JavaScript 插件

jQuery 插件为 Bootstrap 的组件赋予了“生命”。可以简单地一次性引入所有插件，或者逐个引入到你的页面中。

概览

单个还是全部引入

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

建议使用压缩版的 JavaScript 文件

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

组件的 data 属性

不要在同一个元素上同时使用多个插件的 data 属性。例如，button 组件不能同时支持工具提示和控制模态框。如果需要的话，可以在其外面包裹一个额外的元素。

插件之间的依赖关系

某些插件和 CSS 组件依赖于其它插件。如果你是单个引入每个插件的，请确保在文档中检查插件之间的依赖关系。注意，所有插件都依赖 jQuery （也就是说，jQuery必须在所有插件**之前**引入页面）。 `bower.json` (https://github.com/twbs/bootstrap/blob/v3.2.0/bower.json) 文件中列出了 Bootstrap 所支持的 jQuery 版本。

data 属性

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

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

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

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

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

编程方式的 API

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

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

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

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

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

默认设置

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

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

避免命名空间冲突

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

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

事件

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

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

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

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

未对禁用 JavaScript 的浏览器提供补救措施

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

第三方工具库

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

过渡效果 transition.js

关于过渡效果

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

包含的内容

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

模态框 modal.js

实例

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

不支持模态框重叠

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

模态框的 HTML 代码放置的位置

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

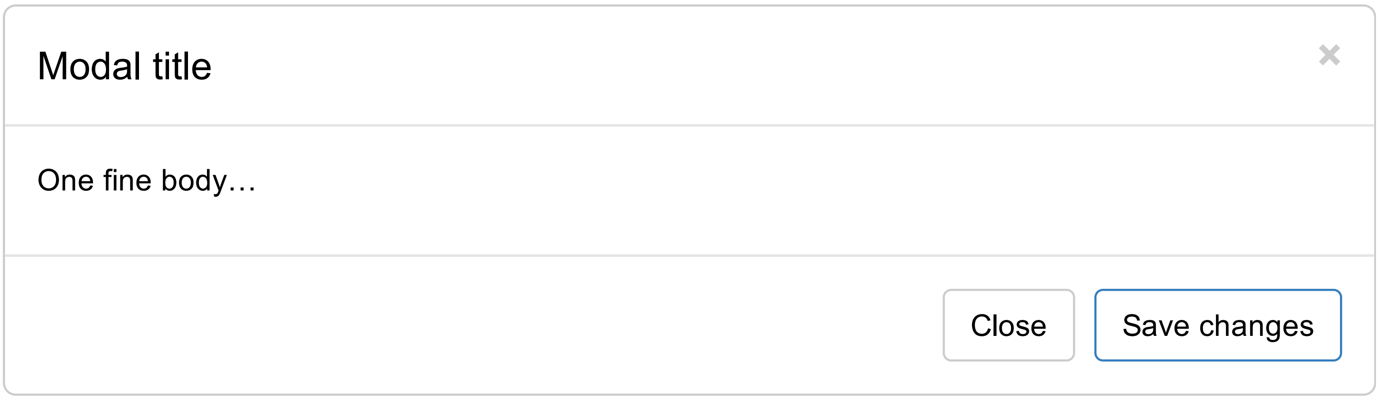
对于移动设备的附加说明

这里提供了在移动设备上使用模态框有一些附加说明。请参考浏览器支持 (../getting-started/#support-fixed-position-keyboards)章节。

静态实例

以下模态框包含了模态框的头、体和一组放置于底部的按钮。

Å®ŽÄ¾‹Ï¼Š



Modal title ×

One fine body…

Close Save changes

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

动态实例

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

Å®ŽÄ¾‹Ï¼Š

Launch demo modal

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

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

增强模态框的可访问性

务必为 `.modal` 添加 `role="dialog"` 属性，`aria-labelledby="myModalLabel"` 属性用于只想模态框的标题栏，`aria-hidden="true"` 用于通知辅助性工具略过模态框的DOM元素。

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

嵌入 YouTube 视频（天朝无用）

在模态框中嵌入 YouTube 视频需要增加一些额外的 JavaScript 代码，用于自动停止重放等功能，这些代码并没有在 Bootstrap 中提供。请参考这份发布在 Stack Overflow 上的文章 (http://stackoverflow.com/questions/18622508/bootstrap-3-and-youtube-in-modal)。

