关于 Boostrap

Bootstrap 是当前世界最受欢迎的响应式、移动设备优先的门户和应用前端框架。在其中，你将发现高质量的 HTML、CSS 以及 JavaScript，使您的 WEB 工程项目变得无比简单，包括官方的 CDN 和启动器服务。

下载：
(1) Bootstrap: https://getbootstrap.com/
(2) jQuery: http://jquery.com/
(3.3.1: https://github.com/jquery/jquery/tree/3.3.1)
(3) Popper.js: https://github.com/FezVrasta/popper.js/releases
(下拉菜单 dropdowns、提示组件 popovers、冒泡组件等都提依赖于 Popper.js)

IE 浏览器支持：
支持 **Internet Explorer 10** 及更高版本，不支持 IE9（即使大多兼容，我们依然不推荐）。请注意，IE10 中不完全支持某些 CSS3 属性和 HTML5 元素，或者需要前缀属性才能实现完整的功能。
如果您需要 IE8-9 支持，请使用 Bootstrap 3 ，它是我们代码中最稳定的版本，官方不再发布新版，但仍然支持严重错误修复和文档维护。
(更多兼容性参考：https://getbootstrap.com/docs/4.3/getting-started/browsers-devices/)

重要提示：
1. 响应式 meta 标签
移动设备优先，Bootstrap 4 不同于历史版本，它首先为移动设备优化代码，然后用 CSS 媒体查询来扩展组件。为了确保所有的设备的渲染和触摸效果，必须在网页的<head>区添加响应式的视图标签，简要的说就是优先引入下面一行。

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

2. HTML5 doctype 头部规范
HTML5 标准的 doctype 头部定义是首要的，否则会导致样式失真（对搜索引擎和浏览器友好）。

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

布局

Container 容器

Container 容器是窗口布局的最基本元素，我们推荐所有样式都定义在.container 或.container-fluid 容器之中-- **这是启用整个栅格系统必不可少的前置条件**，它们分别对应选择一个响应式的、固定宽度的容器，或者选择一个流式自适应浏览器或容器最大合法宽度的窗口（意味着任何时候它的宽度总是 100%）。

(1) .container 容器可以被嵌套，但是大多数布局并不需要这么做（最少层次的嵌套构建出的网页更优雅）：

```
<div class="container">
    <!-- Content here -->
</div>
```

(2) .container-fluid 类，可以使 div 宽度扩展到整个宽度（如果没有被其它 CSS 容器包含，则应是浏览器运行时的宽度，否则应是父容器中允许的最大宽度，一般视为 100%宽度）：

```
<div class="container-fluid">
    <!-- Content here -->
</div>
```

栅格系统

Bootstrap 是基于移动优先的原则开发的，使用了一系列的媒体查询（media queries）方法，为我们的布局和界面创建自适应的的分界点。这些分界点主要是基于视口宽度的最小值，并且当窗口视图改变的时候允许元素缩放。
(分界点大小：576px、768px、992px、1200px)
Bootstrap 包含了一个强大的移动优先的网格系统，它是基于一个 12 列的布局、有 5 种响应尺寸(对应不同的屏幕)。Bootstrap4 是完全基于 flexbox 流式布局构建的，完全支持响应式标准。

	超小屏幕 (新增规格)<576px	小屏幕 次小屏≥576px	中等屏幕 窄屏≥768px	大屏幕 桌面显示器≥992px	超大屏幕 大桌面显示器≥1200px
.container 最大宽度	None (auto)	540px	720px	960px	1140px
类前缀	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
列（column）数	12				
列间隙	30px (每列两侧各15px)				
可嵌套性	Yes				
可排序性	Yes				

```
<div class="container">
    <div class="row">
        <div class="col-xl-3" style="background:#007bff;">
            111
        </div>
        <div class="col-xl-3" style="background:#868e96;">
            222
        </div>
        <div class="col-xl-3" style="background:#28a745;">
            333
        </div>
        <div class="col-xl-3" style="background:#dc3545;">
            444
        </div>
    </div>
</div>
```

自动布局列

等宽布局

所有设备上都是等宽并占满一行，只要简单的应用.col 就可以完成。

```
<div class="container">
    <div class="row">
        <div class="col-sm">
            1-1
        </div>
        <div class="col-sm">
            1-2
```

```
        </div>
    </div>
    <div class="row">
        <!-- <div class="w-100"></div> -->
        <div class="col">
            2-1
        </div>
        <div class="col">
            2-2
        </div>
        <div class="col">
            2-3
        </div>
    </div>
</div>
```

设置一列宽度

```
<div class="container">
    <div class="row">
        <div class="col">
            1-1
        </div>
        <div class="col-6">
            1-2(中间占 6 格,其余 6 格两边等分)
        </div>
        <div class="col">
            1-3
        </div>
    </div>
    <div class="row">
        <div class="col">
            2-1
        </div>
        <div class="col-5">
            2-2(中间占 5 格,其余 7 格两边等分,奇偶都能)
        </div>
        <div class="col">
            2-3
        </div>
    </div>
```

```
</div>
```

可变宽度的弹性空间

用 .col-{breakpoint}-auto 断点方法，可以实现根据其内容的自然宽度来对列进行大小调整。

```
<div class="container">
    <div class="row">
        <div class="col-1">
            1-1
        </div>
        <div class="col-auto">
            1-2
        </div>
        <div class="col-1">
            1-3
        </div>
    </div>
</div>
```

等宽多行

创建跨多个行的等宽列，方法是插入.w-100 要将列拆分为新行。通过混合.w-100 一些还可以影响一些显示状态效果,如按钮排序等。

```
<div class="row">
    <div class="col-md-6">
        1-1
    </div>
    <div class="col-md-6">
        1-2
    </div>
    <!-- </div>
    <div class="row"> -->
    <!-- <div class="w-100"></div> -->
    <div class="col-md-6">
        2-1
    </div>
    <div class="col-md-6">
        2-2
```

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

响应式的 class 选择器

混合布局

```
<div class="container">
    <div class="row">
        <div class="col-sm-6 col-lg-3">
            111
        </div>
        <div class="col-sm-6 col-lg-3">
            222
        </div>
        <div class="col-sm-6 col-lg-3">
            333
        </div>
        <div class="col-sm-6 col-lg-3">
            444
        </div>
        <div class="col-sm-6 col-lg-3">
            555
        </div>
        <div class="col-sm-6 col-lg-3">
            666
        </div>
        <div class="col-sm-6 col-lg-3">
            777
        </div>
        <div class="col-sm-6 col-lg-3">
            888
        </div>
    </div>
</div>
```

对齐

垂直对齐

1. 在 row 上加 .align-items-start/center/end

```
<div class="container">
    <div class="row align-items-start" style="height:100px;border:1px solid #F00;">
        <div class="col" style="background:#007bff;">
            111
        </div>
        <div class="col" style="background:#868e96;">
            222
        </div>
        <div class="col" style="background:#28a745;">
            333
        </div>
    </div>
</div>
```

2. 在 col 上加 .align-self-start/center/end

```
<div class="container">
    <div class="row" style="height:100px;border:1px solid #F00;">
        <div class="col align-self-start" style="background:#007bff;">
            111
        </div>
        <div class="col align-self-center" style="background:#868e96;">
            222
        </div>
        <div class="col align-self-start" style="background:#28a745;">
            333
        </div>
    </div>
</div>
```

水平对齐

在 row 上加 .justify-content-start/center/end/around/between

```
<div class="container">
    <div class="row justify-content-start" style="border:1px solid #F00;">
```

```
            <div class="col-4" style="background:#007bff;">
                111
            </div>
            <div class="col-4" style="background:#28a745;">
                222
            </div>
        </div>
</div>
```

间隙沟槽清除

在 row 上加 .no-gutters

```
<div class="container">
        <div class="row no-gutters" style="border:1px solid #F00;">
            <div class="col-4" style="background:#007bff;">
                111
            </div>
            <div class="col-4" style="background:#28a745;">
                222
            </div>
        </div>
</div>
```

重排序

Class 顺序重定义

使用 .order-* class 选择符，可以对 DIV 空间进行 **可视化排序**,系统提供了.order-1 到.order-12 12 个级别的顺序，在五种浏览器和设备宽度上都能生效。
还可以使用.order-first，快速更改一个顺序到最前面，同时其它元素也相应的获得了 **order:-1** 的属性，这个属性也可以与。.order-*混合使用。

```
<div class="container">
        <div class="row">
            <div class="col">
                111
            </div>
            <div class="col order-12">
                222
```

```
            </div>
            <div class="col order-1">
            <!-- <div class="col order-first"> -->
                333
            </div>
        </div>
</div>
```

列偏移

可以使用两种方式进行列偏应：
1、使用响应式的.offset-*栅格偏移方法。
2、使用边界处理实用程序，它内置了诸如.ml-*、.p-*、.pt-*等实用排工具。

1. class 偏移选择器

使用.offset-md-*类可以使列向右偏移，通过定义*的数字，则可以实现列偏移，如.offset-md-4 则是向右偏移四列。

```
<div class="container">
    <div class="row">
        <div class="col-4" style="background:#007bff;">
            111
        </div>
        <div class="col-4 offset-4" style="background:#28a745;">
            222
        </div>
    </div>
</div>
```

2. Margin 移动布局

在 BootStrap V4 中，你可以使用.ml-auto 与.mr-auto 来强制隔离两边的距离，实现类水平隔离的效果。

```
<div class="container">
    <div class="row">
        <div class="col-4" style="background:#007bff;">
            111
        </div>
        <div class="col-4 ml-md-auto" style="background:#28a745;">
            222
        </div>
    </div>
</div>
```

列嵌套

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）数量推荐不要超过 12 个（其实，没有要求你必须占满 12 列-否则应对页面进行重新规划布局）。

```
<div class="container">
    <div class="row">
        <div class="col-md-4">
            111
        </div>
        <div class="col-md-8">
            <div class="row">
                <div class="col-md-4">
                    222
                </div>
                <div class="col-md-8">
                    333
                </div>
            </div>
        </div>
    </div>
</div>
```

禁用响应式

(1) 设定容器宽度。如.container {width: 980px;}。
(2) 栅格布局使用 .col-*（最小设备栅格类）的样式来代替.col-sm-*、.col-md-*、.col-lg-*、.col-xl-*，这样就能确保栅格能够在所有设备中展开。
(3) 移除 此 CSS 文档中提到的设置浏览器视口（viewport）的标签：<meta>。
(4) 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。

响应式的分界点

Bootstrap 是基于移动优先的原则开发的，使用了一系列的媒体查询（media queries ）方法，

为我们的布局和界面创建自适应的的分界点。这些分界点主要是基于视口宽度的最小值，并且当窗口视图改变的时候允许元素缩放。

```
// Extra small devices (portrait phones, less than 576px)
    // No media query since this is the default in Bootstrap
// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }

// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

// Extra small devices (portrait phones, less than 576px)
// No media query since this is the default in Bootstrap
// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }
// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }
// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }
// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }

```
<style>
    .topTt{font-size:12px;}

    @media(min-width: 576px){
        .topTt{font-size:20px;}
    }

    @media(min-width: 992px){
        .topTt{font-size:30px;}
    }
</style>
```

偶尔也会使用其它方面的媒体查询（指定屏幕的尺寸或更小）：

```
// Extra small devices (portrait phones, less than 576px)
@media (max-width: 575.98px) { ... }

// Small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }

// Medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }

// Large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }

// Extra large devices (large desktops)
// No media query since the extra-large breakpoint has no upper bound on its width
```

// Extra small devices (portrait phones, less than 576px)
@media (max-width: 575.98px) { ... }
// Small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }
// Medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }
// Large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }
// Extra large devices (large desktops)
// No media query since the extra-large breakpoint has no upper bound on its width

以及使用最小和最大断点宽度定位单个屏幕尺寸段的媒体查询和混合定义：

```
// Extra small devices (portrait phones, less than 576px)
@media (max-width: 575px) { ... }

// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) and (max-width: 767px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) and (max-width: 991px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) and (max-width: 1199px) { ... }

// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

// Extra small devices (portrait phones, less than 576px)
@media (max-width: 575px) { ... }
// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) and (max-width: 767px) { ... }
// Medium devices (tablets, 768px and up)

```
@media (min-width: 768px) and (max-width: 991px) { ... }
// Large devices (desktops, 992px and up)
@media (min-width: 992px) and (max-width: 1199px) { ... }
// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

内容

排版

标题

兼容所有 HTML 标题集，涵括从 <h1> 到 <h6>,的六种标题展现。
CSS 选择器也支持以.h1 -- .h6 方式引用，这样可以使字体样式呈现出标题效果，不同是引用.h1 -- .h6 的文本段不会视作 HTML 的标题元素（往往 SEO、读屏器和机器识别时对此很敏感）。

```
<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>

<p class="h1">h1. Bootstrap heading</p>
<p class="h2">h2. Bootstrap heading</p>
<p class="h3">h3. Bootstrap heading</p>
<p class="h4">h4. Bootstrap heading</p>
<p class="h5">h5. Bootstrap heading</p>
<p class="h6">h6. Bootstrap heading</p>
```

自定义标题备注

使用附带的实用类从 Bootstrap 重新创建小的辅助标题文本。

```
<h3>
    主标题主标题
```

```
    <small class="text-muted">副标题副标题</small>
</h3>
```

显式标题

bootstrap 可以传统的标题元素设计得更漂亮，以迎合你的网页内容。如果你想要一个标题醒目，考虑使用显示标题——一种更大型、鲜明的标题样式。

```
<h1 class="display-1">Display 1</h1>
<h1 class="display-2">Display 2</h1>
<h1 class="display-3">Display 3</h1>
<h1 class="display-4">Display 4</h1>
```

Lead 中心内容

通过应用 .lead 样式，可以定义一个中心段落，用于提示这是中心内容或重要内容。

```
<p>苹果苹果苹果苹果</p>
<p class="lead">香蕉香蕉香蕉香蕉</p>
<p>橘子橘子橘子橘子</p>
<p>梨子梨子梨子梨子</p>
```

文本内联元素

del 能更形象的描述意思。ins 代表被插入的文字，u 代表有下划线。
strong、em 等有强调作用，有利于 seo(搜索引擎优化)。

```
<p>看看我是不是<mark>高亮</mark>文本</p>
<p>看看我是不是<span class="mark">高亮</span>文本</p>

<p><small>小号字小号字小号字</small></p>
<p><span class="small">小号字小号字小号字</span></p>

<p><del>删除线删除线删除线</del></p>
<p><s>删除线删除线删除线</s></p>

<p><ins>下划线下划线下划线</ins></p>
<p><u>下划线下划线下划线</u></p>
```

```
<p><strong>粗体粗体粗体</strong></p>
<p><b>粗体粗体粗体</b></p>

<p><em>斜体斜体斜体</em></p>
<p><i>斜体斜体斜体</i></p>
```

abbr 缩略语

```
<p><abbr title="请填写您的邮箱" class="initialism">email</abbr></p>
```

blockquote 来源备注与引用

引用文档中另一个来源的内容块，请在一段 HTML 外面包裹 <blockquote class="blockquote">，作为引用。为了显示直接引用，我们推荐使用<p>作为子级包裹容器。
底部来源：<footer class="blockquote-footer"> 用于标识来源，一般用于页脚（所以有*-footer），然后配合 <cite>使用。
(<cite> 标签通常表示它所包含的文本对某个参考文献的*引用*，比如书籍或者杂志的标题。)
对齐处理：如果需要居中对齐或右对齐，使用.text-center、.text-right 方法配合即可。

```
<blockquote class="blockquote text-right">
    <p class="mb-0">爱上一个地方，就应该背上包去旅游，走得更远。</p>
    <footer class="blockquote-footer">出自商务印书馆的 <cite title="Source Title">《新华字典》</cite></footer>
</blockquote>
```

列表

列表样式初始化

在 ul(或 ol）上使用.list-unstyled 可以删除列表项目上默认的 list-style 以及左外边距（只针对直接子元素），这只生效于在直接子列表项目上，不影响你嵌套的子列表。

```
<ul class="list-unstyled">
    <li>列表 111</li>
    <li>列表 222</li>
    <li>列表 333</li>
```

```
    <li>
        <ul>
            <li>aaa</li>
            <li>bbb</li>
            <li>ccc</li>
        </ul>
    </li>
</ul>
```

分行或单行多列并排

使用.list-inline 、 .list-inline-item 结合，可以实现列表逐行显示（默认不引用且无父元素影响也是逐行显示）、或单行并多列并排（遵循从左对右的原则、并清除 margin 方法）。

```
<!-- <ul class="list-inline"> -->
<ul>
    <li class="list-inline">列表 111</li>
    <li>列表 222</li>
    <li>列表 333</li>
</ul>
```

```
<ul class="list-inline">
    <li class="list-inline-item">首页</li>
    <li class="list-inline-item">最新资讯</li>
    <li class="list-inline-item">公司简介</li>
</ul>
```

dl 表格式水平描述

使用 BootStrap 栅格系统的预定义类（或者说语义化 mixins），可以水平对齐条目和描述。对于较长的条目，你可以视情况添加一个.text-truncate 类，从而用省略号截断文本。

```
<dl class="row">
    <dt class="col-sm-2">姓名</dt>
    <dd class="col-sm-10">张三</dd>

    <dt class="col-sm-2 text-truncate">是否有过带团队的经验以及团队遇到过的问题和解决办法</dt>
    <dd class="col-sm-10">有过 6 年带领团队开发经验</dd>
```

```
    <dt class="col-sm-2">其他</dt>
    <dd class="col-sm-10">
        <dl class="row">
            <dt class="col-md-2">性别</dt>
            <dd class="col-md-10">男</dd>
        </dl>
    </dd>
</dl>
```

代码

内联代码

用<code>包裹内联代码片断，勿忘转义 HTML 尖括号。

```
示例：  <code>&lt;section&gt;</code>  代码嵌入到文本段中。
```

代码块

```
<pre><code>&lt;p&gt;Sample text here...&lt;/p&gt;
&lt;p&gt;And another line of sample text here...&lt;/p&gt;
</code></pre>
```

Var 变量

推荐使用 <var>标签包裹标示变量。

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

用户输入(键盘动作提示)

使用 <kbd> 标签，标明这是一个键盘输入操作。

```
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>
```

示例标注

<samp> 标签代表这是一个示例。

```
这是一个代码示例. <br>
<samp>这是一个代码示例.</samp>
```

图片

响应式图片&缩略图处理

在 Bootstrap 中，给图片添加.img-fluid 样式，或定义 max-width: 100%、height:auto;样式，即可赋得响应式特性，图片大小会随着父元素大小同步缩放。
您可以使用.img-thumbnail 属性来使图片自动被加上一个带圆角且 1px 边界的外框缩略图样式（你也可以使用系统提供的边隙间距方法，如.p-1 再加上边框颜色定义达成）。

```
<div class="row">
    <div class="col-lg-4">
        <img src="images/lg.jpg" alt="" class="img-fluid">
    </div>
    <div class="col-lg-4">
        <img src="images/sm.jpg" alt="" class="w-100 img-fluid">
    </div>
    <div class="col-lg-4">
        <img src="images/lg.jpg" alt="" class="img-fluid img-thumbnail">
    </div>
</div>
```

图像对齐处理

对于.block 属性的块状图片，我们也可以使用 浮动定义规范 或 文字对齐规范，来实现对图像的对齐、浮动控制，带.block 块属性的图片，可以自动获得 .mx-auto 的位置对齐属性。

