

通过 JavaScript 启用弹出框：

```
$('#example').popover(options)
```

选项

可以将选项通过 data 属性或 JavaScript 传递。对于 data 属性，需要将选项名称放到 data- 之后，例如 `data-animation=""`。

方法

```
$().popover(options)
```

为一组元素初始化弹出框。

```
$('#element').popover('show')
```

显示弹出框。

```
$('#element').popover('hide')
```

隐藏弹出框。

```
$('#element').popover('toggle')
```

展示或隐藏弹出框。

```
$('#element').popover('destroy')
```

隐藏并销毁弹出框。

事件

事件类型	描述
show.bs.popover	当 `show` 方法被调用之后，此事件将被立即触发。
shown.bs.popover	当弹出框展示到用户面前之后（同时CSS过渡效果执行完之后）此事件被触发。
hide.bs.popover	当 `hide` 方法被调用之后，此事件被触发。
hidden.bs.popover	当弹出框被隐藏之后（同时CSS过渡效果执行完之后），此事件被触发。

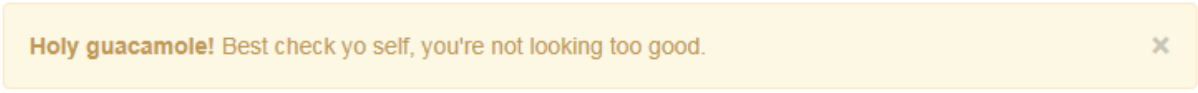
```
$('[data-toggle=popover]').on('shown.bs.popover', function () {
          alert(1);
      })
```

警告框

案例

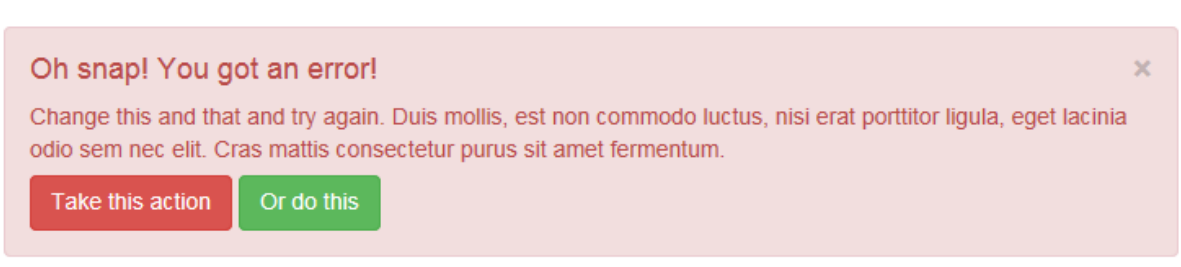
通过这个插件可以为所有警告框增加关闭功能。

```
<div id="alert1" class="alert alert-warning fade in">
       <button id='alert1' type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</button>
       <strong>Holy guacamole!</strong> Best check yo self, you're not looking too good. </div>
```



再来一个小例子

```
<div class="alert alert-danger fade in">
       <button type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</button>
       <h4>Oh snap! You got an error!</h4>
       <p>Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget la
       <p>
         <button type="button" class="btn btn-danger">Take this action</button>
         <button type="button" class="btn btn-success">Or do this</button>
       </p>
     </div>
```



通过 `data-dismiss` 属性即可开始关闭警告框的功能。无须任何的 JavaScript 的代码。只需为关闭按钮设置 `data-dismiss="alert"` 即可自动为警告框赋予关闭功能。

用法

如果通过 JavaScript 启用警告框关闭功能：

我们来修改一下第一简单的小例子

```
<div id="alert1" class="alert alert-warning fade in">
       <button id='alert1' type="button" class="close" onclick="Test()" aria-hidden="true">&times;</button>
       <strong>Holy guacamole!</strong> Best check yo self, you're not looking too good. </div>
```

我们主要是去掉了关闭按钮的 data-dismiss 属性，然后添加了一个 onclick 的单击按钮事件，也就是关闭警示框的事件。

来看一下如何通过 JavaScript 来关闭警示框

```
<script type="text/javascript"> function Test()
 {
    $("#alert1").alert('close');
 } </script>
```