可选尺寸

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

Å®ŽÄ¾‹Ï¼Š

大模态框 小模态框

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

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

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

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

禁止动画效果

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

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

用法

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

闭模态框。

通过 data 属性

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

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

通过 JavaScript 调用

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

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

参数

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

名称	**类型**	**默认值**	**描述**
backdrop	boolean 或 字符串 `'static'`	true	Includes a modal-backdrop element. Alternatively, specify `static` for a backdrop which doesn't close the modal on click.
keyboard	boolean	true	键盘上的 esc 键被按下时关闭模态框。
show	boolean	true	模态框初始化之后就立即显示出来。
remote	path	false	This option is deprecated since v3.2.1 and will be removed in v4. We recommend instead using client-side templating or a data binding framework, or calling jQuery.load (http://api.jquery.com/load/) yourself. 如果提供的是 URL，将利用 jQuery 的 `load` 方法**从此 URL 地址加载要展示的内容（只加载一次）**并插入 `.modal-content` 内。如果使用的是 data 属性 API，还可以利用 `href` 属性指定内容来源地址。下面是一个实例： `<a data-toggle="modal" href="remote.html" data-target="#modal">Click me</a>`

方法

.modal(options)

将页面中的某块内容作为模态框激活。接受可选参数 `object` 。

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

.modal('toggle')

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

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

.modal('show')

手动打开模态框。**在模态框显示之前返回到主调函数中** （也就是，在触发 `shown.bs.modal` 事件之前）。

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

.modal('hide')

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

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

事件

Bootstrap 的模态框类提供了一些事件用于监听并执行你自己的代码。

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

hidden.bs.modal	此事件在模态框被隐藏（并且同时在 CSS 过渡效果完成）之后被触发。
loaded.bs.modal	从 远端的数据源 加载完数据之后触发该事件。

```
$('#myModal').on('hidden.bs.modal', function (e) {
  // do something...
})
```

Dropdowns dropdown.js

Examples

Add dropdown menus to nearly anything with this simple plugin, including the navbar, tabs, and pills.

Within a navbar

Å®ŽÄ¾‹Ï¼Š

Within pills

Å®ŽÄ¾‹Ï¼Š

Regular link   Dropdown ▾   Dropdown ▾   Dropdown ▾

Usage

Via data attributes or JavaScript, the dropdown plugin toggles hidden content (dropdown menus) by toggling the `.open` class on the parent list item.

On mobile devices, opening a dropdown adds a `.dropdown-backdrop` as a tap area for closing dropdown menus when tapping outside the menu, a requirement for proper iOS support. **This means that switching from an open dropdown menu to a different dropdown menu requires an extra tap on mobile.**

Note: The `data-toggle="dropdown"` attribute is relied on for closing dropdown menus at an application level, so it's a good idea to always use it.

Via data attributes

Add `data-toggle="dropdown"` to a link or button to toggle a dropdown.

```
<div class="dropdown">
  <button id="dLabel" type="button" data-toggle="dropdown" aria-haspopup="true"
aria-expanded="false">
    Dropdown trigger
   <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" role="menu" aria-labelledby="dLabel">
    ...
  </ul>
</div>
```

To keep URLs intact with link buttons, use the `data-target` attribute instead of `href="#"`.

```
<div class="dropdown">
  <a id="dLabel" data-target="#" href="http://example.com" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
    Dropdown trigger
    <span class="caret"></span>
  </a>

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

Via JavaScript

Call the dropdowns via JavaScript:

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

`data-toggle="dropdown"` still required

Regardless of whether you call your dropdown via JavaScript or instead use the data-api, `data-toggle="dropdown"` is always required to be present on the dropdown's trigger element.

Options

None

Methods

$().dropdown('toggle')

Toggles the dropdown menu of a given navbar or tabbed navigation.

Events

All dropdown events are fired at the `.dropdown-menu`'s parent element.

Event Type	Description
show.bs.dropdown	This event fires immediately when the show instance method is called. The toggling anchor element is available as the `relatedTarget` property of the event.
shown.bs.dropdown	This event is fired when the dropdown has been made visible to the user (will wait for CSS transitions, to complete). The toggling anchor element is available as the `relatedTarget` property of the event.
hide.bs.dropdown	This event is fired immediately when the hide instance method has been called. The toggling anchor element is available as the `relatedTarget` property of the event.
hidden.bs.dropdown	This event is fired when the dropdown has finished being hidden from the user (will wait for CSS transitions, to complete). The toggling anchor element is available as the `relatedTarget` property of the event.