```
<div class="clearfix" style="border: 1px solid #F00;">
    <img src="images/sm.jpg" alt="" class="rounded float-left">
```

```
    <img src="images/sm2.jpg" alt="" class="rounded float-right">
</div>
```

```
<div style="border: 1px solid #F00;">
    <img src="images/sm.jpg" alt="" class="rounded d-block mx-auto">
</div>
<!-- <div class="text-center" style="border: 1px solid #F00;">
    <img src="images/sm.jpg" alt="" class="rounded">
</div> -->
```

Html 5 标准之 Picture 元素

HTML5 标准提供了一个全新的<picture> 元素，它可以为 <img>指定多个<source> 定义，请确保在<img> 标签里使用使用.img-* CSS 样式进行定义绑定，而不是仅仅认为引用了<img> 就达成了。

```
<picture>
    <source srcset="images/lg.jpg"    media="(min-width: 992px)" >
    <source srcset="images/md.jpg"    media="(min-width: 576px)">
    <img src="images/sm.jpg" alt="">
</picture>
```

图文框

如果你需要显示的内容区包括了一个图片和一个可选的标题，可使用.figure 样式定义。Bootstrap 的.figure 以及.figure-caption 类，为 HTML5 的<figure>以及<figcaption>元素提供了一个基准样式处理。默念认的图片系统不会定义明确的大小，因此请务必将该.img-fluid 类添加到您的<img>标签中才能实现与响应式的完美结合。

文字对齐控制：结合我们的文本实用工具可以轻松对齐图文框的说明文字（比如右对齐、左对齐，只要引用.text-*方法即可）。

```
<div class="container">
    <div class="row">
        <div class="col-lg-4">
            <figure class="figure">
                <img src="images/lg.jpg" class="figure-img img-fluid rounded" alt="">
                <figcaption class="figure-caption">A caption for the above image.</figcaption>
            </figure>
        </div>
```

```
            <div class="col-lg-4">
                <figure class="figure">
                    <img src="images/md.jpg" class="figure-img img-fluid rounded" alt="">
                    <figcaption class="figure-caption text-center">A caption for the above image.</figcaption>
                </figure>
            </div>
            <div class="col-lg-4">
                <figure class="figure d-block">
                    <img src="images/sm.jpg" class="figure-img w-100 img-fluid rounded" alt="">
                    <figcaption class="figure-caption text-right">A caption for the above image.</figcaption>
                </figure>
            </div>
        </div>
</div>
```

表格

(1) 在第三方部件例如日历和日期选择器中广泛使用表格，我们设计了视情况需要加入的表格类。只需要向某个<table>添加一个基类.table，然后通过自定义样式或系统提供的 class 来起作用。

使用最基本的表格标记，下面是 Bootstrap 中 .table 表格的样式（基本样式），**Bootstrap 4 继承了所有的表格样式**，这意味着任何嵌套表格都将以与父类型相同的方式进行样式化。

(2) 你可使用.table-dark class 选择器来产生颜色反转对比效果，即深色背景和浅色文本。

Head 表头处理：与预设的反转样式相似，使用.thead-light 或.thead-dark 中的一个样式，就能使 <thead>区显示出浅黑或深灰。

(3) **条纹状表格**：使用 .table-striped 样式定义 <tbody>，可以产生逐行颜色强烈对比的表格样式（以及增加反转）。(可同.table-dark 结合使用)

(4) **表边框处理**：添加 .table-bordered 类可以产生表格边框与间隙系统。(同)

无边框：添加.table-borderless 无边界表格。(同)

(6) **行悬停效果**：将 .table-hover 定义上，可以产生行悬停效果（鼠标移到行上会出现状态提示）。(同)

(7) **紧缩表格**：添加 .table-sm 可以将表格的 padding 值缩减一半，使表格更加紧凑。(同)

(8) **Captions 表格辅助标题**：<caption> 标签如同一个表格的标题，它默认是隐藏的，可以协助屏幕阅读器用户找到表格、了解表格内容，且决定是否需要阅读它。

```
<table class="table">
    <!-- <caption>List of users</caption> -->
    <thead>
```

```
        <tr>
            <th>#</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Username</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>1</th>
            <td>Mark</td>
            <td>Otto</td>
            <td>@mdo</td>
        </tr>
        <tr>
            <th>2</th>
            <td>Jacob</td>
            <td>Thornton</td>
            <td>@fat</td>
        </tr>
        <tr>
            <th>3</th>
            <td>Larry</td>
            <td>the Bird</td>
            <td>@twitter</td>
        </tr>
    </tbody>
</table>
```

语义状态化

使用语义状态样式对表格逐行或单个单元格进行着色表达。(On rows or On cells (`td` or `th`))

```
<table class="table table-bordered table-hover">
    <thead>
        <tr>
            <th>Type</th>
            <th>Column heading</th>
            <th>Column heading</th>
            <th>Column heading</th>
        </tr>
    </thead>
```

```
    <tbody>
        <tr class="table-active">
            <th>Active</th>
            <td>Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr>
        <tr>
            <th>Default</th>
            <td>Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr>
        <tr class="table-primary">
            <th>Primary</th>
            <td>Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr>
        <!-- <tr>
            <th class="table-primary">Primary</th>
            <td class="table-primary">Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr> -->
        <tr class="table-secondary">
            <th>Secondary</th>
            <td>Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr>
        <tr class="table-success">
            <th>Success</th>
            <td>Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr>
        <tr class="table-danger">
            <th>Danger</th>
            <td>Column content</td>
            <td>Column content</td>
            <td>Column content</td>
        </tr>
        <tr class="table-warning">
```

```
                <th>Warning</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
            <tr class="table-info">
                <th>Info</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
            <tr class="table-light">
                <th>Light</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
            <tr class="table-dark">
                <th>Dark</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
        </tbody>
</table>
```

深色表格上没有固定的背景，你可以使用 文字或背景通用样式 获得类似的样式：

```
<table class="table table-bordered table-hover">
        <thead>
            <tr>
                <th>#</th>
                <th>Column heading</th>
                <th>Column heading</th>
                <th>Column heading</th>
            </tr>
        </thead>
        <tbody>
            <tr class="bg-primary">
                <th>primary</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
```

```
            <!-- <tr>
                <th class="bg-primary">primary</th>
                <td class="bg-primary">Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr> -->
            <tr class="bg-success">
                <th>success</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
            <tr class="bg-warning">
                <th>warning</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
            <tr class="bg-danger">
                <th>danger</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
            <tr class="bg-info">
                <th>info</th>
                <td>Column content</td>
                <td>Column content</td>
                <td>Column content</td>
            </tr>
        </tbody>
    </table>
```

响应式表格

当表格想要始终呈现水平滚动，可在.table 上加入.table-responsive 获得响应式表现，从而支持任何 viewport 窗口。也可以在.table 上，加 .table-responsive{-sm|-md|-lg|-xl}属性来定义多屏幕尺寸响应支持。

```
<div class="table-responsive">
    <table class="table">
        <thead>
```

```
            <tr>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
                <th>HeadingHeadingHeading</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
                <td>CellCellCell</td>
            </tr>
        </tbody>
    </table>
</div>
```

公共样式

边框

使用边框通用定义类来快速设置元素的边框和边框半径，适用于图像、按钮或任何其他元素。

边框

```
<style>
```

```
        span{display:inline-block;width:75px;height:75px;margin:5px;border:1px
solid;background:#F5F5F5;}
</style>

<span></span>
<hr>
<!-- 添加边框属性，显示指定边框。 -->
<span class="border"></span>
<span class="border-top"></span>
<span class="border-right"></span>
<span class="border-bottom"></span>
<span class="border-left"></span>
<hr>
<!-- 在一个空间上定义边框-删除或显示特定边框定义方法。 -->
<span class="border-0"></span>
<span class="border-top-0"></span>
<span class="border-right-0"></span>
<span class="border-bottom-0"></span>
<span class="border-left-0"></span>
<hr>
<!-- 使用我们的主题颜色类方法，定义边框颜色。 -->
<span class="border border-secondary"></span>
<span class="border border-success"></span>
<span class="border border-danger"></span>
<span class="border border-warning"></span>
<span class="border border-info"></span>
<span class="border border-light"></span>
<span class="border border-dark"></span>
<span class="border border-white"></span>
<hr>
```

圆角边框

使用.rounded 元素可以轻松的定义四个圆角的孤度及显示效果。

```
<style>
        img{width:75px;height:75px;margin:5px;}
</style>

<img src="images/sm.jpg" alt="">
<img src="images/sm.jpg" alt="" class="rounded">
<img src="images/sm.jpg" alt="" class="rounded-top">
<img src="images/sm.jpg" alt="" class="rounded-right">
```

```
<img src="images/sm.jpg" alt="" class="rounded-bottom">
<img src="images/sm.jpg" alt="" class="rounded-left">
<img src="images/sm.jpg" alt="" class="rounded-circle">
<img src="images/sm.jpg" alt="" class="rounded-pill" style="width:140px;">
<img src="images/sm.jpg" alt="" class="rounded-0" style="border-radius:5px;">
```

浮动属性&清动浮动

使用 class 样式来切换 float 效果：.float{-sm/md/lg/xl}-left/right/none。
float 类样式是通过添加 .clearfix **到父元素上**来达达到清除目标。

```
<div class="border border-danger clearfix">
    <div class="float-left">left</div>
    <div class="float-right">right</div>
</div>
```

颜色

颜色

```
<p class="text-primary">.text-primary</p>
<p class="text-secondary">.text-secondary</p>
<p class="text-success">.text-success</p>
<p class="text-danger">.text-danger</p>
<p class="text-warning">.text-warning</p>
<p class="text-info">.text-info</p>
<p class="text-light bg-dark">.text-light</p>
<p class="text-dark">.text-dark</p>
<p class="text-muted">.text-muted</p>
<p class="text-white bg-dark">.text-white</p>
```

.text-primary

.text-secondary

.text-success

.text-danger

.text-warning

.text-info

.text-light

.text-dark

.text-muted

.text-white

使用我们提供的悬停和焦点状态(情景)样式，在链接上也能正常使用(呈现)， 注意: e .text-white 、 .text-muted 这两个 class 样式不支持链接上使用 (没有链接样式)。

```
<p><a href="#" class="text-primary">Primary link</a></p>
<p><a href="#" class="text-secondary">Secondary link</a></p>
<p><a href="#" class="text-success">Success link</a></p>
<p><a href="#" class="text-danger">Danger link</a></p>
<p><a href="#" class="text-warning">Warning link</a></p>
<p><a href="#" class="text-info">Info link</a></p>
<p><a href="#" class="text-light bg-dark">Light link</a></p>
<p><a href="#" class="text-dark">Dark link</a></p>
<p><a href="#" class="text-muted">Muted link</a></p>
<p><a href="#" class="text-white bg-dark">White link</a></p>
```

Primary link

Secondary link

Success link

Danger link

Warning link

Info link

Light link

Dark link

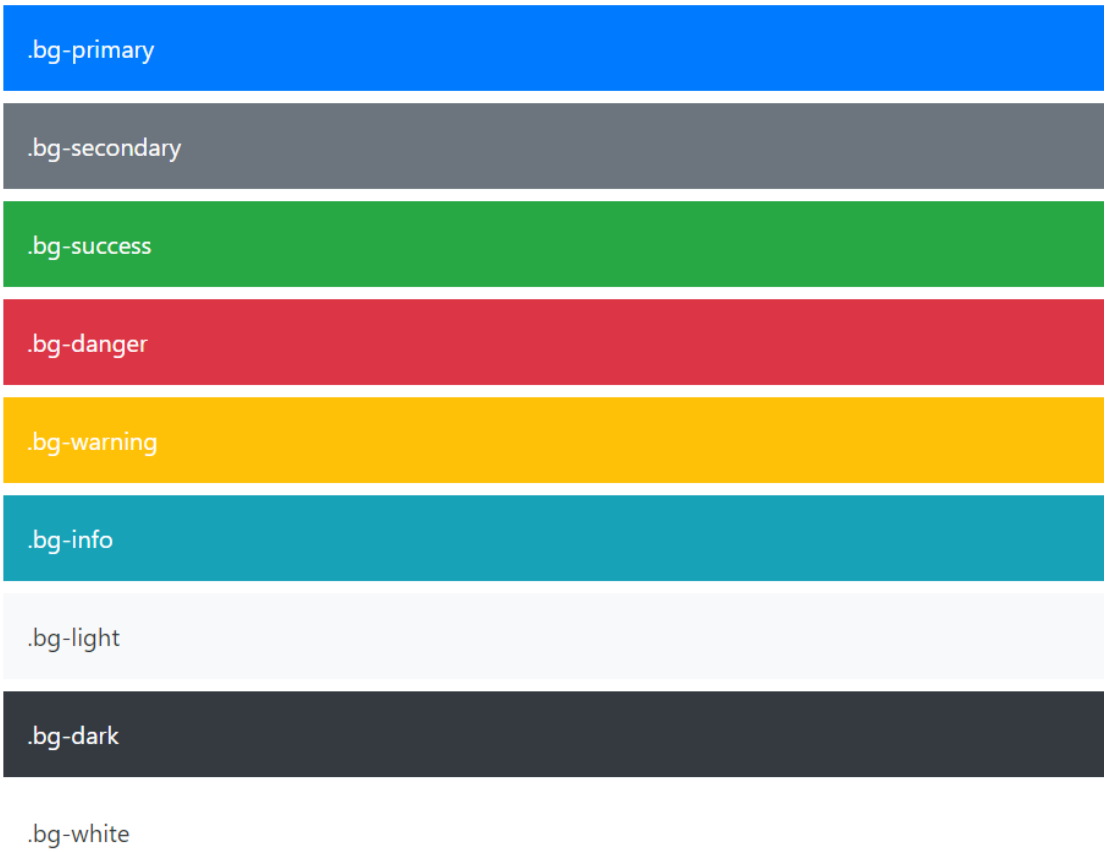
Muted link

White link

背景颜色

```
<style>
    p{padding:1rem;margin-bottom:.5rem;}
</style>

<p class="bg-primary text-white">.bg-primary</p>
<p class="bg-secondary text-white">.bg-secondary</p>
<p class="bg-success text-white">.bg-success</p>
<p class="bg-danger text-white">.bg-danger</p>
<p class="bg-warning text-white">.bg-warning</p>
<p class="bg-info text-white">.bg-info</p>
<p class="bg-light text-dark">.bg-light</p>
<p class="bg-dark text-white">.bg-dark</p>
<p class="bg-white text-dark">.bg-white</p>
```



Display 显示属性

Display 属性

display 类格式：.d{-sm/md/lg/xl}-{value}
display 常用属性(value)：none、inline、inline-block、block、table、table-cell、table-row、flex、inline-flex。

```
<div class="d-inline">aaa</div>
<div class="d-inline">bbb</div>
<hr>
<span class="d-block">111</span>
<span class="d-block">222</span>
```

隐藏元素

为了更快速且友好 的支持移动设备开发，请使用 display classes 来显示和隐藏组件，避免

创建完全不同版本的一个网站（为移动网站建立一个独立的站点），而不是按照每种屏幕尺寸来隐藏元素。
隐藏元素只要使用 .d-none class 或 .d-{sm,md,lg,xl}-none 的任何变量来支持响应式。
如要在指定的屏幕上显示一个元素，则可以将一个.d-*-none class 样式与 .d-*-* class 样式结合起来，如 .d-none.d-md-block.d-xl-none 将隐藏除了中型、大型设备以外的所有屏幕中的元素。

屏幕规格	引用样式
所有屏幕下隐藏	.d-none
只在xs屏幕上隐藏（即仅在手机屏幕上隐藏-其它规格屏幕正常显示）	.d-none .d-sm-block
只在sm屏幕上隐藏（其它屏幕规格均显示）	.d-sm-none .d-md-block
只在md屏幕时隐藏（其它屏幕规格均显示）	.d-md-none .d-lg-block
只在lg屏幕时隐藏（其它屏幕规格均显示）	.d-lg-none .d-xl-block
只在xl屏幕时隐藏（其它屏幕规格均显示）	.d-xl-none
全部可见	.d-block
仅在xs屏幕时可见	.d-block .d-sm-none
仅在sm屏幕时可见	.d-none .d-sm-block .d-md-none
仅在md屏幕时可见	.d-none .d-md-block .d-lg-none
仅在lg屏幕时可见	.d-none .d-lg-block .d-xl-none
仅在xl屏幕时可见	.d-none .d-xl-block

```
<img src="images/sm.jpg" alt="" class="d-none d-sm-inline-block">
```

面向打印的显示属性控制处理

在处理打印样式时，通过 .d-print-{value} 样式来改变相应值处理呈现效果。

```
<p>aaaaaa</p>
<p class="d-print-none">bbbbbb</p>
<p class="d-none d-print-block">cccccc</p>
<p>dddddd</p>
```

文本处理

文本对齐

使用文本对齐.text{-sm/md/lg/xl}-left/center/right/justify 样式类轻松地将文本重新对齐到组件。

```
<p>left</p>
<p class="text-md-center">center</p>
<p class="text-right">right</p>
<p class="text-justify">Ambitioni dedisse scripsisse iudicaretur. Cras mattis iudicium purus sit amet fermentum. Donec sed odio operae, eu vulputate felis rhoncus. Praeterea iter est quasdam res quas ex communi. At nos hinc posthac, sitientis piros Afros. Petierunt uti sibi concilium totius Galliae in diem certam indicere. Cras mattis iudicium purus sit amet fermentum.</p>
```

文本包裹和溢出(换行)处理

.text-nowrap class 样式类可以防止文本换行。

```
<div class="border border-danger text-nowrap" style="width:8rem;">
    This text should overflow the parent.
</div>
```

对于更长的内容，你可以添加一个 .text-truncate class 样式，以省略号截断文本（需要结合 **display: inline-block** 或 **display: block** 来使用）。

```
<div class="border border-danger text-truncate" style="width:8rem;">
    This text should overflow the parent.
</div>
<!-- <span class="border border-danger d-inline-block text-truncate" style="width:8rem;">
    This text should overflow the parent.
</span> -->
```

字母大小写转换

使用文本大小写样式将文字内容由小写，转为大写（不支持中文）。
text-capitalize 仅支持每一个词的第一个字母转为大写，而其它字母不受影响。

```
<p class="text-lowercase">LowerCased text.</p>
<p class="text-uppercase">UpperCased text.</p>
<p class="text-capitalize">CapiTaliZed text.</p>
```

粗细和斜体

```
<p class="font-weight-bold">Bold text.</p>
<p class="font-weight-normal">Normal weight text.</p>
<p class="font-weight-light">Light weight text.</p>
<p class="font-italic">Italic text.</p>
```

等宽字体

将选择更改为我们的等宽字体堆栈.text-monospace。

```
<p>This is in monospace. 这是等宽。</p>
<p class="text-monospace">This is in monospace. 这是等宽。</p>
```

垂直对齐

使用 vertical-alignment 通用样式改变元素的对齐，注意：垂直对齐仅影响 内联 inline、 内联块 inline-block、 内联表 inline-table、 表格单元格 table cell 元素。
可选属性有：.align-baseline、.align-top、.align-middle、.align-bottom、.align-text-bottom、.align-text-top。