为所有警告框加入关闭功能。如果希望警告框被关闭时呈现动画效果，请确保为其添加了 `.fade` 和 `.in`。

事件

Bootstrap 中的警告框暴露了一组事件，允许你进行监听。

事件类型	描述
close.bs.alert	当 `close` 函数被调用之后，此事件被立即触发。
closed.bs.alert	当警告框被关闭之后（同时CSS过渡效果执行完毕），此事件被触发。

```
<script type="text/javascript"> $('#alert1').bind('close.bs.alert', function () {
         alert(1);
   }) </script>
```

添加以上代码之后，再点击关闭按钮的时候会先执行 function 里面的代码，然后再关闭警示框的。

按钮

按钮可以完成很多工作。控制按钮状态或创建按钮组可以产生类似工具条之类的复杂组件。

状态

通过添加 `data-loading-text="` 正在加载..."可以使按钮呈现加载状态。

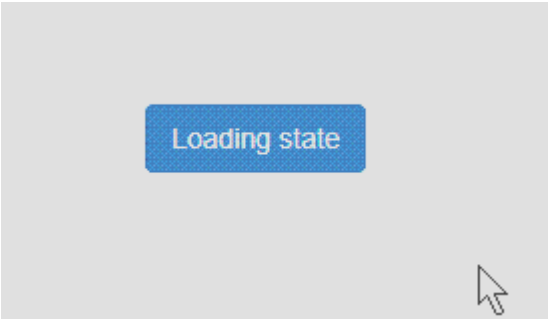
```
<button type="button" id="fat-btn" data-loading-text="正在加载..." class="btn btn-primary"> Loading state </button>
```

不过如果想启用加载的状态，还需要在点击按钮的时候进行手动启动。

```
<script type="text/javascript"> $('#fat-btn').click(function () { var btn = $(this)
       btn.button('loading')
       setTimeout(function () {
         btn.button('reset')
       }, 3000)
     }) </script>
```

上面的 JavaScript 代码，首先是设置让按钮处于加载状态，然后通过 SetTimeout 定时三秒中定义事件，让按钮回到初始化状态。

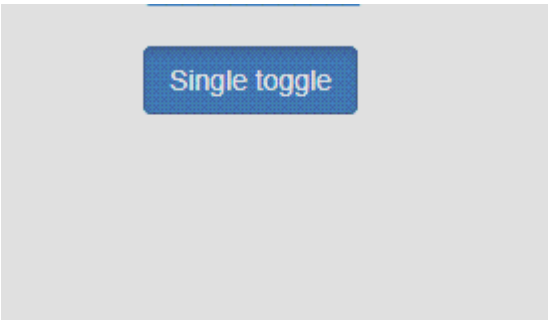
然后看一下点击按钮之后的效果



状态切换

通过添加 `data-toggle="button"` 可以让按钮能够切换状态。

```
<button type="button" class="btn btn-primary" data-toggle="button">Single toggle</button>
```