```
$('#myDropdown').on('show.bs.dropdown', function () {
  // do something…
})
```

ScrollSpy scrollspy.js

Example in navbar

The ScrollSpy plugin is for automatically updating nav targets based on scroll position. Scroll the area below the navbar and watch the active class change. The dropdown sub items will be highlighted as well.

Å®ŽÄ¾‹Ï¼Š

@fat

Ad leggings keytar, brunch id art party dolor labore. Pitchfork yr enim lo-fi before they sold out qui. Tumblr farm-to-table bicycle rights whatever. Anim keffiyeh carles cardigan. Velit seitan mcsweeney's photo booth 3 wolf moon irure. Cosby sweater lomo jean shorts, williamsburg hoodie minim qui you probably haven't heard of them et cardigan trust fund culpa biodiesel wes anderson aesthetic. Nihil tattooed accusamus, cred irony biodiesel keffiyeh artisan ullamco consequat.

@mdo

Veniam marfa mustache skateboard, adipisicing fugiat velit pitchfork beard. Freegan beard aliqua

Usage

Requires relative positioning

No matter the implementation method, scrollspy requires the use of `position: relative;` on the element you're spying on. In most cases this is the `<body>` .

Via data attributes

To easily add scrollspy behavior to your topbar navigation, add `data-spy="scroll"` to the element you want to spy on (most typically this would be the `<body>`). Then add the `data-target` attribute with the ID or class of the parent element of any Bootstrap `.nav` component.

```
body {
  position: relative;
}
```

```
<body data-spy="scroll" data-target=".navbar-example">
  ...
  <div class="navbar-example">
    <ul class="nav nav-tabs" role="tablist">
      ...
    </ul>
  </div>
  ...
</body>
```

Via JavaScript

After adding `position: relative;` in your CSS, call the scrollspy via JavaScript:

```
$('body').scrollspy({ target: '.navbar-example' })
```

Resolvable ID targets required

Navbar links must have resolvable id targets. For example, a `<a href="#home">home</a>` must correspond to something in the DOM like `<div id="home"></div>`.

Non- `:visible` target elements ignored

Target elements that are not `:visible` according to jQuery (http://api.jquery.com/visible-selector/) will be ignored and their corresponding nav items will never be highlighted.

Methods

.scrollspy('refresh')

When using scrollspy in conjunction with adding or removing of elements from the DOM, you'll need to call the refresh method like so:

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

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-offset=""`.

Name	type	default	description
offset	number	10	Pixels to offset from top when calculating position of scroll.

Events

Event Type	Description
activate.bs.scrollspy	This event fires whenever a new item becomes activated by the scrollspy.

```
$('#myScrollspy').on('activate.bs.scrollspy', function () {
  // do something…
})
```

Togglable tabs tab.js

Example tabs

Add quick, dynamic tab functionality to transition through panes of local content, even via dropdown menus.

Å®ŽÄ¾‹Ï¼Š

Home Profile Dropdown

Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcatcher synth. Cosby sweater eu banh mi, qui irure terry richardson ex squid. Aliquip placeat salvia cillum iphone. Seitan aliquip quis cardigan american apparel, butcher voluptate nisi qui.

> Extends tabbed navigation
>
> This plugin extends the tabbed navigation component (../components/#nav-tabs) to add tabbable areas.

Usage

Enable tabbable tabs via JavaScript (each tab needs to be activated individually):

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

You can activate individual tabs in several ways:

```
$('#myTab a[href="#profile"]').tab('show') // Select tab by name
$('#myTab a:first').tab('show') // Select first tab
$('#myTab a:last').tab('show') // Select last tab
$('#myTab li:eq(2) a').tab('show') // Select third tab (0-indexed)
```

Markup

You can activate a tab or pill navigation without writing any JavaScript by simply specifying `data-toggle="tab"` or `data-toggle="pill"` on an element. Adding the `nav` and `nav-tabs` classes to the tab `ul` will apply the Bootstrap tab styling (../components/#nav-tabs), while adding the `nav` and `nav-pills` classes will apply pill styling (../components/#nav-pills).