```
访问<a href="http://www.web-666.com">网战天下</a>观看更多精品教程。
<button class="align-top">点击观看</button>
```

在 table cells 表格单元格：

```
<table class="table table-bordered" style="height:100px;">
    <tbody>
        <tr>
            <td>top</td>
            <td class="align-middle">middle</td>
            <td class="align-bottom">bottom</td>
        </tr>
    </tbody>
```

```
</table>
```

规格与尺寸

宽度和高度可以用.w/h-25/50/75/100 产生，包括 25%、50%、75%、 100% ，你可根据整这些值，从而生产出不同的规格属性。

```
<p class="bg-danger w-25">Width 25%</p>
<p class="bg-danger w-50">Width 50%</p>
<p class="bg-danger w-75">Width 75%</p>
<p class="bg-danger w-100">Width 100%</p>
```

```
<div class="border border-danger" style="height:100px;">
    <div class="d-inline-block bg-primary h-25">Height 25%</div>
    <div class="d-inline-block bg-primary h-50">Height 50%</div>
    <div class="d-inline-block bg-primary h-75">Height 75%</div>
    <div class="d-inline-block bg-primary h-100">Height 100%</div>
</div>
```

你也可使用.mw-100、.mh-100 产生 max-width: 100%; 和 max-height: 100%; 这些通用样式定义

```
<div class="bg-danger" style="width:200px;height:200px;">
    <div class="bg-primary mw-100 mh-100" style="width:300px;height:300px;"></div>
</div>
```

间隔

Spacing 通用样式适用于所有屏幕尺寸，从 xs 到 xl 各种规格尺寸。因为这些类是从 min-width: 0 及以上开始引用的，所以不受媒体查询的约束，然而，其余的屏幕断点（设备解析）包含屏幕尺寸缩写。
对于 xs 屏幕，使用固定格式{property}{sides}-{size} 命名 CSS 方法，对于 sm、md、lg、xl 使用{property}{sides}-{breakpoint}-{size}格式命名 CSS 方法。

如果 **属性**(property) 是下列之一:

- m - 这个 Class 属性会设定 margin 值
- p - 这个 Class 属性会设定 padding 值

边缘(sides) 设定:

- t - 这个 Class 属性会设定 margin-top 或 padding-top
- b - 这个 Class 属性会设定 margin-bottom 或 padding-bottom

- l - 这个 Class 属性会设定 margin-left 或 padding-left
- r - 这个 Class 属性会设定 margin-right 或 padding-right
- x - 这个 Class 属性会设定 *-left 和 *-right 两个值。
- y - 这个 Class 属性会设定 *-top 和 *-bottom 两个值
- 空白 - 这个 Class 属性会设定 margin 或 padding 元素的四个边。

尺寸(size) 规格定义:
0 - 这个 Class 属性会设定 margin 或 padding 的样式值为 0
1 - (默认时)这个 Class 属性会设定 margin 或 padding 以 $spacer * .25 规格呈现
2 - (默认时) 这个 Class 属性会设定 margin 或 padding 以 $spacer * .5 规格呈现
3 - (默认时)这个 Class 属性会设定 margin 或 padding 以 $spacer * 1 规格呈现
4 - (默认时) 这个 Class 属性会设定 margin 或 padding 以 $spacer * 1.5 规格呈现
5 - (默认时)这个 Class 属性会设定 margin 或 padding 以 $spacer * 3 规格呈现
auto - 这个 Class 属性会设定 margin 值 auto（按浏览器默认值自由展现）。

```
<div class="d-inline-block border border-danger p-md-5">
    111
</div>
```

水平居中: Bootstrap 也包括一个 .mx-auto class 样式，用于固定宽度的盒模型水平居中，具有 display: block 和 width 设置水平边距内容的 auto 居中。

```
<div class="d-block border border-danger w-25 mx-auto">
    111
</div>
```

阴影

可以使用.shadow-none 和.shadow{-sm/lg}实用工具类快速添加或删除阴影。

```
<div class="p-3 mb-5 bg-white rounded">网战天下</div>
<div class="p-3 mb-5 bg-white rounded shadow">网战天下</div>
<div class="p-3 mb-5 bg-white rounded shadow-sm">网战天下</div>
<div class="p-3 mb-5 bg-white rounded shadow-lg">网战天下</div>
<div class="p-3 mb-5 bg-white rounded shadow-none" style="box-shadow:10px 10px 5px #EEE;">网战天下</div>
```

position 固顶(底)及定位

通用属性

使用.position-static/relative/absolute/fixed/sticky 样式，可以实现快速定位-虽然它们不包含响应式支持。

```
<div class="position-absolute">网战天下</div>
```

固定在顶(底)部

可以用.fixed-top/bottom 将一个元素固定在可见区域的顶(底)部，从边到边的对齐，用户在使用固定在顶部时请确认效果带来的影响（如覆盖）-必要时增加额外的自定义 CSS。

```
<style>
    body{padding:50px 0;}
</style>

<div class="fixed-top">
    <div class="container bg-danger" style="height:50px;">header</div>
</div>

<div class="container bg-light">
1<br>2<br>3<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br
>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>
1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br
>1<br>1<br>1<br>1<br>1<br>1<br>11<br>12<br>13<br>
</div>

<div class="fixed-bottom">
    <div class="container bg-success" style="height:50px;">footer</div>
</div>
```

贴齐于 top 顶部

将一个元素轩于可见区域的顶部，从边到边-但只在你的浏览器窗口滚动才能激活它，该 .sticky-top 样式使用 position: sticky 不能在所有浏览器中获得支持。
Microsoft Edge 和 IE11 呈现 position: sticky **使用的是** position: relative. 属性，因此我们将这个样式增加了 @supports 动态变量，限制在可支持的浏览器上运行。

```
<div class="container">
    <h1>
        网战天下
        <small class="text-muted">www.web-666.com</small>
    </h1>

    <div class="bg-primary sticky-top" style="height:50px;">导航条</div>

    <div>

    1<br>2<br>3<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>
1<br>1<br>1<br>1<br>1<br>1<br>

    1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>
1<br>1<br>1<br>11<br>12<br>13<br>
    </div>
</div>
```

visibility 显示或隐藏处理

使用通用样式的 visibility 元属，这不会改变元素的 display 值，并且有助于大部分使用者隐藏内容，但仍然保留在屏幕阅读器中。

根据需要选用 .visible 或 .invisible 两个 Class 样式。

```
aaa
<span class="invisible">bbb</span>
ccc
```

关闭图标

使用通用的 close 关闭图标来关闭 modals 模态框提示或 alert 提示组件的内容。

```
<button type="button" class="close">&times;</button>
<!-- <a href="###" class="close">&times;</a> -->
```

嵌入(embed)

创建**响应式**的视频、图像、幻灯片，并能在在任何设备上友好的扩展显示。
将这些规则应用到 <iframe>、<embed>、 <video>、 <object> 上，当需要配合其它属性（如响应式）时，也可以加入 .embed-responsive-item 定义。
你不需要将 frameborder="0" 加入到你的 <iframe>中，因为我们已经为您覆盖处理了这个属性。
用 .embed-responsive 实现同比例收缩，至于 .embed-responsive-item 不是严格要求的-虽然我们鼓励使用它。
长宽比例处理：.embed-responsive-21by9 / 16by9 / 4by3 / 1by1。

```
<div class="embed-responsive embed-responsive-16by9">
    <iframe src="https://www.taobao.com" class="embed-responsive-item"></iframe>
</div>
```

图像替换

使用 .text-hide class 样来隐藏一个元素的文字内容并替换成背景图片。
使用.text-hide class 样式可以保持标签的亲和性及 SEO 优化需求，引入后，需要使用 background-image 属性来提供视觉展示，而不是文字内容（文字内容随即隐藏）。

```
<!-- <h1 class="text-hide">网战天下</h1> -->
<h1 class="text-hide" style="background-image:url('images/sm.jpg');
width:75px;height:75px;">网战天下</h1>
```

读屏器

支持视觉隐藏的内容、但保持可访问的辅助技术，如屏幕阅读器，可以使用.sr-only 类风格。在需要向非视觉用户传达额外的视觉信息或提示（例如通过使用颜色表示的含义）的情况下，这通常很有用。

```
<p class="text-danger">
  <span class="sr-only">Danger: </span>
  This action is not reversible
</p>
```

通过 .sr-only 可定义 **屏幕阅读器支持的元素**。 .sr-only 与 .sr-only-focusable 结合，可以防止被键盘激活后再次显示此地素（如通过键盘）。

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

flex 弹性布局

父级：
1.**启用**：.d{-sm/md/lg/xl}-flex/inline-flex
2.**方向**：.flex{-sm/md/lg/xl}- row /row-reverse / column/column-reverse
3.**内容对齐与对准**：.justify-content{-sm/md/lg/xl}-start/center/end/between/around
4.**对齐项目**：.align-items{-sm/md/lg/xl}-stretch/start/center/end/baseline
5. **Wrap 包裹**：.flex{-sm/md/lg/xl}-nowrap/wrap/wrap-reverse
6.**对齐内容**：.align-content{-sm/md/lg/xl}-stretch/start/center/end/around
子元素：
1.**自对齐**：.align-self{-sm/md/lg/xl}-stretch/start/center/end/baseline
2.**自相等**：.flex{-sm/md/lg/xl}-fill
3.**等宽变幻**：.flex{-sm/md/lg/xl}-grow-0/1
缩小能力：.flex{-sm/md/lg/xl}-shrink-1/0
4.**自浮动**：水平：.mr-auto、.ml-auto
垂直：.mb-auto、.mt-auto (可结合 align-items、flex-direction: column)
5. **Order 排序**：.order{-sm/md/lg/xl}-1 至 12（默认 0，越小越靠前）

```
<style>
    .myFlex{width:500px;height:500px;background:#00F;}
    .myFlex>div{width:100px;background:#F00;}
</style>

<div class="myFlex d-flex flex-lg-row">
    <div>1</div>
    <div>2</div>
    <div>3</div>
</div>
```

组件

警告提示框(Alert)

示例

警告提示框

```
<div class="alert alert-primary">
```

```
  This is a primary alert—check it out!
</div>
<div class="alert alert-secondary">
  This is a secondary alert—check it out!
</div>
<div class="alert alert-success">
  This is a success alert—check it out!
</div>
<div class="alert alert-danger">
  This is a danger alert—check it out!
</div>
<div class="alert alert-warning">
  This is a warning alert—check it out!
</div>
<div class="alert alert-info">
  This is a info alert—check it out!
</div>
<div class="alert alert-light">
  This is a light alert—check it out!
</div>
<div class="alert alert-dark">
  This is a dark alert—check it out!
</div>
```

链接颜色

使用 .alert-link 类可以为带颜色的警告文本框中的链接加上合适的颜色（BootStrap 已经内置了相应的颜色解决方案，会自动对应有一个优化后的链接颜色方案）。

```
<div class="alert alert-primary">
  This is a primary alert with <a href="#">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-primary">
  This is a primary alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-secondary">
  This is a secondary alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
```

```
<div class="alert alert-success">
  This is a success alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-danger">
  This is a danger alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-warning">
  This is a warning alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-info">
  This is a info alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-light">
  This is a light alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
<div class="alert alert-dark">
  This is a dark alert with <a href="#" class="alert-link">网战天下</a>. Give it a click if you like.
</div>
```

额外附加内容

警报还可以包含其他 HTML 元素，如标、段落和分隔符。

```
<div class="alert alert-success">
    <h4 class="alert-heading">Well done!</h4>
    <p>Aww yeah, you successfully read this important alert message. This example text is going to run a bit longer so that you can see how spacing within an alert works with this kind of content.</p>
    <hr>
    <p class="mb-0">Whenever you need to, be sure to use margin utilities to keep things nice and tidy.</p>
</div>
```

关闭警告(小贴士效果)

使用.alert 结合 JavaScript，可以实现警报效果，贴在页面上，并可以自由关闭，其要点如下：

- 确保你正确加载了.alert 警报组件，或者是编译后的 Bootstrap JavaScript 代码组。
- 如果你要自行编译 JavaScript 组件，请将必须的 util.js 包括进去。
- 可以在右上角定义一个.close 关闭按钮效果，则需要在容器中引用 .alert-dismissible 类。
- 警告按钮上要增加 data-dismiss="alert" 触发 JavaScript 动作，同时使用<button>元素，以确保在所有设备上都能获得正确的行为响应。
- 要在关闭警报时生成警报提示，请确保添加.fade 和.show 样式。

(关闭警报会将其从 DOM 中移除)

```
<div class="alert alert-danger alert-dismissible fade show">
    <strong>登录失败!</strong> 您的用户名或密码错误.
    <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>
```

JavaScript 行为

方法

(1) .alert('close')

事件

(1) close.bs.alert
(2) closed.bs.alert

徽章(Badge)

示例

.badge 可以嵌在标题中，并通过标题样式来适配其元素大小，因为其本身是通过相对字体大小和 em 单位的，所以有良好的弹性。

```
<h1>夏季清爽运动鞋 <span class="badge badge-secondary">New</span></h1>
```

```
<h2>夏季清爽运动鞋 <span class="badge badge-secondary">New</span></h2>
<h3>夏季清爽运动鞋 <span class="badge badge-secondary">New</span></h3>
<h4>夏季清爽运动鞋 <span class="badge badge-secondary">New</span></h4>
<h5>夏季清爽运动鞋 <span class="badge badge-secondary">New</span></h5>
<h6>夏季清爽运动鞋 <span class="badge badge-secondary">New</span></h6>
```

徽章可用作链接或按钮的一部分，以提供统计数字样式。

```
<button type="button" class="btn btn-primary">
    通知 <span class="badge badge-light">6</span>
</button>
<!-- <a href="#" class="btn btn-primary">
    通知 <span class="badge badge-light">6</span>
</a> -->
```

情景变化

```
<span class="badge badge-primary">Primary</span>
<span class="badge badge-secondary">Secondary</span>
<span class="badge badge-success">Success</span>
<span class="badge badge-danger">Danger</span>
<span class="badge badge-warning">Warning</span>
<span class="badge badge-info">Info</span>
<span class="badge badge-light">Light</span>
<span class="badge badge-dark">Dark</span>
```

Primary Secondary Success Danger Warning Info Light Dark

椭圆形胶囊标签

使用 .badge-pill 样式，可以使标签更加圆润（具体有较大的 border-radius 边框半径和水平 padding），如果你错过了 V3 的标签这是有用的（这是 Bootstrap 4 中的特色功能）。

```
<span class="badge badge-pill badge-primary">Primary</span>
<span class="badge badge-pill badge-secondary">Secondary</span>
<span class="badge badge-pill badge-success">Success</span>
<span class="badge badge-pill badge-danger">Danger</span>
<span class="badge badge-pill badge-warning">Warning</span>
<span class="badge badge-pill badge-info">Info</span>
<span class="badge badge-pill badge-light">Light</span>
```

```
<span class="badge badge-pill badge-dark">Dark</span>
```

链接

.badge-* 也可以在 <a> 元素上使用，并实现悬停、焦点、状态等效果。

```
<a href="#" class="badge badge-primary">Primary</a>
<a href="#" class="badge badge-secondary">Secondary</a>
<a href="#" class="badge badge-success">Success</a>
<a href="#" class="badge badge-danger">Danger</a>
<a href="#" class="badge badge-warning">Warning</a>
<a href="#" class="badge badge-info">Info</a>
<a href="#" class="badge badge-light">Light</a>
<a href="#" class="badge badge-dark">Dark</a>
```

面包屑导航(Breadcrumb)

```
<nav>
    <ol class="breadcrumb">
        <li class="breadcrumb-item active">首页</li>
    </ol>
</nav>

<nav>
    <ol class="breadcrumb">
        <li class="breadcrumb-item"><a href="#">首页</a></li>
        <li class="breadcrumb-item active">女装</li>
    </ol>
</nav>

<nav>
    <ol class="breadcrumb">
        <li class="breadcrumb-item"><a href="#">首页</a></li>
        <li class="breadcrumb-item"><a href="#">女装</a></li>
        <li class="breadcrumb-item active">连衣裙</li>
    </ol>
</nav>
```

按钮(Button)

示例

```
<button class="btn btn-primary">Primary</button>
<button class="btn btn-secondary">Secondary</button>
<button class="btn btn-success">Success</button>
<button class="btn btn-danger">Danger</button>
<button class="btn btn-warning">Warning</button>
<button class="btn btn-info">Info</button>
<button class="btn btn-light">Light</button>
<button class="btn btn-dark">Dark</button>

<button class="btn btn-link">Link</button>
```

按钮标签

.btn 可以在<button>元素上使用，您也可以在 <a>、或 <input> 元素上使用這些 Class，同样能带来按钮效果（在少数浏览器下会有不同的渲染变异）。

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

轮廓按钮

.btn 在引用中，如果需要一个按钮，但不需要带来的巨大的背景颜色（背景边框缩小）？用默认修饰符类替换.btn-outline-*任何按钮上的所有背景颜色和图像。

```
<button class="btn btn-outline-primary">Primary</button>
<button class="btn btn-outline-secondary">Secondary</button>
```

```
<button class="btn btn-outline-success">Success</button>
<button class="btn btn-outline-danger">Danger</button>
<button class="btn btn-outline-warning">Warning</button>
<button class="btn btn-outline-info">Info</button>
<button class="btn btn-outline-light">Light</button>
<button class="btn btn-outline-dark">Dark</button>
```

尺寸规格与大小定义

配合.btn-lg 、 .btn-sm 两个邻近元素，可分别实现大规格按钮、小规格按钮的定义。

```
<button class="btn btn-primary btn-lg">提交</button>
<button class="btn btn-primary">提交</button>
<button class="btn btn-primary btn-sm">提交</button>

<button class="btn btn-primary btn-block">提交</button>
```

启用与禁用状态

启用：.btn 样式定义的按钮，默认就是启用状态（背景较深、边框较暗、带内阴影），如果你一定要使按钮固定为启用状态、不需要点击反馈，可以增加.active 样式。
禁用：直接将 disabled 布尔属性添加到任何<button>元素（直接嵌套在 html 标签中，使按钮看起来处于非活动的禁用状态（点击不会有响应和弹性）。
使用 <a>标签的禁用有所不同：

- <a>标签不支持 disabled 属性，所以你必须增加 .disabled 属性，使之达到视觉禁用的效果。
- 未来，将包括更多的友好风格，以禁用按钮上的 pointer-events 属性，在支持该属性的浏览器中，会你看不到禁用的光标。

```
<button class="btn btn-primary">提交</button>
<button class="btn btn-primary active">提交</button>
<button class="btn btn-primary" disabled>提交</button>
<a href="#" class="btn btn-primary disabled">提交</a>
```

按钮插件

切换状态

添加 data-toggle="button" 属性，可以切换按钮的 active 状态，如果你预先需要切换按钮，必须将.active 样式。

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

复选框和单选框

Bootstrap 的.button 样式也可以使用于其它元素，比如<label>HTML 组件上，从而实现单选、复选效果。添加 data-toggle="buttons" 到.btn-group 下的元素里，来启用它们的样式切换。

这些按钮的检查状态，只能通过 click 事件 进行更新，如果你使用其它方法更新输入，用 <input type="reset"> 或手动应用输入 checked 属性，人为的在<label> 上增加 .active 状态。

注意：预先选中的按钮需要你手动将 .active 定义上，在 <label>中。