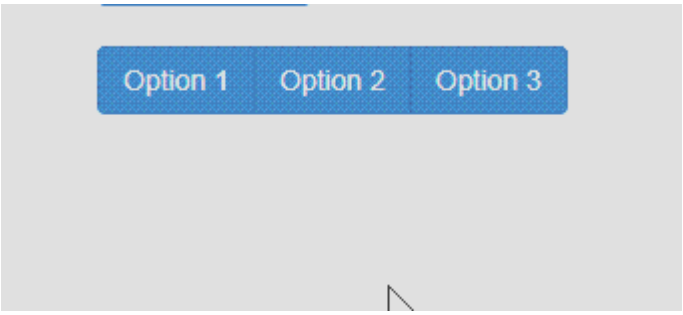


感觉按钮的颜色还是有些许的变化的。

选择框

通过向按钮组添加 `data-toggle="buttons"` 可以使按钮组具有类似选择框的选择/取消功能。

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary">
    <input type="checkbox"> Option 1 </label>
  <label class="btn btn-primary">
    <input type="checkbox"> Option 2 </label>
  <label class="btn btn-primary">
    <input type="checkbox"> Option 3 </label>
</div>
```

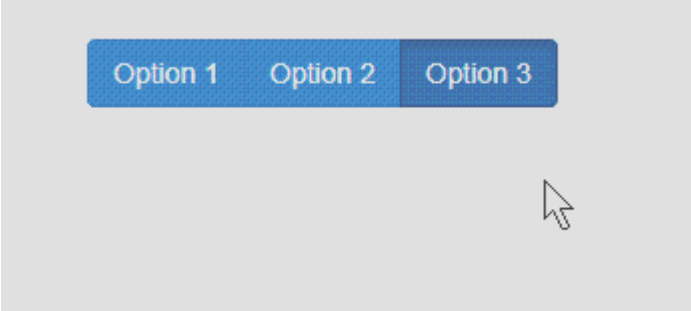


单选

通过向按钮组添加 `data-toggle="buttons"` 可以让按钮组具有单选框的功能。

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary">
    <input type="radio" name="options" id="option1"> Option 1 </label>
  <label class="btn btn-primary">
    <input type="radio" name="options" id="option2"> Option 2 </label>
  <label class="btn btn-primary">
```

```
    <input type="radio" name="options" id="option3"> Option 3 </label>
</div>
```



这个可只有单选的效果吧。

用法

```
<button class="btn btn-primary" id="btn" data-toggle="button">单独开关</button>
    <a href="javascript:void(0)" class="btn btn-success" onClick="_switch()">切换</a>
```

上面我们只是通过 data 属性来切换按钮的状态，现在我们来用 JavaScript 来实现。

```
<script type="text/javascript"> function Switch()
      {
          $("#btn").button('toggle');
          } </script>
```

标记

按钮插件完整支持 data 属性。通过下面的案例可以看到各种使用方式。

方法

```
$().button('toggle')
```

切换按钮状态。赋予按钮被激活时的状态和外观。

自动切换

可以使用 data-toggle 属性让按钮具有自动切换状态的能力。

```
<button type="button" class="btn" data-toggle="button">...</button>
```

上面已经有实例了，在此我就简单的举个例子。

```
$().button('loading')
```

设置按钮状态为 loading - 即将按钮置为禁用状态并将文字内容切换为 loading。通过使用 `data-loading-text` 可以在按钮元素上定义 loading 文本。

```
<button type="button" class="btn" data-loading-text="loading stuff...">...</button>
```

这个效果上面也有了。

跨浏览器兼容性

Firefox 会在多个页面加载之间保持按钮的禁用状态。可以通过添加 `autocomplete="off"` 来解决提到的问题。

```
$().button('reset')
```

重置按钮状态 - 并将按钮上的文本还原为原始值。

```
$().button(string)
```

重置按钮状态 - 并将按钮上的文本重置为传入的值。

```
<button type="button" class="btn" data-complete-text="finished!" >...</button>
<script> $('.btn').button('complete') </script>
```