```
<!-- Nav tabs -->
<ul class="nav nav-tabs" role="tablist">
  <li class="active"><a href="#home" role="tab" data-toggle="tab">Home</a></li>
  <li><a href="#profile" role="tab" data-toggle="tab">Profile</a></li>
  <li><a href="#messages" role="tab" data-toggle="tab">Messages</a></li>
  <li><a href="#settings" role="tab" data-toggle="tab">Settings</a></li>
</ul>

<!-- Tab panes -->
<div class="tab-content">
  <div class="tab-pane active" id="home">...</div>
  <div class="tab-pane" id="profile">...</div>
  <div class="tab-pane" id="messages">...</div>
  <div class="tab-pane" id="settings">...</div>
</div>
```

Fade effect

To make tabs fade in, add `.fade` to each `.tab-pane`. The first tab pane must also have `.in` to properly fade in initial content.

```
<div class="tab-content">
  <div class="tab-pane fade in active" id="home">...</div>
  <div class="tab-pane fade" id="profile">...</div>
  <div class="tab-pane fade" id="messages">...</div>
  <div class="tab-pane fade" id="settings">...</div>
</div>
```

Methods

$().tab

Activates a tab element and content container. Tab should have either a `data-target` or an `href` targeting a container node in the DOM.

```
<ul class="nav nav-tabs" role="tablist" id="myTab">
  <li class="active"><a href="#home" role="tab" data-toggle="tab">Home</a></li>
  <li><a href="#profile" role="tab" data-toggle="tab">Profile</a></li>
  <li><a href="#messages" role="tab" data-toggle="tab">Messages</a></li>
  <li><a href="#settings" role="tab" data-toggle="tab">Settings</a></li>
</ul>

<div class="tab-content">
  <div class="tab-pane active" id="home">...</div>
  <div class="tab-pane" id="profile">...</div>
  <div class="tab-pane" id="messages">...</div>
  <div class="tab-pane" id="settings">...</div>
</div>

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

Events

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

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

Tooltips tooltip.js

Inspired by the excellent jQuery.tipsy plugin written by Jason Frame; Tooltips are an updated version, which don't rely on images, use CSS3 for animations, and data-attributes for local title storage.

Tooltips with zero-length titles are never displayed.

Examples

Hover over the links below to see tooltips:

Å®ŽÄ¾‹Ï¼Š

Tight pants next level keffiyeh you probably haven't heard of them. Photo booth beard raw denim letterpress vegan messenger bag stumptown. Farm-to-table seitan, mcsweeney's fixie sustainable quinoa 8-bit american apparel have a terry richardson vinyl chambray. Beard stumptown, cardigans banh mi lomo thundercats. Tofu biodiesel williamsburg marfa, four loko mcsweeney's cleanse vegan chambray. A really ironic artisan whatever keytar, scenester farm-to-table banksy Austin twitter handle freegan cred raw denim single-origin coffee viral.

Static tooltip

Four options are available: top, right, bottom, and left aligned.

Å®ŽÄ¾‹Ï¼Š

Tooltip on the left ▸ Tooltip on the top ▾ ▴ Tooltip on the bottom

◂ Tooltip on the right

Four directions

Å®ŽÄ¾‹Ï¼Š

Tooltip on left | Tooltip on top | Tooltip on bottom | Tooltip on right

```
<button type="button" class="btn btn-default" data-toggle="tooltip" data-
placement="left" title="Tooltip on left">Tooltip on left</button>

<button type="button" class="btn btn-default" data-toggle="tooltip" data-
placement="top" title="Tooltip on top">Tooltip on top</button>

<button type="button" class="btn btn-default" data-toggle="tooltip" data-
placement="bottom" title="Tooltip on bottom">Tooltip on bottom</button>

<button type="button" class="btn btn-default" data-toggle="tooltip" data-
placement="right" title="Tooltip on right">Tooltip on right</button>
```

Opt-in functionality

For performance reasons, the Tooltip and Popover data-apis are opt-in, meaning **you must initialize them yourself**.

Tooltips in button groups and input groups require special setting

When using tooltips on elements within a `.btn-group` or an `.input-group`, you'll have to specify the option `container: 'body'` (documented below) to avoid unwanted side effects (such as the element growing wider and/or losing its rounded corners when the tooltip is triggered).

Don't try to show tooltips on hidden elements

Invoking `$(...).tooltip('show')` when the target element is `display: none;` will cause the tooltip to be incorrectly positioned.

Tooltips on disabled elements require wrapper elements

To add a tooltip to a `disabled` or `.disabled` element, put the element inside of a `<div>` and apply the tooltip to that `<div>` instead.

Usage

The tooltip plugin generates content and markup on demand, and by default places tooltips after their trigger element.