```
<div class="btn-group" data-toggle="buttons">
    <label class="btn btn-secondary active">
        <input type="checkbox" name="checkbox[]" checked> Java
    </label>
    <label class="btn btn-secondary">
        <input type="checkbox" name="checkbox[]"> PHP
    </label>
    <label class="btn btn-secondary">
        <input type="checkbox" name="checkbox[]"> Python
    </label>
</div>

<div class="btn-group" data-toggle="buttons">
    <label class="btn btn-secondary active">
        <input type="radio" name="radio" checked> Java
    </label>
    <label class="btn btn-secondary">
        <input type="radio" name="radio"> PHP
    </label>
    <label class="btn btn-secondary">
        <input type="radio" name="radio"> Python
    </label>
```

```
</div>
```

上面的实例对应传统使用环境。Bootstrap 4 提供了.btn-group-toggle 全新的复选与单选 解决方案：

```
<div class="btn-group-toggle" data-toggle="buttons">
    <label class="btn btn-secondary active">
        <input type="checkbox"> Checked
    </label>
</div>

<div class="btn-group btn-group-toggle" data-toggle="buttons">
    <label class="btn btn-secondary active">
        <input type="checkbox" name="checkbox[]" checked> Java
    </label>
    <label class="btn btn-secondary">
        <input type="checkbox" name="checkbox[]"> PHP
    </label>
    <label class="btn btn-secondary">
        <input type="checkbox" name="checkbox[]"> Python
    </label>
</div>

<div class="btn-group btn-group-toggle" data-toggle="buttons">
    <label class="btn btn-secondary active">
        <input type="radio" name="radio" checked> Java
    </label>
    <label class="btn btn-secondary">
        <input type="radio" name="radio"> PHP
    </label>
    <label class="btn btn-secondary">
        <input type="radio" name="radio"> Python
    </label>
</div>
```

JavaScript 行为：

方法

(1) .button('toggle')：切换状态，给予按钮已经启用的外观。

下拉菜单(Dropdowns)

示例

将下拉列表的切换（按钮或链接）和下拉菜单包含在.dropdown 中，或者另外声明 position: relative;元素;可以从<a> 或 <button>触发下拉菜单，以适应你的使用的需求。

单一按钮的下拉菜单

任何一个 .btn 块都可以定义变更为下拉菜单，可以使用<button>或<a>元素做下拉菜单的示例。

```
<div class="dropdown">
<!-- <div class="btn-group"> -->
    <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown">Dropdown</button>
    <!-- <a href="#" class="btn btn-primary dropdown-toggle" data-toggle="dropdown">Dropdown</a> -->

    <div class="dropdown-menu">
        <a href="#" class="dropdown-item">aaa</a>
        <a href="#" class="dropdown-item">bbb</a>
        <a href="#" class="dropdown-item">ccc</a>
    </div>
</div>
```

可以.btn 颜色及样式类来定义下拉菜单的外在表现：

下拉菜单通过切换.show 父列表项上的类来切换隐藏内容。

分裂式按钮下拉菜单

```
<div class="btn-group">
    <button class="btn btn-success">Dropdown</button>
```

```
    <button class="btn btn-success dropdown-toggle dropdown-toggle-split" data-toggle="dropdown"></button>

    <div class="dropdown-menu">
        <a href="#" class="dropdown-item">AAA</a>
        <a href="#" class="dropdown-item">BBB</a>
        <a href="#" class="dropdown-item">CCC</a>
    </div>
</div>
```

尺寸大小

通过添加 btn.btn-lg/sm 改变下拉框尺寸大小。

变形

可以用 .dropup、.dropright、.dropleft 改变下拉菜单的指向。

```
<div class="btn-group dropleft">
    <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown">Dropdown</button>
    <div class="dropdown-menu">
        <a href="#" class="dropdown-item">aaa</a>
        <a href="#" class="dropdown-item">bbb</a>
        <a href="#" class="dropdown-item">ccc</a>
    </div>
</div>

<div class="btn-group">
    <div class="btn-group dropleft">
        <button class="btn btn-success dropdown-toggle dropdown-toggle-split" data-toggle="dropdown"></button>
        <div class="dropdown-menu">
            <a href="#" class="dropdown-item">AAA</a>
            <a href="#" class="dropdown-item">BBB</a>
            <a href="#" class="dropdown-item">CCC</a>
        </div>
    </div>
    <button class="btn btn-success">Dropdown</button>
</div>
```

菜单

旧版 Boostrap(v3)下拉菜单中的子菜单项必须是链接，但 v4 不再是这种情况，现在你可选择使用<button>下拉列表中的元素，而不是仅仅 <a>标签。
你可以创建非交互式下拉菜单项 .dropdown-item-text. 还可以随意使用定制的 CSS 或文本实用程序进一步设计样式.

```
<div class="dropdown">
    <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown">Dropdown</button>
    <div class="dropdown-menu show">
        <a href="#" class="dropdown-item">aaa</a>
        <button class="dropdown-item">bbb</button>
        <span class="dropdown-item-text">ccc</span>
    </div>
</div>
```

有效&不可用

加上 .active 让下拉列表中的项 样式为有效菜单。
加上.disabled 让下拉列表中的项 样式为不可用菜单。

对齐

默认情况下，一个下拉菜单自动从顶部和左侧的父级 100%定位。添加.dropdown-menu-right 到.dropdown-menu 右侧轻松对齐下拉菜单。

响应式对齐:
如果你想使用响应式对齐，请通过添加 data-display="static"属性禁用动态定位，并使用响应式变体类。
为了下拉菜单左/右对齐和给定断点或更大的断点，加上.dropdown-menu-{sm|-md|-lg|-xl}-left/right。
你不需要添加 data-display="static"属性设置为**导航栏**中的下拉按钮，因为导航条中不使用 popper.js。

```
<div class="btn-group">
    <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown" data-display="static">Dropdown Dropdown Dropdown Dropdown</button>
```

```
    <div class="dropdown-menu dropdown-menu-lg-right">
        <a href="#" class="dropdown-item">aaa</a>
        <a href="#" class="dropdown-item">bbb</a>
        <a href="#" class="dropdown-item">ccc</a>
    </div>
</div>
```

内容

头部

添加 h6.dropdown-header 标题来标记任何下拉菜单中的操作部分。

分割线

使用 div.dropdown-divider 分隔符分割相关菜单子项，呈现出分组和分割线效果。

文本

在文本下拉菜单中放置任何自由格式的文本并使用间隔工具。请注意，您可能需要额外的大小调整样式来限制菜单宽度。

```
<div class="dropdown-menu p-4 text-muted" style="width:200px;">
    <p>一些示例文本在下拉菜单中自由流动。</p>
    <p>这是更多示例文本。</p>
</div>
```

表单

将表单放在下拉菜单中，或将其放入下拉菜单中，并使用 margin 或 padding 通用 CSS 样式调整空间和位置。

```
<div class="dropdown-menu p-3" style="width:240px;">
    <form action="">
```

```
            <div class="form-group">
                <label for="">邮箱：</label>
                <input type="email" class="form-control">
            </div>
            <div class="form-group">
                <label for="">密码：</label>
                <input type="password" class="form-control">
            </div>
            <button class="btn btn-primary">登录</button>
        </form>
</div>
<!-- <form action="" class="dropdown-menu p-3" style="width:240px;">
        <div class="form-group">
            <label for="">邮箱：</label>
            <input type="email" class="form-control">
        </div>
        <div class="form-group">
            <label for="">密码：</label>
            <input type="password" class="form-control">
        </div>
        <button class="btn btn-primary">登录</button>
</form> -->
```

下拉选项

使用 data-offset 或 data-reference 更改下拉菜单的位置。

```
<button class="btn btn-primary dropdown-toggle" data-toggle="dropdown" data-offset="10,20">Dropdown</button>

<button class="btn btn-success">Dropdown</button>
<button class="btn btn-success dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" data-reference="parent"></button>
```

JavaScript 行为

事件

(1) show.bs.dropdown：当调用 show 显示方法时，此事件会立即触发。

(2) shown.bs.dropdown：当下拉菜单对用户可见时，会触发此事件(将等待 CSS 转换完成)。
(3) hide.bs.dropdown：当调用隐藏实例方法时，会立即触发此事件。
(4) hidden.bs.dropdown：当下拉菜单从用户隐藏完毕时，会触发此事件(将等待 CSS 转换完成)。

按钮组(Btn-group)

基本示例

将一系列的 .btn 包裹在.btn-group 内，并使用我们提供的插件，可以实现选择按钮、选取块状区的行为功能。
大小尺寸：.btn-group-lg/sm

```
<div class="btn-group btn-group-lg">
    <button type="button" class="btn btn-secondary">Left</button>
    <button type="button" class="btn btn-secondary">Middle</button>
    <button type="button" class="btn btn-secondary">Right</button>
</div>
```

按钮工具栏

用.btn-toolbar 定义按钮工具栏，根据需要使用样式定义，对按钮进行群组、间隔等定义，将按钮组的组合组合成为更复杂组件的按钮工具栏。

```
<div class="btn-toolbar">
    <div class="btn-group mr-2">
        <button type="button" class="btn btn-secondary">1</button>
        <button type="button" class="btn btn-secondary">2</button>
        <button type="button" class="btn btn-secondary">3</button>
        <button type="button" class="btn btn-secondary">4</button>
    </div>

    <div class="btn-group mr-2">
        <button type="button" class="btn btn-secondary">5</button>
        <button type="button" class="btn btn-secondary">6</button>
        <button type="button" class="btn btn-secondary">7</button>
    </div>

    <div class="btn-group">
        <button type="button" class="btn btn-secondary">8</button>
```

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

嵌套

将.btn-group 放在另一个 .btn-group 里，可以实现按钮组与**下拉菜单**的组合。

```
<div class="btn-group-vertical">
    <button type="button" class="btn btn-primary">Left</button>
    <button type="button" class="btn btn-primary">Middle</button>

    <div class="btn-group">
        <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown">Dropdown</button>
        <div class="dropdown-menu">
            <a href="#" class="dropdown-item">aaa</a>
            <a href="#" class="dropdown-item">bbb</a>
            <a href="#" class="dropdown-item">ccc</a>
        </div>
    </div>

    <div class="btn-group">
        <button class="btn btn-primary">Dropdown</button>
        <button class="btn btn-primary dropdown-toggle dropdown-toggle-split" data-toggle="dropdown"></button>
        <div class="dropdown-menu">
            <a href="#" class="dropdown-item">aaa</a>
            <a href="#" class="dropdown-item">bbb</a>
            <a href="#" class="dropdown-item">ccc</a>
        </div>
    </div>
</div>
```

垂直排列

用.btn-group-vertical 将一组按钮垂直排列，而不是水平排列，**不支持**分割式下拉菜单的定义。

```
<div class="btn-group-vertical">
    <button type="button" class="btn btn-primary">Left</button>
    <button type="button" class="btn btn-primary">Middle</button>
```

```
    <div class="btn-group">
        <button class="btn btn-primary dropdown-toggle" data-toggle="dropdown">Dropdown</button>
        <div class="dropdown-menu">
            <a href="#" class="dropdown-item">aaa</a>
            <a href="#" class="dropdown-item">bbb</a>
            <a href="#" class="dropdown-item">ccc</a>
        </div>
    </div>
</div>
```

Input 输入框及输入框群组

基本示例

规格尺寸定义: 将相对表单大小的 class 样式加到 .input-group 中, 其内容会自动调整大小, 如.input-group-lg、.input-group-sm， 不需要在每个元素上重重使用样式控制其大小。

```
<label for="">商品价格</label>
<div class="input-group">
    <div class="input-group-prepend">
        <span class="input-group-text">$</span>
    </div>

    <input type="text" class="form-control">
    <!-- <textarea class="form-control"></textarea> -->

    <div class="input-group-append">
        <span class="input-group-text">.00</span>
    </div>
</div>
```

输入组插件

勾选或单选框组合

```
<div class="input-group-prepend">
```

```
    <div class="input-group-text">
        <input type="checkbox/radio" name="" id="">
    </div>
</div>
```

多个输入

尽管可视化支持多个 <input>但验证样式仅适用于具有单个<input>的输入组。

多类型控件组合

支持多种控件结合，比如复选框和、文本、input 框混合使用。

```
<div class="input-group-prepend">
    <span class="input-group-text">$</span>
    <span class="input-group-text">&yen;</span>
</div>
```

按钮组合

```
<div class="input-group-prepend">
    <button class="btn btn-outline-secondary">Button</button>
    <!-- <button class="btn btn-outline-secondary">Button</button> -->
</div>
```

下拉菜单

```
<div class="input-group-prepend">
    <button class="btn btn-outline-secondary dropdown-toggle" data-toggle="dropdown">Dropdown</button>
    <div class="dropdown-menu">
        <a href="" class="dropdown-item">aaa</a>
        <a href="" class="dropdown-item">bbb</a>
```

```
            <a href="" class="dropdown-item">ccc</a>
        </div>
</div>
<div class="input-group-prepend">
        <button class="btn btn-outline-secondary">Dropdown</button>
        <button class="btn btn-outline-secondary dropdown-toggle dropdown-toggle-split" data-toggle="dropdown"></button>
        <div class="dropdown-menu">
            <a href="" class="dropdown-item">aaa</a>
            <a href="" class="dropdown-item">bbb</a>
            <a href="" class="dropdown-item">ccc</a>
        </div>
</div>
```

自定义表单

输入组包括对自定义选择和自定义文件输入的支持。 这些浏览器的默认版本不受支持。

```
<select class="custom-select">
        <option value="">-请选择-</option>
        <option value="">aaa</option>
        <option value="">bbb</option>
        <option value="">ccc</option>
</select>

<div class="custom-file">
        <input type="file" class="custom-file-input">
        <label for="" class="custom-file-label">选择</label>
</div>
```

表单(Form)

表单控件

1. 文本控件（如 <input>、<select>、 <textarea>）统一采用 .form-control 样式进行处理优化，包括常规外观、focus 选（点）中状态、尺寸大小等。
2. 对于 input **文件**选择控件，Bootstrap v4 采用.form-control-file 取代了.form-control。
3. **大小规格**：使用 .form-control-lg 和 .form-control-sm 属性定控件大小高度。
4. **输入范围**：使用设置水平滚动范围输入.form-control-range。

5. 只读属性：在 input 控件上增加 readonly (布尔值)标签定义，以防止修改 input 中的值。仅能阅读的 input 控件显示较谈(就像禁用的输入框)，但保留鼠标效果。
6. **只读纯文本**：如果你希望将 <input readonly>属性进一步处理，显示为纯文本（没有控件框），你只要引用 .form-control-plaintext class 样式，就能移除预设的表单样式，并保留适当的边距和填充间隙。
7. **提示文本**：small.form-text.text-muted

```
<form action="">
    <div class="form-group">
        <label for="">邮箱</label>
        <input type="text" class="form-control">
    </div>
    <div class="form-group">
        <label for="">密码</label>
        <input type="password" class="form-control">
    </div>
    <div class="form-check">
        <input type="checkbox" class="form-check-input">
        <label for="" class="form-check-label">记住我</label>
    </div>
    <button class="btn btn-primary">登录</button>
</form>
```

复选框与单选框

使用.form-check 可以格式化复选框和单选框按钮，用以**改进它们的默认布局和动作呈现**，复选框用于在列表中选择一个或多个选项，单选框则用于多许多选项中选择一个。
复选框和单选框也是可以禁用的，只要 not-allowed 在父级的悬停上提供定义，<label>需要将该.disabled 类添加到父级.form-check，同时控件也会淡化文字颜色以灰色显示禁用状态。

默认堆叠

```
<div class="form-group">
    <div class="form-check">
        <input type="checkbox/radio" class="form-check-input">
        <label class="form-check-label">苹果</label>
    </div>
    <div class="form-check">
        <input type="checkbox/radio" class="form-check-input">
        <label class="form-check-label">香蕉</label>
```

```
    </div>
    <div class="form-check">
        <input type="checkbox/radio" class="form-check-input" disabled>
        <label class="form-check-label">橘子</label>
    </div>
</div>
```

水平排列

通过添加 .form-check-inline 到任何一个组，会使 加到任何.form-check 中的选取框平行排列。

没有标签

添加 .position-static 到 .form-check 选择器上，可以实现没有文本的输入。

```
<div class="form-check">
    <input type="checkbox/radio" class="form-check-input position-static">
</div>
```

布局

表单栅格排列

可使用我们的栅格系统构建更复杂的表单，包括建立多列、多种宽度和其它对齐选项的布局。

```
<form>
    <div class="row">
        <div class="col">
            <input type="email" class="form-control" placeholder="邮箱">
        </div>
        <div class="col">
            <input type="password" class="form-control" placeholder="密码">
        </div>
    </div>
</form>
```

你也可以使用 .form-row 来取代.row（它们二者很多时候可以互换使用），因为.form-row 提供更小的沟槽缝隙。

```
<form>
    <div class="form-row">
        <div class="form-group col-sm-8">
            <label>邮箱</label>
            <input type="email" class="form-control" placeholder="邮箱">
        </div>
        <div class="form-group col-sm-4">
            <label>密码</label>
            <input type="password" class="form-control" placeholder="密码">
        </div>
    </div>
</form>
```

垂直排列表单

通过添加 .row class 类，并使用 .col-*-* 等栅格组件来指定标签和宽度，可以建立起水平表单。

确保添加.col-form-label 到您<label>上，以便他们垂直居中与他们相关的表单控件。<legend>元素，可以.col-form-legend 样式定义，与普通<label>元素相似。.