折叠

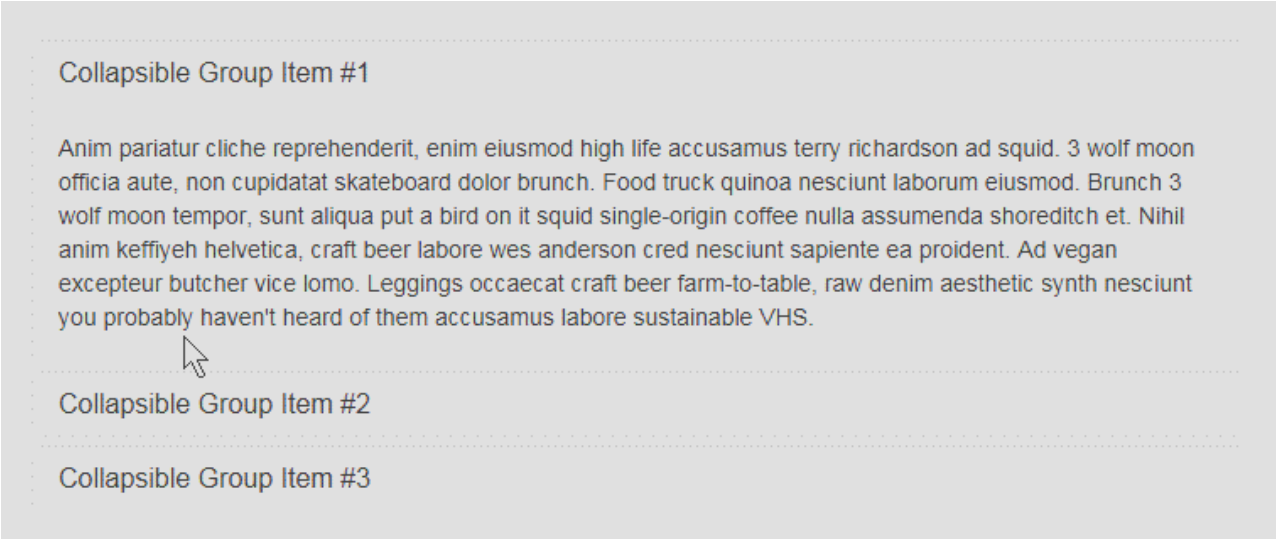
对为支持折叠功能的组件，例如 accordions 和导航，赋予基本样式和灵活的支持。

- 插件依赖
- 折叠插件依赖过渡效果插件。

案例

使用折叠插件，通过扩展 panel 组件从而构建了一个简单的 accordion 组件。

先来看一下效果。



接下来看一下代码的实现。

```
<div class="container" style="margin-top:140px;">
     <div class="panel-group" id="accordion">
      <div class="panel panel-default">
        <div class="panel-heading">
          <h4 class="panel-title">
            <a data-toggle="collapse" data-toggle="collapse" data-parent="#accordion" href="#collapseOne"> Collapsible
          </h4>
        </div>
        <div id="collapseOne" class="panel-collapse collapse in">
          <div class="panel-body"> Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardso
        </div>
      </div>
```

```
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a data-toggle="collapse" data-toggle="collapse" data-parent="#accordion" href="#collapseTwo"> Collapsible
        </h4>
      </div>
      <div id="collapseTwo" class="panel-collapse collapse">
        <div class="panel-body"> Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richards
      </div>
    </div>
    <div class="panel panel-default">
      <div class="panel-heading">
        <h4 class="panel-title">
          <a data-toggle="collapse" data-toggle="collapse" data-parent="#accordion" href="#collapseThree"> Collapsib
        </h4>
      </div>
      <div id="collapseThree" class="panel-collapse collapse">
        <div class="panel-body"> Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richards
      </div>
    </div>
  </div>
</div>
```

- 第一步：首先最外面那层 panel-group 这层下面包括几个小组。
- 第二步：看一下几个简单的组

```
<div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-toggle="collapse" data-parent="#accordion" href="#collapseOne"> Collapsible Grou
      </h4>
    </div>
    <div id="collapseOne" class="panel-collapse collapse in">
      <div class="panel-body">
      </div>
    </div>
</div>
```

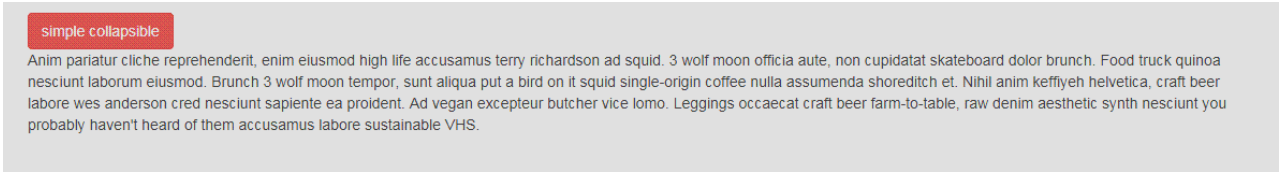
通过代码也比较清楚的可以看出一个 panel 的结构。

`panel-header` 和 `pandl-body`，然后 `panel-header` 里面可以包含标题，链接。通过链接和 `panel-body` 相连。 - 第三步：你可以发现在 `panel-group` 中有一个 `id="accordion"`，然后下面每个标题下链接中有个`data-parent="#accordion"` 。