Trigger the tooltip via JavaScript:

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

Markup

The required markup for a tooltip is only a `data` attribute and `title` on the HTML element you wish to have a tooltip. The generated markup of a tooltip is rather simple, though it does require a position (by default, set to `top` by the plugin).

Multiple-line links

Sometimes you want to add a tooltip to a hyperlink that wraps multiple lines. The default behavior of the tooltip plugin is to center it horizontally and vertically. Add `white-space: nowrap;` to your anchors to avoid this.

```
<!-- HTML to write -->
<a href="#" data-toggle="tooltip" title="Some tooltip text!">Hover over me</a>

<!-- Generated markup by the plugin -->
<div class="tooltip top" role="tooltip">
  <div class="tooltip-arrow"></div>
  <div class="tooltip-inner">
    Some tooltip text!
  </div>
</div>
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-animation=""`.

Name	Type	Default	Description
animation	boolean	true	Apply a CSS fade transition to the tooltip
container	string | false	false	Appends the tooltip to a specific element. Example: `container: 'body'`. This option is particularly useful in that it allows you to position the tooltip in the flow of the document near the triggering element - which will prevent the tooltip from floating away from the triggering element during a window resize.

delay	number | object	0	Delay showing and hiding the tooltip (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: `delay: { "show": 500, "hide": 100 }`
html	boolean	false	Insert HTML into the tooltip. If false, jQuery's `text` method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
placement	string | function	'top'	How to position the tooltip - top | bottom | left | right | auto. When "auto" is specified, it will dynamically reorient the tooltip. For example, if placement is "auto left", the tooltip will display to the left when possible, otherwise it will display right. When a function is used to determine the placement, it is called with the tooltip DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the tooltip instance.
selector	string	false	If a selector is provided, tooltip objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have tooltips added. See this (https://github.com/twbs/bootstrap/issues/4215) and an informative example (http://jsbin.com/zopod/1/edit).
template	string	`'<div class="tooltip" role="tooltip"><div class="tooltip-arrow"></div><div class="tooltip-inner"></div></div>'`	Base HTML to use when creating the tooltip. The tooltip's `title` will be injected into the `.tooltip-inner`. `.tooltip-arrow` will become the tooltip's arrow. The outermost wrapper element should have the `.tooltip` class.
title	string | function	''	Default title value if `title` attribute isn't present.

			If a function is given, it will be called with its `this` reference set to the element that the tooltip is attached to.
trigger	string	'hover focus'	How tooltip is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space.
viewport	string | object	{ selector: 'body', padding: 0 }	Keeps the tooltip within the bounds of this element. Example: `viewport: '#viewport'` or `{ "selector": "#viewport", "padding": 0 }`

Data attributes for individual tooltips

Options for individual tooltips can alternatively be specified through the use of data attributes, as explained above.

Methods

$().tooltip(options)

Attaches a tooltip handler to an element collection.

.tooltip('show')

Reveals an element's tooltip.

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

.tooltip('hide')

Hides an element's tooltip.

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

.tooltip('toggle')

Toggles an element's tooltip.

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

.tooltip('destroy')

Hides and destroys an element's tooltip.

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

Events

Event Type	Description
show.bs.tooltip	This event fires immediately when the `show` instance method is called.
shown.bs.tooltip	This event is fired when the tooltip has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.tooltip	This event is fired immediately when the `hide` instance method has been called.
hidden.bs.tooltip	This event is fired when the tooltip has finished being hidden from the user (will wait for CSS transitions to complete).

```
$('#myTooltip').on('hidden.bs.tooltip', function () {
  // do something…
})
```

Popovers popover.js

Examples

Add small overlays of content, like those on the iPad, to any element for housing secondary information.

Popovers whose both title and content are zero-length are never displayed.

Plugin dependency

Popovers require the tooltip plugin to be included in your version of Bootstrap.

Opt-in functionality

For performance reasons, the Tooltip and Popover data-apis are opt-in, meaning **you must initialize them yourself**.

Popovers in button groups and input groups require special setting

When using popovers on elements within a `.btn-group` or an `.input-group`, you'll have to specify the option `container: 'body'` (documented below) to avoid unwanted side effects (such as the element growing wider and/or losing its rounded corners when the popover is triggered).

Don't try to show popovers on hidden elements

Invoking `$(...).popover('show')` when the target element is `display: none;` will cause the popover to be incorrectly positioned.

Popovers on disabled elements require wrapper elements

To add a popover to a `disabled` or `.disabled` element, put the element inside of a `<div>` and apply the popover to that `<div>` instead.

Multiple-line links

Sometimes you want to add a popover to a hyperlink that wraps multiple lines. The default behavior of the popover plugin is to center it horizontally and vertically. Add `white-space: nowrap;` to your anchors to avoid this.

Static popover

Four options are available: top, right, bottom, and left aligned.

Å®ŽÄ¾‹Ï¼Š

Popover top

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover right

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover bottom

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Popover left

Sed posuere consectetur est at lobortis. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.

Live demo

Å®ŽÄ¾‹Ï¼Š

Click to toggle popover