```
<form>
    <div class="form-group row">
        <label class="col-sm-1 col-form-label">邮箱</label>
        <div class="col-sm-11">
            <input type="email" class="form-control" placeholder="邮箱">
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-1 col-form-label">密码</label>
        <div class="col-sm-11">
            <input type="password" class="form-control" placeholder="密码">
        </div>
    </div>
    <div class="row">
        <div class="offset-sm-1 col-sm-11">
            <button class="btn btn-primary">登录</button>
        </div>
    </div>
</form>
```

垂直排列表单尺寸规格定义:
使用.col-form-label-sm、.col-form-label-lg 到 <label>上，可以定义控件大小，还有 .form-control-lg、.form-control-sm 样式也起相应作用。

自动调整大小

下面的示例使用一个 flexbox 弹性布局垂直居中的内容，我们将.col 改为.col-auto，这样的列只占用本身内容所需要的宽度，换句话说列的大小就是内容的大小（宽度）

```
<form>
    <div class="form-row">
        <div class="col-auto">
            <input type="email" class="form-control" placeholder="邮箱">
        </div>
        <div class="col-auto">
            <input type="password" class="form-control" placeholder="密码">
        </div>
        <div class="col-auto">
            <button class="btn btn-primary">登录</button>
        </div>
    </div>
</form>
```

内联式表单

使用 .form-inline 样式在单个水平行上显示一系列标签，表单控件和按钮。内联表单中的表单控件与默认状态略有不同:

- 基于 display: flex 控件组件，并允许您使用 间隙隔离 和 flexbox 弹性布局样式。
- 控制组件和 input 接受 width: auto 以覆盖预设的 width: 100%。
- 控制组件只会在 viewport 576px 宽度 时才会显示在行内，以便在移动设备上完整呈现。

```
<form class="form-inline">
    <input type="email" class="form-control mb-2 mr-sm-2" placeholder="邮箱">
    <input type="password" class="form-control mb-2 mr-sm-2" placeholder="密码">
    <div class="form-check mb-2 mr-sm-2">
        <input type="checkbox" class="form-check-input">
        <label for="" class="form-check-label">记住我</label>
    </div>
```

```
    <button class="btn btn-primary">登录</button>
</form>
```

禁用表单

```
<form>
    <fieldset disabled>
    ……
    </fieldset>
</form>
```

验证

自定义样式

对于自定义 Bootstrap 表单验证消息，您需要将 novalidate 属性添加到您的<form>。这将禁用浏览器默认的反馈工具提示，但仍提供对 JavaScript 中的表单验证 API 有效支持。尝试提交以下表格；我们的 JavaScript 将拦截提交按钮并向您传递反馈:
尝试提交时，您将看到:invalid 和 :valid 的样式应用于表单控件。

```
<form action="" novalidate class="needs-validation">
    <div class="form-group">
        <label for="">邮箱</label>
        <input type="text" class="form-control" placeholder="邮箱" required>
        <div class="invalid-feedback">邮箱格式不正确!</div>
    </div>
    <div class="form-group">
        <label for="">密码</label>
        <input type="password" class="form-control" placeholder="密码" required>
        <div class="invalid-feedback">密码格式不正确!</div>
    </div>
    <div class="form-check">
        <input type="checkbox" name="" id="" class="form-check-input">
        <label for="" class="form-check-label">记住我</label>
    </div>
    <button class="btn btn-primary">登录</button>
</form>

<script>
```

```
    // Example starter JavaScript for disabling form submissions if there are invalid fields
    (function() {
        'use strict';
        window.addEventListener('load', function() {
            // Fetch all the forms we want to apply custom Bootstrap validation styles to
            var forms = document.getElementsByClassName('needs-validation');
            // Loop over them and prevent submission
            var validation = Array.prototype.filter.call(forms, function(form) {
                form.addEventListener('submit', function(event) {
                    if (form.checkValidity() === false) {
                        event.preventDefault();
                        event.stopPropagation();
                    }
                    form.classList.add('was-validated');
                }, false);
            });
        }, false);
    })();
</script>
```

服务器端

我们建议使用客户端验证，但是如果您需要使用服务器端验证，则可以使用.is-invalid 和.is-valid 来表示无效和有效的表单字段。注意，.invalid-feedback 这些类也支持。

```
<form>
    <div class="form-group">
        <label>邮箱</label>
        <input type="email" class="form-control is-valid" placeholder="邮箱">
        <div class="valid-feedback">邮箱正确!</div>
        <div class="invalid-feedback">邮箱不正确!</div>
    </div>
    <div class="form-group">
        <label>密码</label>
        <input type="password" class="form-control is-invalid" placeholder="密码">
        <div class="valid-feedback">密码格式正确!</div>
        <div class="invalid-feedback">密码格式不正确!</div>
    </div>
    <button class="btn btn-primary">登录</button>
</form>
```

自定义表单

为了使自定义表单和跨浏览器保持一致性，请使用自定义的表单元素来替换浏览器的默认值，它们建立在语义和具备友了的标记之上，因此它们是可以替代任何默认表单控制元件的。

Checkbox 勾选

```
<div class="custom-control custom-checkbox">
    <input type="checkbox" class="custom-control-input" id="remember">
    <label class="custom-control-label" for="remember">记住我</label>
</div>
```

Radio 单选项

```
<div class="custom-control custom-radio">
    <input type="radio" name="sex" id="man" class="custom-control-input">
    <label class="custom-control-label" for="man">男</label>
</div>
<div class="custom-control custom-radio">
    <input type="radio" name="sex" id="woman" class="custom-control-input">
    <label class="custom-control-label" for="woman">女</label>
</div>
```

一致：在.custom-control 添加.custom-control-inline。

IOS 风格开关:
开关具有自定义复选框的标记，使用 .custom-switch 类来呈现切换开关。开关还支持 disabled 属性(v4.2.1 新增组件)。

```
<div class="custom-control custom-switch">
    <input type="checkbox" class="custom-control-input" id="customSwitch1">
    <label class="custom-control-label" for="customSwitch1">Toggle this switch element</label>
</div>
```

Select 下拉选择菜单

自定义<select>下拉选择菜单只需要一个.custom-select CSS 即可触发自定义样式。

```
<select class="custom-select">
    <option>-请选择-</option>
    <option>aaa</option>
    <option>bbb</option>
    <option>ccc</option>
</select>
```

大小和尺寸：可以在.custom-select 上添加.custom-select-lg/sm 改变大小。

Range 范围

创建自定义<input type="range">与控制.custom-range。轨道（背景）和大拇指（值）都被设置为跨浏览器显示相同。由于只有 IE 和 Firefox 支持从拇指的左侧或右侧“填充”它们的轨迹，以作为视觉指示进度的手段，所以我们目前不支持它。

```
<input type="range" class="custom-range">
```

File 文件浏览器

文件浏览（选取）是比较原始粗糙的，它需要额外的 JavaScript 定义支持，如果你将 Choose file…文件选取和所选文件的名称关联。

```
<div class="custom-file">
    <input type="file" class="custom-file-input" id="avatar">
    <label class="custom-file-label" for="avatar">选择文件</label>
</div>
```

轮播效果(Carousel)

轮播不带幻灯片尺寸标准化处理，因此你可能需要使用其它通用样式可自定义样式来调整其大小使之适当匹配。虽然轮播组件支持上一个/下一个控制和指令，但它们不是必备元素，可根据你的需要添加或自定义（展现不同的效果）。

通过 .carousel 命名样式引入轮播组件，同时为此控件设置唯的 ID-尤其是当你在同一页面

使用多个轮播效果时这是必须的。
这是一个经典的幻灯片示例，请注意轮播上的图像引用了 .d-block 、 .w-100 两个样式，以修正浏览器预设的图像对齐带来的影响。
将 .active 样式添加到其中一个幻灯片（一般第一张），否则轮播效果将无法正常运行（展现）。

```
<div class="carousel slide" data-ride="false/carousel" data-interval="5000" data-wrap="true/false" data-pause="hover/false">
    <div class="carousel-inner">
        <div class="carousel-item active">
            <img src="images/slider/1.jpg" class="d-block w-100" alt="">
        </div>
        <div class="carousel-item">
          <img src="images/slider/2.jpg" class="d-block w-100" alt="">
        </div>
        <div class="carousel-item">
          <img src="images/slider/3.jpg" class="d-block w-100" alt="">
        </div>
    </div>
</div>
```

带控制器的效果

加上了上一个/下一个控制器:

```
<div class="carousel-inner">
    ……
</div>
<a class="carousel-control-prev" href="#myCarousel" data-slide="prev">
    <span class="carousel-control-prev-icon"></span>
</a>
<a class="carousel-control-next" href="#myCarousel" data-slide="next">
    <span class="carousel-control-next-icon"></span>
</a>
```

包含姿态指示器

也可以将当前所在幻灯片状态指示器添加到轮播效果控件中:

```
<ol class="carousel-indicators">
    <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
    <li data-target="#myCarousel" data-slide-to="1"></li>
```

```
    <li data-target="#myCarousel" data-slide-to="2"></li>
</ol>
<div class="carousel-inner">
    ……
</div>
```

包含字幕的轮播

在 .carousel-item 中使用 .carousel-caption 添加字幕到您的轮播控件中，如果是输小的浏览器 viewport 上，会自动隐藏（隐藏文字呈现主图片轮播），引用的是.d-none 定义，一旦到了中型 md 浏览设备或屏幕则调用.d-md-block 样式使之呈现。

```
<div class="carousel-item active">
    <img src="images/slider/1.jpg" class="d-block w-100" alt="">
    <div class="carousel-caption d-none d-sm-block">
        <h5>第一张图</h5>
        <p>第一张图描述！第一张图描述！第一张图描述！</p>
    </div>
</div>
```

交替变化

加上 .carousel-fade 到你的滑动里，使交替变化代替滑动。

单个间隔

加上 data-interval="" 到一个 .carousel-item 更改自动循环到下一项之间的延迟时间.

JavaScript 行为

选项

可以透過資料屬性或 JavaScript 調整選項。對於資料屬性，將選項名稱附加到 data-，如

data-interval=""。 可以通过数据属性或 JavaScript 调整（传递）选项，对于数据属性，选项名称追加到 data-，如 data-interval=""。

名称	Type类型	默认值	描述
interval	number	5000	自动循环项目之间的延迟时间（即滚动时间），。果为false，则传送带不会滚动。
keyboard	boolean	true	旋转木马是否应对键盘事件作出反应。
pause	string | boolean	"hover"	如果设置为"hover"，则鼠标移在动画屏幕上暂停旋转，并在移开鼠标后恢复旋转事件（这是默认属性）；如设置为false,则鼠标移上去轮播动画不会暂停。 在触摸屏幕上，当设置为"hover"属性时，循环将在触控时暂停（一旦用户完成与旋转事件的交互）两个时间间隔自动恢复。 请注意，这是上述鼠标行为的补充。
ride	string	false	在用户手动循环第一个项目后自动播放传送带， 如果"carousel"则加载时自动播放传送带。
wrap	boolean	true	转盘是否应该连续循环或难以停止。

方法

(1) .carousel(options)：通过 object 初始化，启动并执行轮播。

```
$('#myCarousel').carousel({
      interval:2000,
});
```

(2) .carousel('cycle')：从左到右循环播放。
(3) .carousel('pause')：通过事件停止幻灯片播放。
(4) .carousel(number)：将轮播循环到特定的帧（基于 0，类似数组），在 **目标被显示之前**回传给调用用者（即 slid.bs.carousel 事件之前)。
(5) .carousel('prev')：将轮播指向前一帧幻灯片，**在前一个目标被显示之前**回传给调用者（即 slid.bs.carousel 事件之前)。
(6) .carousel('next')：将轮播指向后一帧幻灯片，**在前一个目标被显示之前**回传给调用者（即 slid.bs.carousel 事件之前)。

事件

所有的轮播事件都在轮播本身（即)下被触发。

事件类型	描述
slide.bs.carousel	当用 slide 方法时，此事件会立即触发。
slid.bs.carousel	轮播完成切换后，此事件即被触发。

Bootstrap 提供了两下事件给轮播控件使用，这两个事件都具有以下附加属性：

- direction：轮播滚动的方向（"left" 或 "right"）。
- relatedTarget：作为活动项目滑动到指定的 DOM 元素。
- from：当前项目的项目的索引。
- to:下一个项目的索引。

```
$('#myCarousel').on('slide.bs.carousel', function(options){

});
```

Hero 广告大块屏幕(Jumbotron)

```
<div class="jumbotron">
    <h1 class="display-3">JavaScript</h1>
    <p class="lead">JavaScript 是一种属于网络的脚本语言,已经被广泛用于 Web 应用开发,常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。</p>
    <hr class="my-4">
    <p>通常 JavaScript 脚本是通过嵌入在 HTML 中来实现自身的功能的。</p>
    <p class="lead">
        <a href="" class="btn btn-primary btn-lg">了解更多</a>
    </p>
</div>
```

想要让广告大屏幕占满当前显示浏览器全屏、不带有圆角，只要添加.jumbotron-fluid CSS 修饰符，并在里面添加一个.container 或 .container-fluid 内容空间即可。

```
<div class="jumbotron jumbotron-fluid">
    <div class="container">
        <h1 class="display-4">JavaScript</h1>
        <p class="lead">JavaScript 是一种属于网络的脚本语言,已经被广泛用于 Web 应用开发,常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。</p>
    </div>
</div>
```

列表组(List-group)

基本示例

启用：添加 .active 到 .list-group-item 下的其中一行或多行，以指示当前有效的选择状态。

禁用：添加 .disabled 到 .list-group-item 下的其中一行或多行，以显示出 禁用 状态。注意：一些带有 .disabled 的元素还需要自定义 JavaScript 脚本才能完全禁用其点击事件（如链接）。

```
<ul class="list-group">
    <li class="list-group-item active">aaa</li>
    <li class="list-group-item">bbb</li>
    <li class="list-group-item disabled">ccc</li>
</ul>
```

链接和按钮

使用 <a>或 <button>来创建具有 hover、禁用、悬停和活动状态的列表组.list-group-item-action，我们分离这些 Class 样式，以确保由非交互元素组成的列表群组(如 或)不提供可点击或触击（即可以用一个父 代替<ul>）。

```
<div class="list-group">
    <a href="#" class="list-group-item list-group-item-action active">aaa</a>
    <a href="#" class="list-group-item list-group-item-action">bbb</a>
    <a href="#" class="list-group-item list-group-item-action disabled">ccc</a>
</div>
```

使用<button>元素，你还可以使用 disabled 属性来达到禁用状态指示（或引用 .disabled 样式），不过这一属性不支持 HTML5 中的 <a>标签。

```
<div class="list-group">
    <button class="list-group-item list-group-item-action active">aaa</button>
    <button class="list-group-item list-group-item-action">bbb</button>
    <button class="list-group-item list-group-item-action" disabled>ccc</button>
</div>
```

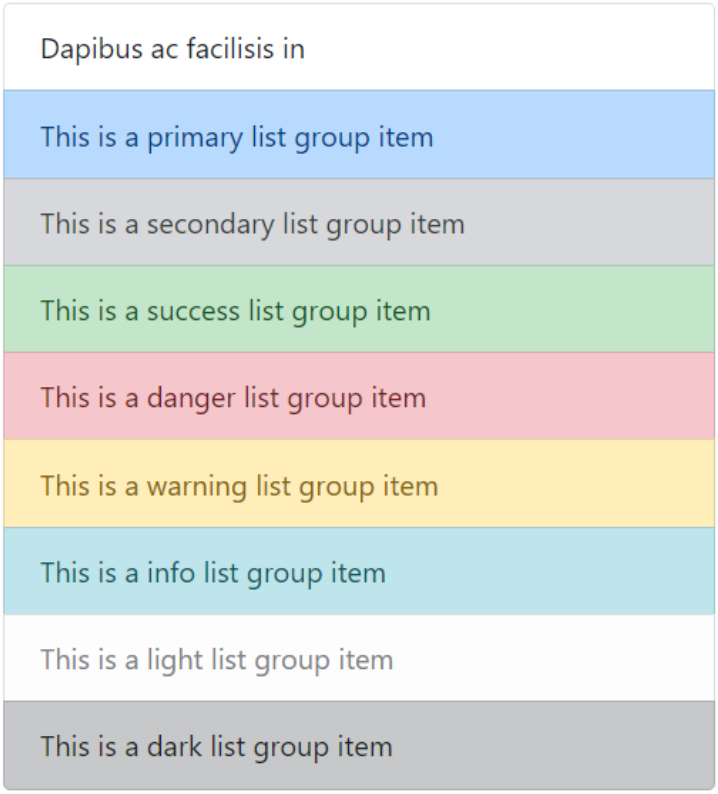
Flush 紧致贴齐

加入.list-group-flush 选择器，可以实现移除部分边框以及圆角，从而产生边缘贴齐的列表组，这在诸如 Card 卡片等容器中很实用（达成更好的呈现效果）。

上下文语境颜色呈现样式

使用情景式 class 样式来设计具有状态背景和颜色的列表组，呈现默认或链接状态。

```
<ul class="list-group">
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item list-group-item-primary">This is a primary list group item</li>
    <li class="list-group-item list-group-item-secondary">This is a secondary list group item</li>
    <li class="list-group-item list-group-item-success">This is a success list group item</li>
    <li class="list-group-item list-group-item-danger">This is a danger list group item</li>
    <li class="list-group-item list-group-item-warning">This is a warning list group item</li>
    <li class="list-group-item list-group-item-info">This is a info list group item</li>
    <li class="list-group-item list-group-item-light">This is a light list group item</li>
    <li class="list-group-item list-group-item-dark">This is a dark list group item</li>
</ul>
```



情景式 class 也可以使用 .list-group-item-action 样式，注意，在上述示例中，不存在 hover 样式指示器，事实上它是支持的，而且还支持 e .active 状态指示--我们可以应用它们在上

下文情景列表组的项目上进行活动状态选择指示。

```
<div class="list-group">
    <a href="#" class="list-group-item list-group-item-action">Dapibus ac facilisis in</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-primary">This is a primary list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-secondary">This is a secondary list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-success">This is a success list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-danger">This is a danger list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-warning">This is a warning list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-info">This is a info list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-light">This is a light list group item</a>
    <a href="#" class="list-group-item list-group-item-action list-group-item-dark">This is a dark list group item</a>
</div>
```

引入 badge 微章

在通用样式定义的帮助下，可向任何列表项目添加.badge 标签以显示未读计数、活动状态等。

```
<ul class="list-group">
    <li class="list-group-item d-flex justify-content-between align-items-center">
        aaa
        <span class="badge badge-primary badge-pill">14</span>
    </li>
    <li class="list-group-item d-flex justify-content-between align-items-center">
        bbb
        <span class="badge badge-primary badge-pill">2</span>
    </li>
    <li class="list-group-item d-flex justify-content-between align-items-center">
        ccc
        <span class="badge badge-primary badge-pill">1</span>
    </li>
</ul>
```

自定义内容

在 flexbox 通用样式定义的支持下，列表组中几乎可以添加任意的 HTML 内容，包括标签、内容、链接。

```
<div class="list-group">
    <a href="" class="list-group-item list-group-item-action">
        <div class="d-flex w-100 justify-content-between">
            <h5 class="mb-1">标题</h5>
            <small>一天前</small>
        </div>
        <p class="mb-1">描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述</p>
        <small>子内容子内容子内容</small>
    </a>
</div>
```

JavaScript 行为

使用数据属性：.list-group-item 上使用数据属性，可以指定 data-toggle="list" 或在元素上编写任意的 JavaScript 事件来激活列表组导航，而需要编写任何的 JavaScript 脚本。

fade 淡入淡出效果：要使定位的元素有淡入淡出效果，请将.fade 添加到每个 .tab-pane 子项中，且第一个列表项目定义 .show 样式使之初始可见。

```
<div class="row">
    <div class="col-3">
        <div class="list-group">
            <a href="#home" class="list-group-item list-group-item-action active" data-toggle="list">Home</a>
            <a href="#profile" class="list-group-item list-group-item-action" data-toggle="list">Profile</a>
            <a href="#messages" class="list-group-item list-group-item-action" data-toggle="list">Messages</a>
            <a href="#settings" class="list-group-item list-group-item-action" data-toggle="list">Settings</a>
        </div>
    </div>
    <div class="col-9">
        <div class="tab-content">
            <div class="tab-pane fade show active" id="home">1111111111</div>
            <div class="tab-pane fade" id="profile">2222222222</div>
```

```
            <div class="tab-pane fade" id="messages">3333333333</div>
            <div class="tab-pane fade" id="settings">4444444444</div>
        </div>
    </div>
</div>
```

事件

(1) show.bs.tab：此事件在标签显示时触发，但在新标签显示之前。使用 event.target 和 event.relatedTarget 将目前与此前启用（如果可用）的作为目标定位。

(2) shown.bs.tab：此事件发生在选项卡显示后、新选项卡被显示之前，使用 event.target 和 event.relatedTarget 分别定位启用中的选项卡和上一个活动选项卡（如果可用）。

(3) hide.bs.tab：当要显示新的选项卡（因此先前的活动选项卡将被隐藏）时，此事件将触发。分别使用 event.target 和 event.relatedTarget 定位当前活动选项卡和新的即将启用的的选项卡。

(4) hidden.bs.tab：在显示新标签页之后触发此事件（因此先前的活动标签页被隐藏），使用 event.target 和 event.relatedTarget 分别定位上一个活动选项卡和新的活动选项卡。