如果去掉的话，那么效果就是点击其他链接后，原来的 panel 并不会再缩起来了。

你可以通过另一个方式来展示折叠的效果。

```
<div class="container" style="margin-top:140px;">
 <button type="button" class="btn btn-danger" data-toggle="collapse" data-target="#demo"> simple collapsible </button>

<div id="demo" class="collapse in">Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardso
```



用法

折叠插件通过几个简单的类来控制样式

- .collapse 隐藏内容
- .collapse in 显示内容
- .collapsing。 折叠被添加后过渡效果就有了，然后如果被移除了它就结束了。

通过 data 属性

仅仅通过向页面元素添加 `data-toggle="collapse"` 和 `data-target` 就可以为其赋予控制可折叠页面元素的能力。`data-target` 属性接受一个CSS选择器作为其控制对象。请确保为可折叠页面元素添加 `collapse class` 。如果你希望可折叠页面元素的默认状态是展开的，请添加in class。

为了给一组可折叠页面元素添加控制器，添加 `data-parent="#selector"` 即可。请参考上面的例子即可。

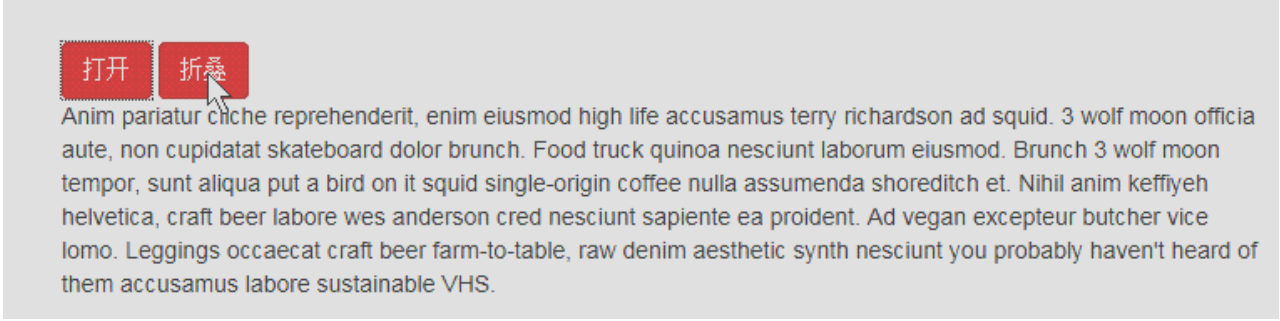
通过 JavaScript

```
<button type="button" class="btn btn-danger" onClick="Open()">打开</button>
<button type="button" class="btn btn-danger" onClick="Hide()">折叠</button>

<div id="demo" class="collapse in">Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardso
```

```
<script type="text/javascript"> $(function(){
         $("#demo").collapse({
    toggle: false })
        })
     function Open(){
```

```
        $("#demo").collapse("show");
        }
    function Hide(){
        $("#demo").collapse("hide");
        } </script>
```

来看一下上面的效果



方法

赋予页面元素可折叠功能。接受一个可选的 object 作为参数。

```
$("#demo").collapse({toggle: false})
```

这样元素在初始化的时候会是展开的。

1. collapse('toggle')展示或隐藏一个可折叠的页面元素。
2. collapse('show')展示一个可折叠页面元素。
3. collapse('hide')隐藏一个可折叠页面元素。

事件

Bootstrap 中的折叠插件对外暴露了一组可以监听的事件。

事件类型	描述
show.bs.collapse	当 show 方法被调用之后立即触发此事件。
shown.bs.collapse	当可折叠页面元素显示出来之后（同时CSS过渡效果也已执行完毕），此事件被触发。
hide.bs.collapse	当 hide 方法被调用之后，此事件被立即触发。
hidden.bs.collapse	当可折叠页面元素隐藏之后（同时CSS过渡效果也已执行完毕），此事件被触发。

```
$('#demo').on('hidden.bs.collapse', function () {
          alert(1);
    })
```