```
$('a[data-toggle="list"]').on('shown.bs.tab', function (e) {
    console.log(e.target) // newly activated tab
    console.log(e.relatedTarget) // previous active tab
});
```

媒体对象/图文混排(Media-object)

示例

```
<div class="media">
    <img src="images/sm.jpg" alt="" class="mr-3" width="64">
    <!-- <a href="" class="pr-3"><img src="images/sm.jpg" alt="" width="64"></a> -
->
    <div class="media-body">
        <h5 class="mt-0">标题</h5>
        内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！
内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！
    </div>
    <!-- <img src="images/sm.jpg" alt="" class="ml-3" width="64"> -->
</div>
```

嵌套

媒体对象可以无限嵌套，尽管我们建议您在某些时候尽量减少网页的嵌套层级，但它在技术原理上来说是合法的，嵌套.media 在.media-body 下面即可:

```
<div class="media">
    <img src="images/sm.jpg" alt="" class="mr-3" width="64">
    <div class="media-body">
        <h5 class="mt-0">标题</h5>
        内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！

        <div class="media mt-3">
            <img src="images/sm.jpg" alt="" class="mr-3" width="64">
            <div class="media-body">
                <h5 class="mt-0">标题</h5>
                内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！内容！
            </div>
        </div>
    </div>
</div>
```

对齐

媒体对象中的媒体可以与 flexbox 流式布局一样，实现对齐到顶部、中间、底部，自由便利，只要在.media-body 的父级加上 img.align-self-start/center/end 等属性。

列表呈现

媒体对象的结构要求很少，您还可以在列表 HTML 元素上使用这些类，在 or 添加.list-unstyled 从而删除浏览器默认列表样式，然后在 li 中添加.media 元素块，也可以根据自己的需要进行间距调整。

```
<ul class="list-unstyled">
    <li class="media">
        <img src="images/sm.jpg" alt="" class="mr-3" width="64">
```

```
        <div class="media-body">
            <h5 class="mt-0">标题</h5>
            内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容!
内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容!
        </div>
    </li>
    <li class="media my-4">
        <img src="images/sm.jpg" alt="" class="mr-3" width="64">
        <div class="media-body">
            <h5 class="mt-0">标题</h5>
            内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容!
内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容!
        </div>
    </li>
    <li class="media">
        <img src="images/sm.jpg" alt="" class="mr-3" width="64">
        <div class="media-body">
            <h5 class="mt-0">标题</h5>
            内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容!
内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容! 内容!
        </div>
    </li>
</ul>
```

弹出提示框(Toast)

Toasts 出于性能原因选择加入,因此 你必须自己初始化它们。

```
<div class="toast hide" id="myToast" data-delay="500" data-animation="true/false"
data-autohide="true/false" style="position:fixed;top:0;right:0;">
    <div class="toast-header">
        <img src="images/logo_sm.png" alt="" width="20" class="rounded mr-2">
        <strong class="mr-auto">标题</strong>
        <small>1 分钟前</small>
        <button class="close ml-2 mb-1" data-dismiss="toast">&times;</button>
    </div>
    <div class="toast-body">
        您好! 这是一个 Toasts 弹出提示框.
    </div>
</div>

<script>
```

```
    $('#btn1').click(function(){
        $('#myToast').toast('show');
    });
</script>
```

JavaScript 行为

选项

选项可以通过数据属性或 JavaScript 传递。对于数据属性，请将选项名称附加到 data-，如在 data-animation=""。

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the toast
autohide	boolean	true	Auto hide the toast
delay	number	500	Delay hiding the toast (ms)

方法

(1) .toast(options)：将 toast 处理附加到元素集合。
(2) .toast('show')：揭示元素的提示框。**在实际显示提示框之前返回调用者**(i.e. 之前 shown.bs.toast 事件发生). 你必须手动调用此方法，代替你的提示框不会显示。
(3) .toast('hide')：隐藏元素的提示框。**在实际隐藏 toast 之前返回调用者**(i.e. 之前 hidden.bs.toast 事件发生). 如果你做了，你必须手动调用此方法 autohide 为 false.

事件

(1) show.bs.toast
(2) shown.bs.toast
(3) hide.bs.toast
(4) hidden.bs.toast

提示冒泡(Tooltip)

示例

在网页上初始化所有的 tooltip 提示冒泡插件一个途径就是用 data-toggle="tooltip" 来选择它们:

```
<button class="btn btn-primary" data-toggle="tooltip" title="标题内容 123！">button</button>

$(function () {
    $('[data-toggle="tooltip"]').tooltip()
})
```

方向：添加 data-placement="top/right/bottom/left"设置工具提示方向:顶部、右侧、底部和左侧。

```
<button class="btn btn-primary" data-toggle="tooltip" data-placement="top" title="标题内容 123！">button</button>
```

自定义 HTML：添加 data-html="true" 。

```
<button class="btn btn-primary" data-toggle="tooltip" data-html="true" title="<em>标题</em>内容 123！">button</button>
```

禁用元素

具有 disabled disabled 属性的元素不是交互式的，这意味着用户不能集中注意力,、悬停或单击它们来触发工具提示(或弹窗).作为一种解决方案，你将希望从包装器 或 ,触发工具提示，最好使用 tabindex="0"，并覆盖 pointer-events 禁用元素.

```
<span class="d-inline-block" tabindex="0" data-toggle="tooltip" title="标题内容 123！">
    <button class="btn btn-primary" style="pointer-events: none;" disabled>button</button>
</span>
```

JavaScript 行为

选项

名称	Type 类型	默认值	描述
animation	boolean	true	将 CSS 淡入淡出应用于 tooltip 提示冒泡。
delay	number | object	0	显示和隐藏弹出提示框的延迟(ms)-不适用于手动触发类型。 如果向这个选项提供一个数字，隐藏/显示都会应用这个延迟。 对象结构是：delay: { "show": 500, "hide": 100 }
trigger	string	'hover focus'	如何触发提示冒泡 - click | hover | focus | manual.，你可以传递多个触发器，用空格隔开它们，但是`manual`不能与别的触发器结合使用。
offset	number | string	0	提示冒泡框对于其目标的偏移

方法

(1) .tooltip(options)：将一个元素附加一个提示冒泡处理程序。
(2) .tooltip('show')：显示一个元素的提示冒泡， 在提示冒泡真正显示之前返回给调用者（即 shown.bs.tooltip 事件发生前)。这将被识别为一个“人为”的手动触发提示冒泡，零长度的提示框不会显示。
(3) .tooltip('hide')：隐藏元素的冒泡提示，在提示冒泡框实际被隐藏之前返回给调用者（即 hidden.bs.tooltip 事件发生前)。这将被识别为一个“人为”的手动触发提示冒泡。
(4) .tooltip('toggle')：切换元素的工具提示冒泡，在提示冒泡真正显示或隐藏之前返回给调用者（即 shown.bs.tooltip 或 hidden.bs.tooltip 事件发生前)。这将被识别为一个“人为”的手动触发提示冒泡。

事件

Event 事件类型	描述
show.bs.tooltip	当调用 show 实例方法时，会立即触发该事件。
shown.bs.tooltip	当提示冒泡对用户来说可见时（需要等待 CSS 过渡完成），会触

	发该事件。
hide.bs.tooltip	当调用 hide 实例方法时，会立即触发该事件。
hidden.bs.tooltip	当提示冒泡对用户来说终于完成隐藏时（需要等待 CSS 过渡完成），会触发该事件。
inserted.bs.tooltip	将提示冒泡框加到 DOM 后，会在 show.bs.tooltip 事件后触发此事件。

POP 提示(Popover)

示例

在网页上初始化所有的 Popover 提示冒泡插件一个途径就是用 data-toggle="popover" 来选择它们:

```
<button class="btn btn-primary" data-toggle="popover" title="标题！" data-content="内容内容内容。">button</button>

$(function () {
    $('[data-toggle="popover"]').popover()
})
```

方向：添加 data-placement=" right/top/bottom/left"设置工具提示方向：右侧、顶部、底部和左侧。

```
<button class="btn btn-primary" data-toggle="popover" title="标题！" data-content="内容内容内容。" data-placement="right">button</button>
```

自定义 HTML：添加 data-html="true" 。

```
<button class="btn btn-primary" data-toggle="popover" title="<em>标题！</em>" data-content="<b>内容</b>内容内容。" data-html="true">button</button>
```

禁用元素

具有 disabled 属性的元素不是交互式的这意味着用户不能悬停或点击它们来触发弹出窗口（或工具提示）。 作为一种解决方法，您需要从包装器 or 中触发弹出窗口，并覆盖 disabled 元素上的指针事件。
对于禁用的弹出式触发器，您也可能更喜欢 data-trigger="hover" ，以便弹出窗口显示为用

户的直接视觉反馈，因为他们可能不希望单击禁用的元素。

```
<span class="d-inline-block" tabindex="0" data-toggle="tooltip" title="标题内容 123！ ">
    <button class="btn btn-primary" style="pointer-events: none;" disabled>button</button>
</span>
```

在下次点击时收回

使用 focus 触发器，达到提示必须在用户下一次点击时才能收回（移除）提示事件。
为正确处理（兼容）跨浏览器和跨平台行为，你必须使用则必须使用<a> 标签，而 不是标签，你还必须包括一个 tabindex 属性

```
<a tabindex="0" class="btn btn-primary" data-toggle="popover" title="标题！ " data-content="内容内容内容。" data-trigger="focus">button</a>
```

JavaScript 行为

选项

名称	Type 类型	默认值	描述
animation	boolean	true	将 CSS 淡入淡出应用于 POP 提示。
delay	number | object	0	显示和隐藏弹出提示框的延迟(ms)-不适用于手动触发类型。 如果向这个选项提供一个数字，隐藏/显示都会应用这个延迟。 对象结构是：delay: { "show": 500, "hide": 100 }
trigger	string	'click'	如何触发 POP 提示 - click | hover | focus | manual.，你可以传递多个触发器，用空格隔开它们，但是`manual`不能与别的触发器结合使用。
offset	number | string	0	POP 提示框对于其目标的偏移

方法

(1) .popover(options)：将一个元素附加一个提示冒泡处理程序。

(2) .popover('show')：显示一个元素的提示冒泡， 在提示冒泡真正显示之前返回给调用者(即 shown.bs.tooltip 事件发生前)。这将被识别为一个“人为”的手动触发提示冒泡，零长度的提示框不会显示。
(3) .popover('hide')：隐藏元素的冒泡提示，在提示冒泡框实际被隐藏之前返回给调用者（即 hidden.bs.tooltip 事件发生前)。这将被识别为一个“人为”的手动触发提示冒泡。
(4) .popover('toggle')：切换元素的工具提示冒泡，在提示冒泡真正显示或隐藏之前返回给调用者（即 shown.bs.tooltip 或 hidden.bs.tooltip 事件发生前)。这将被识别为一个“人为”的手动触发提示冒泡。

事件

Event 事件类型	描述
show.bs.popover	当调用 show 实例方法时，会立即触发该事件。
shown.bs.popover	当提示冒泡对用户来说可见时（需要等待 CSS 过渡完成），会触发该事件。
hide.bs.popover	当调用 hide 实例方法时，会立即触发该事件。
hidden.bs.popover	当提示冒泡对用户来说终于完成隐藏时（需要等待 CSS 过渡完成），会触发该事件。
inserted.bs.popover	将提示冒泡框加到 DOM 后，会在 show.bs.popover 事件后触发此事件。

弹出模态框(Modal)

示例

(1) 按 ESC 关闭：添加 tabindex="-1"。
(2) **动画效果**：如果弹出模态需要淡入淡出效果，请从你的模态窗元素中添加 .fade 即可。
(3) **滚动长内容**：当用户 viewport 视口（弹出内容区）或设备的模态变得较长时，它们会自动滚动页面。
(4) **居中显示**：将 .modal-dialog-centered 添加到 .modal-dialog 对话框以垂直居中模式。
(5) **尺寸大小**：模态框有两个可选大小，可以通过 class 定义.modal-dialog 添加.modal-xl/lg/sm，这些尺寸会在某些中断点调整，以避免在较小的 viewport 窗口上出现水平滚动条。

```
<div class="modal fade" tabindex="-1" id="myModal">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
```

```
                    <h5 class="modal-title">标题！ </h5>
                    <button class="close" data-dismiss="modal">&times;</button>
                </div>
                <div class="modal-body">
                    <p>内容内容内容内容内容内容。</p>
                </div>
                <div class="modal-footer">
                    <button  class="btn  btn-secondary"  data-dismiss="modal"> 取 消
</button>
                    <button class="btn btn-primary">确定</button>
                </div>
            </div>
        </div>
</div>


<button class="btn btn-primary" data-toggle="modal" data-
target="#myModal">Modal</button>
```

工具提示和弹出提示框

Tooltips 工具提示 和 popovers 提示框，可以根据需要放置在模态框中。当模态框关闭时，其包含的任何提示和 Pop 提示都会同步关闭。

```
<div class="modal-body">
    <h5>Tooltip 工具提示</h5>
    <p>这个是 <a href="#" data-toggle="tooltip" title="标题内容 123！ ">Tooltip</a>
工具提示。</p>
    <hr>
    <h5>Popover 弹出提示框</h5>
    <p>这个是 <button class="btn btn-success" data-toggle="popover" title="标题！ "
data-content="内容内容内容。 ">Popover</button> 弹出提示框</p>
</div>
```

使用栅格

在.modal-body 中加入 .container-fluid 栅格系统，可以在动态视窗中使用 Bootsrap 栅格系统，并在任何地方使用正常的栅格系统 class 定义。

```
<div class="modal-body">
    <div class="container-fluid">
```

```
            <div class="row">
                <div class="col-sm-4">aaa</div>
                <div class="col-sm-8">bbb</div>
            </div>
        </div>
</div>
```

JavaScript 行为

选项

名称	Type 类型	默认值	描述
backdrop	boolean or the string 'static'	true	包括动态视窗背景元素，或者指定 static 在点击背景时不关闭动态模态框。
keyboard	boolean	true	按下 esc 时关闭动态视窗。

方法

(1) .modal(options)：激活您的内容作为模态，将选项加入到 object 内。
(2) .modal('show')：手动打开动态模态框，在动态模态框实际显示之前返回给调用者（即在 shown.bs.modal 事件发生前)。
(3) .modal('hide')：手动隐藏动态模态框，在动态模态框实际隐藏之前返回给调用者（即在 hidden.bs.modal 事件发生前)。
(4) .modal('toggle')：手动切换动态模态框，在动态模态框实际显示或隐藏之前返回给调用者（即在 shown.bs.modal 或 hidden.bs.modal 事件发生之前)。

事件

Event 事件类型	描述
show.bs.modal	当调用 show 实例方法时，会立即触发该事件。如果是由点击引起的，被点击的元素是可用的，成为 Event 对象的 relatedTarget 属性。
shown.bs.modal	当模态框对用户来说可见时（需要等待 CSS 过渡完成），会触发该事件。如果是由点击引起的，被点击的元素是可用的，成为 Event

	对象的 relatedTarget 属性。
hide.bs.modal	当调用 hide 实例方法时，会立即触发该事件。
hidden.bs.modal	当模态框对用户来说终于完成隐藏时（需要等待 CSS 过渡完成），会触发该事件。

导航/滑动门(nav)

基本导航样式

Bootstrap 中提供的导航可共享通用标记和样式，从基础.nav 样式类到活动与禁用状态。交换 class 选择符以在每种样式之间切换。
基础.nav 组件采用 **Flexbox 弹性布局**构建，并为构建所有类型的导航组件提供了坚实的基础，包括一些风格覆盖（以及列表）、一些更大 pading 连接间隙和基本的禁用样式。
基础的 .nav 组件不包含任何的 .active 状态，以下范例包括该类别，主要是为了说明这个 class 不会触发任何特殊的样式。

```
<ul class="nav">
    <li class="nav-item">
        <a href="#" class="nav-link active">选项一</a>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link">选项二</a>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link">选项三</a>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link disabled">选项四</a>
    </li>
</ul>
```

.nav 的 class 可以使用在很多地方，所以你的标记可以非常灵活，比如使用在 列表，或者自定义一个 <nav>组件，因为 .nav 基于 display: flex 定义，导航链接的行为与导航项目相同，不需要额外的标记。

```
<nav class="nav">
    <a href="#" class="nav-link active">选项一</a>
    <a href="#" class="nav-link">选项二</a>
    <a href="#" class="nav-link">选项三</a>
    <a href="#" class="nav-link disabled">选项四</a>
```

```
</nav>
```

可用样式

水平对齐

使用 **flexbox 通用布局属性**更改导航的水平对齐方式，默认情况下，导航按左对齐，但你可以用 .justify-content-center/end 居中/右对齐。

垂直排列

使用.flex-column 通用样式更改 Flexa 弹性项目的方向来设计导航，如在特定的 viewport 屏幕下需要堆叠，可使用响应式定义（如 .flex-sm-column 样式）。

Tabs 标签

从上面了解的基本导航，并加入 .nav-tabs 以生成选项卡标签（滑动门，同时结合 tab JavaScript 插件来构建 tabs 滑动门。

胶囊式标签页

HTML 标记相同，但使用 .nav-pills 替代

填充和对齐

.nav 内容有两种扩展 class 属性， 使用.nav-fill 会将 .nav-item 按照比例分配空间。注意：这会占用所有的水平空间，但不是每个导航项目的宽度相同。

当使用<nav>基于导航时，请确保包含 .nav-item 在 A 链接上。

对于等宽元素，请使用.nav-justified。所有水平空间将被导航链接占用，但与上述.nav-fill 不同，每个导航项目将具有相同的宽度。
与 .nav-fill 的例子近似，使用基于 <nav>的导航，确保在链接上包含 .nav-item 子项定义。

使用 Flex 弹性布局

如果需要吊牌应式的导航变化，请使用一系列的 flexbox 弹性布局类别，这结通用类别能提供不同的弹性布局，下例中，我们的导及将会堆叠在最小的屏幕标准上，然后从小断点开始填充可用宽度的水平布局。

```
<nav class="nav nav-pills flex-column flex-sm-row">
    <a class="flex-sm-fill text-sm-center nav-link active" href="#">Active</a>
    <a class="flex-sm-fill text-sm-center nav-link" href="#">Link</a>
    <a class="flex-sm-fill text-sm-center nav-link" href="#">Link</a>
    <a class="flex-sm-fill text-sm-center nav-link disabled" href="#">Disabled</a>
</nav>
```

使用下拉菜单

Tabs 式和胶囊式都可用。

```
<ul class="nav nav-pills">
    <li class="nav-item">
        <a href="#" class="nav-link active">选项一</a>
    </li>
    <li class="nav-item dropdown">
        <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown">下拉菜单</a>
        <div class="dropdown-menu">
            <a href="#" class="dropdown-item">aaa</a>
            <a href="#" class="dropdown-item">bbb</a>
            <a href="#" class="dropdown-item">ccc</a>
        </div>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link">选项三</a>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link disabled">选项四</a>
    </li>
```

```
</ul>
```

滑动门

1.只要在远素上指定 data-toggle="tab" 或 data-toggle="pill" 即可启动选项卡或胶囊式导航，而无需编写任何 JavaScrip，并可在 .nav-tabs 或 .nav-pills 使用这些数据属性。
2.要使标签淡入淡出，请添加.fade 到每个. tab-pane，第一个选项卡窗格还必须定义.show 使默认初始内容可见。

```
<ul class="nav nav-tabs">
    <li class="nav-item">
        <a href="#tab1" class="nav-link active" data-toggle="tab">选项一</a>
    </li>
    <li class="nav-item">
        <a  href="#"  class="nav-link"  data-toggle="tab"  data-target="#tab2">选项二
</a>
    </li>
    <li class="nav-item">
        <a href="#tab3" class="nav-link" data-toggle="tab">选项三</a>
    </li>
    <li class="nav-item">
        <a href="#tab4" class="nav-link disabled" data-toggle="tab">选项四</a>
    </li>
</ul>
<div class="tab-content">
    <div class="tab-pane fade show active" id="tab1">111</div>
    <div class="tab-pane fade" id="tab2">222</div>
    <div class="tab-pane fade" id="tab3">333</div>
    <div class="tab-pane fade" id="tab4">444</div>
</div>
```

还可以做成垂直形式:

```
<div class="row">
    <div class="col-3">
        <div class="nav nav-pills flex-column">
            <a href="#tab1" class="nav-link active" data-toggle="tab">选项一</a>
            <a href="#tab2" class="nav-link" data-toggle="tab">选项二</a>
            <a href="#tab3" class="nav-link" data-toggle="tab">选项三</a>
            <a href="#tab4" class="nav-link" data-toggle="tab">选项四</a>
        </div>
    </div>
```

```
    <div class="col-9">
        <div class="tab-content">
            <div class="tab-pane fade show active" id="tab1">111</div>
            <div class="tab-pane fade" id="tab2">222</div>
            <div class="tab-pane fade" id="tab3">333</div>
            <div class="tab-pane fade" id="tab4">444</div>
        </div>
    </div>
</div>
```

JavaScript 行为

事件

事件类型	描述
show.bs.tab	这个事件在一个标签选项卡内容显示上触发，但在选项卡显示之前，使用 event.target 和 event.relatedTarget 来分别定位选项卡和上一个选项卡标签（如果可用）。
shown.bs.tab	这个事件在一个标签选项卡显示之后触发，使用 event.target 和 event.relatedTarget 来分别定位选项卡和上一个启用的选项卡标签（如果可用）。
hide.bs.tab	在显示新标签选项卡之后触发此事件（因此以前的活动选项卡标签被隐藏），使用 event.target 和 event.relatedTarget 来分别定位前一个使用的选项卡和新的即将启用的选项卡（如果有）。
hidden.bs.tab	在显示新标签选项卡之后触发此事件（因此以前的活动选项卡标签被隐藏），使用 event.target 和 event.relatedTarget 来分别定位前一个使用的选项卡和新启用的选项卡。

```
$('a[data-toggle="tab"]').on('shown.bs.tab', function (e) {
    console.log(e.target)       // newly activated tab
    console.log(e.relatedTarget)        // previous active tab
})
```

导航栏(navbar)

支持的内容

Navbar 导航栏内置支持少量子组件。根据需要从以下选择:

- .navbar-brand 为您的公司，产品或项目名称。
- .navbar-nav 提供完整的高和轻便的导航（包括对下拉菜单的支持）。
- .navbar-toggler 用於我們的折疊插件和其他 navigation toggling 行為。
- .form-inline 用于任何表单控件和操作。
- .navbar-text 用于添加垂直居中的文本字符串。
- .collapse.navbar-collapse 用于通过父断点进行分组和隐藏导航列内容。

品牌

.navbar-brand 可以用于大多数元素，但对于链接最有效，因为某些元素可能需要通用样式类别 class 或自定义样式。

```
<nav class="navbar navbar-light bg-light">
    <a href="#" class="navbar-brand">
        <!-- <img src="images/logo.png" alt="" width="30" height="30" class="d-inline-block align-top"> -->
        网战天下
    </a>
    <!-- <span class="navbar-brand h1 mb-0">网战天下</span> -->
</nav>
```

nav 导航

1. 导航栏链接建立在我们的.nav 上，享有使用专属的 class 样式，并可以使用 toggler 切换触发器来进行响应式切换，在导航栏中的元件，也装饰占用更多的水平空间，以保持导览列内容安全对齐。
2. 活动状态指示：用 .active 表示当前页面，可直接应用于.nav-link 或 .nav-item 上。
3. When the container is within your navbar, its horizontal padding is removed at breakpoints lower than your specified .navbar-expand{-sm|-md|-lg|-xl} class. This ensures we're not doubling up on padding unnecessarily on lower viewports when your navbar is collapsed.
4. 右侧有一个 LOGO(主标题)，左边是 MENU 切换按钮:

```
<nav class="navbar navbar-expand-sm navbar-light bg-light">
    <a href="#" class="navbar-brand">网战天下</a>
    <button class="navbar-toggler" data-toggle="collapse" data-target="#navbarNav">
```

```
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a href="#" class="nav-link active">首页</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">产品</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">最新资讯</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link disabled">公司简介</a>
            </li>
        </ul>
    </div>
</nav>
```

如果你喜欢（或有需要），也可以不使用 ul、ol 式的列（直接用 A 其它元素作为列表子项)。

```
<div class="collapse navbar-collapse" id="navbarNav">
    <div class="navbar-nav">
        <a href="#" class="nav-item nav-link active">首页</a>
        <a href="#" class="nav-item nav-link">产品</a>
        <a href="#" class="nav-item nav-link">最新资讯</a>
        <a href="#" class="nav-item nav-link disabled">公司简介</a>
    </div>
</div>
```

您还可以在导航栏中使用下拉列表，下拉菜单最好使用一个菜单进行位置定位包裹，请确保使用单独的元素嵌套.nav-item 和 .nav-link，如下所示。

```
<li class="nav-item dropdown">
    <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown">下拉菜单
</a>
    <div class="dropdown-menu">
        <a href="#" class="dropdown-item">aaa</a>
        <a href="#" class="dropdown-item">bbb</a>
        <a href="#" class="dropdown-item">ccc</a>
    </div>
</li>
```

Form 表单

在导航栏中使用 .form-inline 放置各种表单控制元件和组件。

```
<nav class="navbar navbar-expand-md navbar-light bg-light">
    <a href="#" class="navbar-brand">网战天下</a>
    <button class="navbar-toggler" data-toggle="collapse" data-target="#navbarNav">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a href="#" class="nav-link active">首页</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">产品</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">最新资讯</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link disabled">公司简介</a>
            </li>
        </ul>
        <form action="" class="form-inline my-2 my-md-0">
            <input type="text" class="form-control mr-sm-2" placeholder="请输入关
键字">
            <button class="btn btn-outline-success my-2 my-md-0">搜索</button>
        </form>
    </div>
</nav>
```

Text 文本处理

可以使用.navbar-text 选择器来包裹文字-这已经对文本字符串的垂直对齐、水平间距作了处理优化。

```
<span class="navbar-text">文本内容</span>
```

Color 颜色选择器(配色方案)

基于主题类 class 和 background-color 通用样式 class 定义，导航栏的配色方案和主题选择从未如此简单！你可以选择.navbar-light 来定义导航颜色反转（强黑白对比），也可以使用.navbar-dark 用于深色背景定义，然后再引用.bg-*类通用定义来进行大小处理。

```
<nav class="navbar navbar-dark bg-dark">
    <!-- Navbar content -->
</nav>

<nav class="navbar navbar-dark bg-primary">
    <!-- Navbar content -->
</nav>

<nav class="navbar navbar-light" style="background-color: #e3f2fd;">
    <!-- Navbar content -->
</nav>
```

.Container 主内容-容器

你可以把导航条包裹在一个 .container 容器中，从而使之在网页中呈现居中效果（或在导航栏内部居中）--虽然这不是强制的。

定位

使用系统提供的 position 位置间距定位通用样式可以使导航栏呈现出随浏览器滚动的效果（非固定位置），可选的流动可以包括固定在顶部、固定在底部、或粘到顶部（与页面滚动，直到顶部并停留到那里）。固定导航栏可以使用 position: fixed 属性，这意味着它们从 DOM 的正常流动和拉动可能需要自定义的 CSS(如在 <body>上定义 padding-top)，以防止其重叠覆盖了其它元素。
注意：在.sticky-top 使用 position: sticky，目前不支持所有常用浏览器。

固定在顶部：.fixed-top

固定在底部：.fixed-bottom

呈现粘性(随屏滚动)于浏览器窗口顶部：.sticky-top

卡片(Card)

如果你对 Bootstrap 3 很熟悉，卡片代替了我们旧的 panel、well 和 thumbnail 样式--那些组件类似的功能可以通过卡片的修饰类来实现。

内容类型

主体

引用.card-body 样式，可以建立起卡片的内容主体，如果你需要在一个.card 中装置带边框的内容时，可以使用它。

```
<div class="card">
    <div class="card-body">
        主体主体主体主体主体主体主体主体主体主体主体主体。
    </div>
</div>
```

标题、文字和链接

通过.card-title 和 <h*>组合，可以添加卡片标题。同亲将.card-link 与 <a>结合使用，可以方便添加平行的链接。
通过 .card-subtitle 和 <h*> 结合，可以添加副标题，如果 .card-title 和 .card-subtitle 组合放在 .card-body 中，则可对齐主、副标题。

```
<div class="card">
    <div class="card-body">
        <h5 class="card-title">标题！ </h5>
        <h6 class="card-subtitle mb-2 text-muted">子标题！ </h6>
        <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。
</p>
        <!-- <a href="" class="btn btn-primary">查看详情</a> -->
        <a href="" class="card-link">立即下载</a>
        <a href="" class="card-link">了解更多</a>
    </div>
</div>
```

图片

.card-img-top 定义一张图片在卡片的顶部， .card-text 定义文字在卡片中，当然你也可以在.card-text 中设计自己的个性化 HTML 标签样式。

```
<div class="card" style="width:18rem;">
    <img src="images/sm.jpg" alt="" class="card-img-top">
    <div class="card-body">
        <p class="card-text">主体主体主体主体主体主体主体主体主体主体主体主体主体。
</p>
    </div>
    <!-- <img src="images/sm.jpg" alt="" class="card-img-bottom"> -->
</div>
```

图像叠加覆盖:
将图像转换为卡片背景，并覆盖卡片的文字。根据图像，你可以选择是否需要额外的样式或其它属性定义。

```
<div class="card" style="width:18rem;">
    <img src="images/sm.jpg" alt="" class="card-img-top">
    <div class="card-img-overlay">
        <h5 class="card-title">标题！ </h5>
        <p class="card-text">主体主体主体主体主体主体主体主体主体主体主体主体主体。
</p>
    </div>
</div>
```

列表组

建立一个包含内容的列表组卡片。

```
<div class="card" style="width:18rem;">
    <!-- <div class="card-header">Title</div> -->
    <ul class="list-group list-group-flush">
        <li class="list-group-item">aaaaaa</li>
        <li class="list-group-item">bbbbbb</li>
        <li class="list-group-item">cccccc</li>
    </ul>
</div>
```

混合样式

混合并结合多种内容形式来创建个性化卡片，下例即是将图像、块、文字以及列表整合在一个固定宽度的卡片中。

```
<div class="card" style="width:18rem;">
    <img src="images/sm.jpg" alt="" class="card-img-top">
    <div class="card-body">
        <h5 class="card-title">标题！ </h5>
        <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。
</p>
    </div>
    <ul class="list-group list-group-flush">
        <li class="list-group-item">列表项一列表项一</li>
        <li class="list-group-item">列表项二列表项二</li>
        <li class="list-group-item">列表项三列表项三</li>
    </ul>
    <div class="card-body">
        <a href="" class="card-link">立即下载</a>
        <a href="" class="card-link">了解更多</a>
    </div>
</div>
```

页眉页脚

在卡内添加可选的页眉和/或页脚。
您还可以更改卡上的页眉和页脚所需的边框，也能使用.bg-transparent 删除其 background-color 背景颜色。

```
<div class="card text-center">
    <div class="card-header">页眉</div>
    <div class="card-body">
        <h5 class="card-title">标题！ </h5>
        <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述。</p>
        <a href="" class="btn btn-primary">查看详情</a>
    </div>
    <div class="card-footer text-muted">
        页脚
    </div>
```

```
</div>
```

使用 blockquote：

```
<div class="card">
    <div class="card-header">页眉</div>
    <div class="card-body">
        <blockquote class="blockquote mb-0">
            <p>爱上一个地方，就应该背上包去旅游，走得更远。</p>
            <footer class="blockquote-footer">出自商务印书馆的 <cite title="Source Title">《新华字典》</cite></footer>
        </blockquote>
    </div>
</div>
```

缩放_文本对齐

缩放

卡片没有特定的宽度 width 定义，除非另有定义声明，否则它们的真实宽度将是 100%，您可以根据需要使用自定义 CSS，引入栅格系统、定义栅格系统 Sass mixins 或其它程式进行更改。

(1) 使用栅格系统

(2) 使用通用全局属性。如 .w-25 等。

(3) 自定义 CSS。如 style="width:18rem;"等。

文本对齐

使用我们的**文本对齐类**(.text-left/center/right)来更改或特定部份的文本对齐。

导航

使用 Bootstrap 导航组件将导航元件添加到卡片的标题或块中

```
<div class="card text-center">
    <div class="card-header">
        <ul class="nav nav-tabs card-header-tabs">
```

```
            <!-- <ul class="nav nav-pills card-header-pills"> -->
                <li class="nav-item">
                    <a href="" class="nav-link active">选项一</a>
                </li>
                <li class="nav-item">
                    <a href="" class="nav-link">选项二</a>
                </li>
                <li class="nav-item">
                    <a href="" class="nav-link">选项三</a>
                </li>
            </ul>
        </div>
        <div class="card-body">
            <h5 class="card-title">标题！ </h5>
            <p class="card-text">
                描述描述描述描述描述描述描述描述描述描述描述描述描述描述。
            </p>
            <a href="" class="btn btn-primary">查看详情</a>
        </div>
    </div>
```

卡片样式

背景和颜色

使用 文字和通用背景定义 定义卡片的显示颜色。

```
<div class="row">
        <div class="col-3">
            <div class="card text-white bg-primary mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card text-white bg-secondary mb-3">
                <div class="card-header">Header</div>
```

```
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card text-white bg-success mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card text-white bg-danger mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card text-white bg-warning mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card text-white bg-info mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
```

```
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card bg-light mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card text-white bg-dark mb-3">
                <div class="card-header">Header</div>
                <div class="card-body">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
                </div>
            </div>
        </div>
</div>
```

边框

使用 **边框通用样式** 来改变卡片的 border-color 、 .text-{color} ， 或者在父层的 .card 上显示内容。

```
<div class="row">
        <div class="col-3">
```

```
            <div class="card border-primary mb-3">
                <div class="card-header">Header</div>
                <div class="card-body text-primary">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card border-secondary mb-3">
                <div class="card-header">Header</div>
                <div class="card-body text-secondary">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card border-success mb-3">
                <div class="card-header">Header</div>
                <div class="card-body text-success">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card border-danger mb-3">
                <div class="card-header">Header</div>
                <div class="card-body text-danger">
                    <h5 class="card-title">标题！ </h5>
                    <p class="card-text">描述描述描述描述描述描述描述描述描述描述描
述描述。</p>
                </div>
            </div>
        </div>
        <div class="col-3">
            <div class="card border-warning mb-3">
                <div class="card-header">Header</div>
                <div class="card-body text-warning">
                    <h5 class="card-title">标题！ </h5>
```

```
                <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
            </div>
        </div>
    </div>
    <div class="col-3">
        <div class="card border-info mb-3">
            <div class="card-header">Header</div>
            <div class="card-body text-info">
                <h5 class="card-title">标题！ </h5>
                <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
            </div>
        </div>
    </div>
    <div class="col-3">
        <div class="card border-light mb-3">
            <div class="card-header">Header</div>
            <div class="card-body">
                <h5 class="card-title">标题！ </h5>
                <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
            </div>
        </div>
    </div>
    <div class="col-3">
        <div class="card border-dark mb-3">
            <div class="card-header">Header</div>
            <div class="card-body text-dark">
                <h5 class="card-title">标题！ </h5>
                <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
            </div>
        </div>
    </div>
</div>
```

卡片排版

Bootstrap 除了对卡片内的内容可以进行设计排版外，还包括一系列布置选项，目前这些布置选项还**不支持响应式**。

卡片组

.card-group 将多个卡片结为一个群组，使用他们具有相同的宽度和高度列。卡片组使用 display: flex;来实现统一的布局。

```
<div class="card-group">
    <div class="card">
        <img src="images/sm.jpg" alt="" class="card-img-top">
        <div class="card-body">
            <h5 class="card-title">标题！ </h5>
            <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
        </div>
    </div>
    <div class="card">
        <img src="images/sm.jpg" alt="" class="card-img-top">
        <div class="card-body">
            <h5 class="card-title">标题！ </h5>
            <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
        </div>
    </div>
    <div class="card">
        <img src="images/sm.jpg" alt="" class="card-img-top">
        <div class="card-body">
            <h5 class="card-title">标题！ </h5>
```

```
                <p class="card-text">描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述描述。</p>
            </div>
        </div>
</div>
```

Card decks 卡片阵列

需要一套相互不相连，但宽度和高度相同的卡片？使用卡片阵列.card-deck 吧。

多列卡片浮动排版

将卡片包在.card-columns 中，可以将做出象 Masonry 网站的瀑布式排列卡片效果，卡片是使用 column 属性，而不是基于 flexbox 弹性布局，从而实现更方便实用的浮动对齐，顺序是从上到下、从左到右。
注意：为了防止卡片排列突出栏目，我们必须为它们设置为 display: inline-block （当 column-break-inside: avoid 这个解决方案还没有生效时。

折叠面板(Collapse)

示例

点击下面任何一个按钮，通过类更改显示和隐藏另一个 class 包含的元素：

- .collapse 隐藏内容
- .collapsing 带动态效果的切换
- .collapse.show 显示内容

你可以使用带 href 属性的连接,、或者带 data-target 属性的按钮来创建折叠效果-这两种情况下 data-toggle="collapse" 属性都是必须的。

```
<p>
    <a href="#myCollapse" class="btn btn-primary" data-toggle="collapse">btn1</a>
    <button class="btn btn-primary" data-toggle="collapse" data-target="#myCollapse">btn2</button>
</p>
<div class="collapse" id="myCollapse">
    <div class="card card-body">
```

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

多目标控制

可以在<button>或者 <a> 标签上，通过 JQuery 选择器来显示和隐藏多个元素（或者多个<button>、 <a>元素来控制显示/隐藏一个元素素)），如果被引用对象的 href 或者 data-target 属性定义正确。

```
<p>
    <a href=".myCol" class="btn btn-primary" data-toggle="collapse">btn1</a>
    <button class="btn btn-primary" data-toggle="collapse" data-target=".myCol">btn2</button>
</p>
<div class="row">
    <div class="col">
        <div class="collapse myCol">
            <div class="card card-body">
                aaaaaa
            </div>
        </div>
    </div>
    <div class="col">
        <div class="collapse myCol">
            <div class="card card-body">
                bbbbbb
            </div>
        </div>
    </div>
</div>
```

手风琴折叠范例

结合 card 卡片组件使用，可以扩展折叠组件为手风琴效果。