这样就为元素绑定了隐藏时的事件。

轮播

下面就是一个轮播组件的案例。

代码：

```
<body style="width:900px; margin-left:auto; margin-right:auto;">
    <div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
      <!-- Indicators -->
      <ol class="carousel-indicators">
        <li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
        <li data-target="#carousel-example-generic" data-slide-to="1"></li>
        <li data-target="#carousel-example-generic" data-slide-to="2"></li>
      </ol>

      <!-- Wrapper for slides -->
      <div class="carousel-inner" style="text-align:center">
        <div class="item active">
          <img alt="First slide" src="http://placehold.it/900x500/78EB09/FFEB09/&text=First slide" ></img>
        </div>
        <div class="item">
          <img alt="Second slide" src="http://placehold.it/900x500/78EB09/FFEB09/&text=Second slide" ></img>
        </div>
        <div class="item">
```

```
          <img alt="Third slide" src="http://placehold.it/900x500/78EB09/FFEB09/&text=Third slide"  ></img>
        </div>
      </div>

      <!-- Controls -->
      <a class="left carousel-control" href="#carousel-example-generic" data-slide="prev">
        <span class="glyphicon glyphicon-chevron-left"></span>
      </a>
      <a class="right carousel-control" href="#carousel-example-generic" data-slide="next">
        <span class="glyphicon glyphicon-chevron-right"></span>
      </a>
    </div>
 <script src="js/jquery-2.0.3.min.js"></script>
 <script src="js/bootstrap.min.js"></script>
 <script type="text/javascript">
//$('.carousel').carousel('next');
 </script>
 </body>
```

Internet Explorer 8 & 9 不支持过渡动画效果

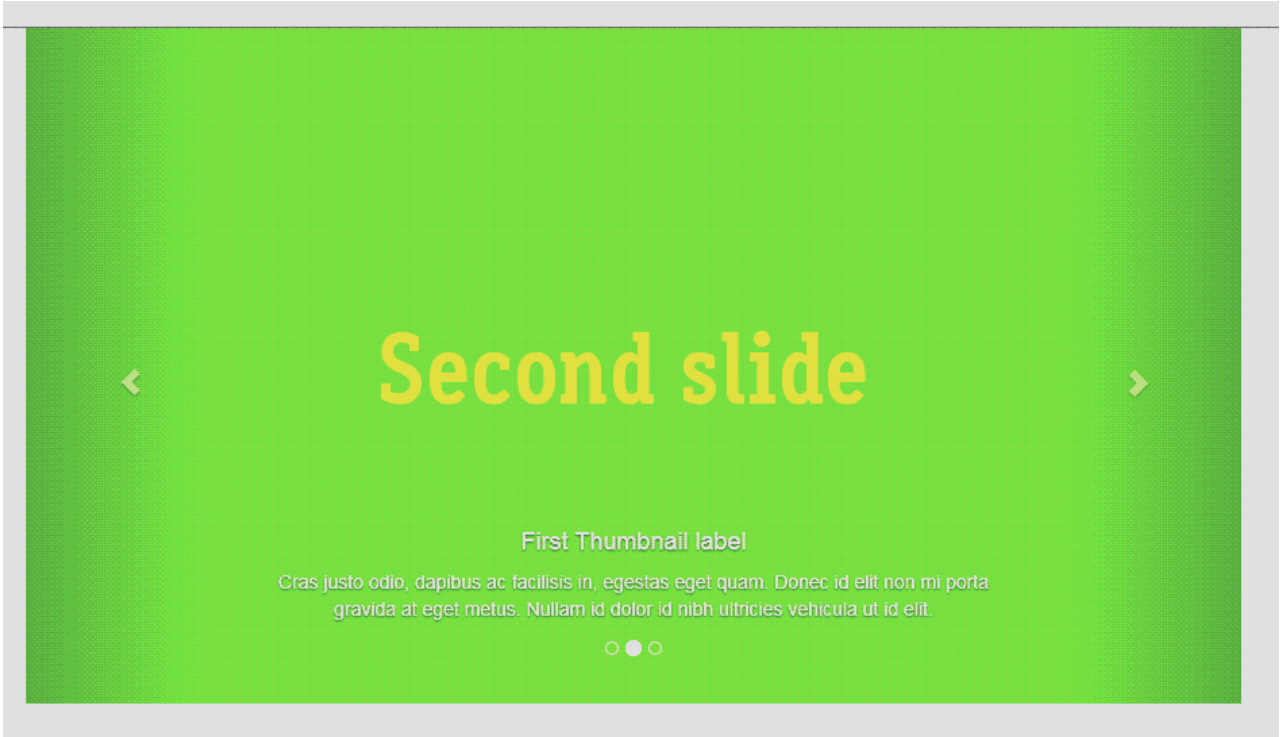
Bootstrap 基于 CSS3 实现动画效果，但是 Internet Explorer 8 & 9 不支持这些必要的 CSS 属性。因此，使用这两种浏览器时将会丢失过渡动画效果。而且，Bootstrap 并不打算使用基于 jQuery 实现替代功能。