```
<div id="accordion">
    <div class="card">
        <div class="card-header">
            <h5 class="mb-0">
                <button class="btn btn-link" data-toggle="collapse" data-
```

```
target="#collapseOne">选项一</button>
                </h5>
            </div>
            <div class="collapse show" id="collapseOne" data-parent="#accordion">
                <div class="card-body">
                    aaaaaa
                </div>
            </div>
        </div>

        <div class="card">
            <div class="card-header">
                <h5 class="mb-0">
                    <button class="btn btn-link" data-toggle="collapse" data-
target="#collapseTwo">选项二</button>
                </h5>
            </div>
            <div class="collapse" id="collapseTwo" data-parent="#accordion">
                <div class="card-body">
                    bbbbbb
                </div>
            </div>
        </div>

        <div class="card">
            <div class="card-header">
                <h5 class="mb-0">
                    <button class="btn btn-link" data-toggle="collapse" data-
target="#collapseThree">选项三</button>
                </h5>
            </div>
            <div class="collapse" id="collapseThree" data-parent="#accordion">
                <div class="card-body">
                    cccccc
                </div>
            </div>
        </div>
</div>
```

你也可以使用自定义样式创建手风琴效果，只要添加 data-children 属性，并指定一组相邻元素来切换(如.item)，然后使用与上述相同的属性和 class，来切换/隐藏其关联的内容。

```
<div id="accordion" data-children=".item">
    <div class="item">
```

```
        <a href="#collapseOne" data-toggle="collapse">选项一</a>
        <div class="collapse show" id="collapseOne" data-parent="#accordion">
            <p>
                aaaaaa
            </p>
        </div>
    </div>
    <div class="item">
        <a href="#collapseTwo" data-toggle="collapse">选项二</a>
        <div class="collapse" id="collapseTwo" data-parent="#accordion">
            <p>
                bbbbbb
            </p>
        </div>
    </div>
    <div class="item">
        <a href="#collapseThree" data-toggle="collapse">选项三</a>
        <div class="collapse" id="collapseThree" data-parent="#accordion">
            <p>
                cccccc
            </p>
        </div>
    </div>
</div>
```

JavaScript 行为

方法

(1) .collapse(options)：启用你的可折叠对象，通过 object 方法。
(2) .collapse('show')：显示可折叠元素，在可折叠元素实际显示之前（即 shown.bs.collapse 事件发生之前)返回给调用者。
(3) .collapse('hide')：隐藏可折叠元素，在可折叠元素实际上被隐藏之前（即 hidden.bs.collapse 事件发生之前)返回给调用者。
(4) .collapse('toggle')：即发生 shown.bs.collapse 或 hidden.bs.collapse 事件前)返回给调用者。

事件

事件类别	描述
show.bs.collapse	当调用 show 方法时，会立即触发事件。
shown.bs.collapse	用户可见折叠面板中的块时，会触发此事件（需要等 CSS 加载过渡完成）。
hide.bs.collapse	当调用 hide 方法时，立即触发该事件。
hidden.bs.collapse	当折叠面板中的块对于用户完全隐藏时（需要等 CSS 加载过渡完成），会触发该事件）。

分页(Pagination)

1. 使用图标：«，»
2. 禁用和活动状态：.disabled，.active

.disabled 使用 pointer-events: none 來禁用 <a>的链接功能，但该 CSS 属性尚未标准化（使用的时候要注意浏览器兼容性调式）。

为了安全起见，请在这些链接上添加一个 tabindex="-1"定义并使用自定义 JavaScript 来完全禁用其功能。

3. 规格尺寸：.pagination-lg/sm
4. 对齐：使用 **flexbox 弹性布局通用样式**，可用.justify-content-center/end 更改分页组件的对齐方式。

```
<nav>
    <ul class="pagination">
        <li class="page-item">
            <a href="" class="page-link">上一页</a>
        </li>
        <li class="page-item">
            <a href="" class="page-link">1</a>
        </li>
        <li class="page-item">
            <a href="" class="page-link">2</a>
        </li>
        <li class="page-item">
            <a href="" class="page-link">3</a>
        </li>
        <li class="page-item">
            <a href="" class="page-link">下一页</a>
        </li>
    </ul>
```

```
</nav>
```

进度条(Progress)

示例

1. 标签：在.progress-bar 中放置文字内容,可将标签添加到进度条中。
2. 高度：我们只在.progress 上设置了一个 height 值，所以如果你改变了这个值，那么内部的 .progress-bar 高度会自动调整大小。

```
<div class="progress">
    <div class="progress-bar" style="width:60%;">60%</div>
</div>
```

背景

使用背景通用样式 Class 样式来定义进度条的外观。

```
<div class="progress">
    <div class="progress-bar w-25">25%</div>
</div>
<br>
<div class="progress">
    <div class="progress-bar bg-success w-25">25%</div>
</div>
<br>
<div class="progress">
    <div class="progress-bar bg-info w-50">50%</div>
</div>
<br>
<div class="progress">
    <div class="progress-bar bg-warning w-75">75%</div>
</div>
<br>
<div class="progress">
    <div class="progress-bar bg-danger w-100">100%</div>
</div>
```

多进度条进度（嵌套）

```
<div class="progress">
    <div class="progress-bar" style="width:15%;"></div>
    <div class="progress-bar bg-success" style="width:30%;"></div>
    <div class="progress-bar bg-info" style="width:20%;"></div>
</div>
```

条纹进度指示

将.progress-bar-striped 添加到 **.progress-bar** 上，可以使用 CSS 渐变对背景颜色加上条纹效果。

```
<div class="progress">
    <div class="progress-bar progress-bar-striped w-25"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped bg-success w-25"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped bg-info w-50"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped bg-warning w-75"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped bg-danger w-100"></div>
</div>
```

动画条纹进度指示

条纹渐变也可以做成动画效果，将.progress-bar-animated 加到 **.progress-bar** 上，即实现 CSS3 绘制的从右到左的动画效果。
动画条纹进度条不适用于 Opera 12 浏览器—因为它不支持 CSS3 动画。

```
<div class="progress">
```

```
    <div class="progress-bar progress-bar-striped progress-bar-animated w-25"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped progress-bar-animated bg-success w-25"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped progress-bar-animated bg-info w-50"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped progress-bar-animated bg-warning w-75"></div>
</div>
<br>
<div class="progress">
    <div class="progress-bar progress-bar-striped progress-bar-animated bg-danger w-100"></div>
</div>
```

滚动监听(Scrollspy)

Scrollspy 滑动监控组件正常运行的几个要点:

- 如果从源代码构建 JS，请 包括 util.js。
- 它必须在 Bootstrap nav 导航组年 或 list group 列表组上使用。
- Scrollspy 需要在你监控的元素上使用 position: relative; ，通常是 <body>。
- 当需要对 <body>以外的元素进行监控时，请确保具有 height 高度值和 overflow-y: scroll; 属性。
- 锚点 (<a>)是必须的，并且必须指向一个 id 上。

实施成功后，你的导航或列表群组将相应地更新，根据 **.active** 关联的目标，从一个项目移到另一个项目。

通过数据属性：要轻松添加滚动行为到您的顶栏导航，添加 data-spy="scroll"到您要窥探的元素（通常是<body>）。然后添加 data-target 属性到任何 Bootstrap .nav 组件的父元素 ID 或类的 Class 属性。

在 navbar 导航中的示例

```
<nav class="navbar navbar-light bg-light">
    <a href="" class="navbar-brand">前端</a>
    <ul class="nav nav-pills">
        <li class="nav-item">
            <a href="#h5" class="nav-link">@HTML5</a>
        </li>
        <li class="nav-item">
            <a href="#c3" class="nav-link">@CSS3</a>
        </li>
        <li class="nav-item dropdown">
            <a href="" class="nav-link dropdown-toggle" data-toggle="dropdown">JavaScript</a>
            <div class="dropdown-menu">
                <a href="#vue" class="dropdown-item">Vue.js</a>
                <a href="#react" class="dropdown-item">React.js</a>
            </div>
        </li>
    </ul>
</nav>
<div style="position:relative;height:200px;overflow-y:scroll;" data-spy="scroll">
    <h4 id="h5">HTML5</h4>
    <p>
        标准通用标记语言下的一个应用 HTML 标准自 1999 年 12 月发布的 HTML4.01 后，后继的 HTML5 和其它标准被束之高阁，为了推动 Web 标准化运动的发展，一些公司联合起来，成立了一个叫做 Web Hypertext Application Technology Working Group（Web 超文本应用技术工作组 -WHATWG） 的组织。WHATWG 致力于 Web 表单和应用程序，而 W3C（World Wide Web Consortium，万维网联盟） 专注于 XHTML2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。
        <br>
        HTML5 草案的前身名为 Web Applications 1.0，于 2004 年被 WHATWG 提出，于 2007 年被 W3C 接纳，并成立了新的 HTML 工作团队。
        <br>
        HTML 5 的第一份正式草案已于 2008 年 1 月 22 日公布。HTML5 仍处于完善之中。然而，大部分现代浏览器已经具备了某些 HTML5 支持。
    </p>
    <p>
        2012 年 12 月 17 日，万维网联盟（W3C）正式宣布凝结了大量网络工作者心血的 HTML5 规范已经正式定稿。根据 W3C 的发言稿称：“HTML5 是开放的 Web 网络平台的奠基石。”
        <br>
        2013 年 5 月 6 日， HTML 5.1 正式草案公布。该规范定义了第五次重大版本，
```

第一次要修订万维网的核心语言：超文本标记语言（HTML）。在这个版本中，新功能不断推出，以帮助 Web 应用程序的作者，努力提高新元素互操作性。

本次草案的发布，从 2012 年 12 月 27 日至今，进行了多达近百项的修改，包括 HTML 和 XHTML 的标签，相关的 API、Canvas 等，同时 HTML5 的图像 img 标签及 svg 也进行了改进，性能得到进一步提升。

<h4 id="c3">CSS3</h4>

CSS3 是 CSS（层叠样式表）技术的升级版本，于 1999 年开始制订，2001 年 5 月 23 日 W3C 完成了 CSS3 的工作草案，主要包括盒子模型、列表模块、超链接方式、语言模块、背景和边框、文字特效、多栏布局等模块。

CSS 演进的一个主要变化就是 W3C 决定将 CSS3 分成一系列模块。浏览器厂商按 CSS 节奏快速创新，因此通过采用模块方法，CSS3 规范里的元素能以不同速度向前发展，因为不同的浏览器厂商只支持给定特性。但不同浏览器在不同时间支持不同特性，这也让跨浏览器开发变得复杂。

<h4 id="vue">Vue.js</h4>

Vue.js 是一套构建用户界面的渐进式框架。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，并且非常容易学习，非常容易与其它库或已有项目整合。另一方面，Vue 完全有能力驱动采用单文件组件和 Vue 生态系统支持的库开发的复杂单页应用。

Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

Vue.js 自身不是一个全能框架——它只聚焦于视图层。因此它非常容易学习，非常容易与其它库或已有项目整合。另一方面，在与相关工具和支持库一起使用时，Vue.js 也能完美地驱动复杂的单页应用。

<h4 id="react">React.js</h4>

由于 React 的设计思想极其独特，属于革命性创新，性能出众，代码逻辑却非常简单。所以，越来越多的人开始关注和使用，认为它可能是将来 Web 开发的主流工具。

这个项目本身也越滚越大，从最早的 UI 引擎变成了一整套前后端通吃的 Web App 解决方案。衍生的 React Native 项目，目标更是宏伟，希望用写 Web App 的方式

```
去写 Native App。如果能够实现，整个互联网行业都会被颠覆，因为同一组人只需要写一次 UI ，就能同时运行在服务器、浏览器和手机。
    </p>
    <p>
        React 主要用于构建 UI。你可以在 React 里传递多种类型的参数，如声明代码，帮助你渲染出 UI、也可以是静态的 HTML DOM 元素、也可以传递动态变量、甚至是可交互的应用组件。
    </p>
</div>
```

嵌套的导航示例

```
<div class="row">
    <div class="col-3">
        <nav class="navbar navbar-light bg-light flex-column">
            <a href="" class="navbar-brand">网战天下</a>
            <nav class="nav nav-pills flex-column">
                <a href="#item-1" class="nav-link">Item 1</a>
                <nav class="nav nav-pills flex-column">
                    <a href="#item-1-1" class="nav-link ml-3 my-1">Item 1-1</a>
                    <a href="#item-1-2" class="nav-link ml-3 my-1">Item 1-2</a>
                </nav>

                <a href="#item-2" class="nav-link">Item 2</a>

                <a href="#item-3" class="nav-link">Item 3</a>
                <nav class="nav nav-pills flex-column">
                    <a href="#item-3-1" class="nav-link ml-3 my-1">Item 3-1</a>
                    <a href="#item-3-2" class="nav-link ml-3 my-1">Item 3-2</a>
                </nav>
            </nav>
        </nav>
    </div>
    <div class="col-9">
        <div style="position:relative;height:350px;overflow-y:scroll;" data-spy="scroll">
            <h4 id="item-1">Item 1</h4>
            <p>
                Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>
            </p>
            <h5 id="item-1-1">Item 1-1</h5>
```

```
            <p>
                Item 1-1<br>Item 1-1<br>Item 1-1<br>Item 1-1<br>Item 1-1<br>Item 1-1<br>Item 1-1<br>Item 1-1<br>Item 1-1<br>Item
                1-1<br>Item 1-1<br>
            </p>
            <h5 id="item-1-2">Item 1-2</h5>
            <p>
                Item 1-2<br>Item 1-2<br>Item 1-2<br>Item 1-2<br>Item 1-2<br>Item 1-2<br>Item 1-2<br>Item 1-2<br>Item 1-2<br>Item
                1-2<br>Item 1-2<br>
            </p>

            <h4 id="item-2">Item 2</h4>
            <p>
                Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item
                2<br>Item 2<br>
            </p>

            <h4 id="item-3">Item 3</h4>
            <p>
                Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item
                3<br>Item 3<br>
            </p>
            <h5 id="item-3-1">Item 3-1</h5>
            <p>
                Item 3-1<br>Item 3-1<br>Item 3-1<br>Item 3-1<br>Item 3-1<br>Item 3-1<br>Item 3-1<br>Item 3-1<br>Item 3-1<br>Item
                3-1<br>Item 3-1<br>Item 3-1<br>
            </p>
            <h5 id="item-3-2">Item 3-2</h5>
            <p>
                Item 3-2<br>Item 3-2<br>Item 3-2<br>Item 3-2<br>Item 3-2<br>Item 3-2<br>Item 3-2<br>Item 3-2<br>Item 3-2<br>Item
                3-2<br>Item 3-2<br>Item 3-2<br>
            </p>
        </div>
    </div>
</div>
```

列表组示例

```
<div class="row">
    <div class="col-3">
        <div class="list-group">
            <a href="#item-1" class="list-group-item list-group-item-action">Item 1</a>
            <a href="#item-2" class="list-group-item list-group-item-action">Item 2</a>
            <a href="#item-3" class="list-group-item list-group-item-action">Item 3</a>
        </div>
    </div>
    <div class="col-9">
        <div style="position:relative;height:350px;overflow-y:scroll;" data-spy="scroll">
            <h4 id="item-1">Item 1</h4>
            <p>
                Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>Item 1<br>
            </p>

            <h4 id="item-2">Item 2</h4>
            <p>
                Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>Item 2<br>
            </p>

            <h4 id="item-3">Item 3</h4>
            <p>
                Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>Item 3<br>
            </p>
        </div>
    </div>
</div>
```

JavaScript 行为

选项

名称	Type 类型	默认值	描述
offset	number	10	计算滚动位置时，从顶部开始的计算的像偏移距离。

事件

Event 事件类型	描述
activate.bs.scrollspy	每当新项目被滚动激活时，该事件就会在滚动元素上触发。

```
$('[data-spy="scroll"]').on('activate.bs.scrollspy', function () {
    // do something…
})
```

旋转特效(Spinners)

圆形旋转

```
<div class="spinner-border"></div>
```

颜色:

```
<div class="spinner-border"></div>
<div class="spinner-border text-primary"></div>
<div class="spinner-border text-secondary"></div>
<div class="spinner-border text-success"></div>
<div class="spinner-border text-danger"></div>
<div class="spinner-border text-warning"></div>
<div class="spinner-border text-info"></div>
<div class="spinner-border text-light"></div>
<div class="spinner-border text-dark"></div>
```

渐变缩放

```
<div class="spinner-grow"></div>
```

颜色:

```
<div class="spinner-grow"></div>
<div class="spinner-grow text-primary"></div>
<div class="spinner-grow text-secondary"></div>
<div class="spinner-grow text-success"></div>
<div class="spinner-grow text-danger"></div>
<div class="spinner-grow text-warning"></div>
<div class="spinner-grow text-info"></div>
<div class="spinner-grow text-light"></div>
<div class="spinner-grow text-dark"></div>
```

对准

边距

用**边距**设置， 如 .m-5 更简单。

```
<div class="spinner-border m-5"></div>
```

位置

(1) 弹性

```
<div class="d-flex justify-content-center">
    <div class="spinner-border"></div>
</div>
```

```
<div class="d-flex align-items-center">
    <strong>Loading...</strong>
    <div class="spinner-border ml-auto"></div>
</div>
```

(2) 浮动

```
<div class="clearfix">
    <div class="spinner-border float-right"></div>
</div>
```

(3) 对齐方向

```
<div class="text-center">
    <div class="spinner-border"></div>
</div>
```

大小

加上 .spinner-border-sm 和 .spinner-grow-sm 为了制作一个更小的转轮，可以快速地在其他组件中使用。

```
<div class="spinner-border"></div>
<div class="spinner-border spinner-border-sm"></div>

<div class="spinner-grow"></div>
<div class="spinner-grow spinner-grow-sm"></div>
```

或者,根据需要使用自定义 CSS 或内联样式更改维度:

```
<div class="spinner-border"></div>
<div class="spinner-border" style="width:3rem;height:3rem;"></div>

<div class="spinner-grow"></div>
<div class="spinner-grow" style="width:3rem;height:3rem;">></div>
```

按钮类型

在按钮中使用旋转器指示当前正在处理或正在进行的操作。你还可以从 spinner 元素中交换文本，并根据需要使用按钮文本。

```
<button class="btn btn-primary" disabled>
  <span class="spinner-border spinner-border-sm"></span>
</button>
<button class="btn btn-primary" disabled>
  <span class="spinner-border spinner-border-sm"></span>
  Loading...
</button>

<button class="btn btn-primary" disabled>
  <span class="spinner-grow spinner-grow-sm"></span>
</button>
<button class="btn btn-primary" disabled>
  <span class="spinner-grow spinner-grow-sm"></span>
  Loading...
</button>
```

延伸(图标)

首选推荐

我们亲自测试并使用了这些图标库：

- Font Awesome
- Iconic
- Octicons

更多选择

虽然我们还没有使用下面这些图片库，但它们也提供了包括 SVG 在内的多种格式图标样式，呈现出强大的发展潜力：

- Bytesize
- Google Material icons
- Ionicons
- Feather
- Dripicons
- Ikons
- Glyph
- Icons8