可选选项

在任何 `.item` 中均可以通过添加 `.carousel-caption` 从而为每帧幻灯片添加说明文字。也可以添加任何 HTML 代码，这些 HTML 代码将会被自动排列和格式化。

```
<div class="item active">
          <img alt="First slide" src="http://placehold.it/900x500/78EB09/FFEB09/&text=First slide" ></img>
          <div class="carousel-caption">
              <h4>First Thumbnail label</h4>
              <p>Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at ege
           </div>
        </div>
```

为三个项，分别加上，然后效果就有了额。



可访问性问题

轮播组件并不兼容可访问性标准。如果需要兼容，请考虑其他展示幻灯片的方案。

用法

通过 data 属性

通过 data 属性可以很容易的控制轮播的定位。data-slide 可以接受控制播放位置的 prev 或 next 关键字。另外，还可以通过 data-slide-to 传递以 0 开始的幻灯片下标。

data-ride="carousel" 属性用来标记在页面加载之后即开始启动的轮播组件。

```
    <!-- Controls -->
    <a class="left carousel-control" href="#carousel-example-generic" data-slide="prev">
      <span class="glyphicon glyphicon-chevron-left"></span>
    </a>
    <a class="right carousel-control" href="#carousel-example-generic" data-slide="next">
      <span class="glyphicon glyphicon-chevron-right"></span>
    </a>
  </div>
```

在最外层的轮播容器中添加即可

```
<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
```

通过 JavaScript

手动启动轮播组件（上面我们通过使用 data-ride 属性进行自动开启轮播组件。）：

```
$('.carousel').carousel()
```

选项

可以将选项通过 data 属性或 JavaScript 传递。对于 data 属性，需要将选项名称放到 data- 之后，例如 data-interval=""。

名称	类型	默认值	描述
interval	number	5000	幻灯片轮换的等待时间。如果为false，轮播将不会自动开始循环。
pause	string	"hover"	鼠标停留在幻灯片区域即暂停轮播，鼠标离开即启动轮播。
wrap	boolean	true	轮播是否持续循环。

方法

```
$("").carousel(options)
```

初始化轮播组件，接受一个可选的 object 类型的 options 参数，并开始幻灯片循环。

```
$('.carousel').carousel({
  interval: 2000
})
```

.carousel('cycle') 从左到右循环各帧。

.carousel('pause') 停止轮播。

.carousel(number) 将轮播定位到指定的帧上（帧下标以0开始，类似数组）。

.carousel('prev') 返回到上一帧。

.carousel('next') 转到下一帧。

事件

Bootstrap 的轮播组件暴露了两个事件用于监听。

事件类型	描述
slide.bs.carousel	此事件在 `slide` 方法被调用之后立即出发。
slid.bs.carousel	当所有幻灯片播放完之后被触发。

```
$('#carousel-example-generic').on('slide.bs.carousel', function () {
    alert(1);
})
```

就这样为轮播组件绑定事件，然后在相应的运行时就会执行的。这个在之前的 JavaScript 插件中讲解的也比较多，形式都是通用的，所以只要会用就可以了。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台

更多信息请访问

http://wiki.jikexueyuan.com/project/bootstrap/