```
<button type="button" class="btn btn-lg btn-danger" data-toggle="popover"
title="Popover title" data-content="And here's some amazing content. It's very
engaging. Right?">Click to toggle popover</button>
```

Four directions

Å®ŽÄ¾‹Ï¼Š

Popover on left | Popover on top | Popover on bottom | Popover on right

```
<button type="button" class="btn btn-default" data-container="body" data-
toggle="popover" data-placement="left" data-content="Vivamus sagittis lacus vel augue
laoreet rutrum faucibus.">
  Popover on left
</button>

<button type="button" class="btn btn-default" data-container="body" data-
toggle="popover" data-placement="top" data-content="Vivamus sagittis lacus vel augue
laoreet rutrum faucibus.">
  Popover on top
</button>

<button type="button" class="btn btn-default" data-container="body" data-
toggle="popover" data-placement="bottom" data-content="Vivamus
sagittis lacus vel augue laoreet rutrum faucibus.">
  Popover on bottom
</button>

<button type="button" class="btn btn-default" data-container="body" data-
toggle="popover" data-placement="right" data-content="Vivamus sagittis lacus vel
augue laoreet rutrum faucibus.">
  Popover on right
</button>
```

Dismiss on next click

Use the `focus` trigger to dismiss popovers on the next click that the user makes.

Specific markup required for dismiss-on-next-click

For proper cross-browser and cross-platform behavior, you must use the `<a>` tag, *not* the `<button>` tag, and you also must include a `tabindex` (https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes#tabindex) attribute.

Å®ŽÄ¾‹Ï¼Š

Dismissible popover

```
<a href="#" tabindex="0" class="btn btn-lg btn-danger" data-toggle="popover" data-
trigger="focus" title="Dismissible popover" data-content="And here's some amazing
content. It's very engaging. Right?">Dismissible popover</a>
```

Usage

Enable popovers via JavaScript:

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

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-` , as in `data-animation=""` .

Name	**Type**	**Default**	**Description**
animation	boolean	true	Apply a CSS fade transition to the popover
container	string | false	false	Appends the popover to a specific element. Example: `container: 'body'` . This option is particularly useful in that it allows you to position the popover in the flow of the document near the triggering element - which will prevent the popover from floating away from the triggering element during a window resize.
content	string | function	''	Default content value if `data-content` attribute isn't present. If a function is given, it will be called with its `this` reference set to the element that the popover is attached to.
delay	number | object	0	Delay showing and hiding the popover (ms) - does not apply to manual trigger type If a number is supplied, delay is applied to both hide/show Object structure is: `delay: { "show": 500, "hide": 100 }`
html	boolean	false	Insert HTML into the popover. If false, jQuery's `text` method will be used to insert content into the DOM. Use text if you're worried about XSS attacks.
placement	string | function	'right'	How to position the popover - top | bottom | left | right | auto. When "auto" is specified, it will dynamically

			reorient the popover. For example, if placement is "auto left", the popover will display to the left when possible, otherwise it will display right. When a function is used to determine the placement, it is called with the popover DOM node as its first argument and the triggering element DOM node as its second. The `this` context is set to the popover instance.
selector	string	false	If a selector is provided, popover objects will be delegated to the specified targets. In practice, this is used to enable dynamic HTML content to have popovers added. See this (https://github.com/twbs/bootstrap/issues/4215) and an informative example (http://jsbin.com/zopod/1/edit).
template	string	`'<div class="popover" role="tooltip"><div class="arrow"></div><h3 class="popover-title"></h3><div class="popover-content"></div></div>'`	Base HTML to use when creating the popover. The popover's `title` will be injected into the `.popover-title`. The popover's `content` will be injected into the `.popover-content`. `.arrow` will become the popover's arrow. The outermost wrapper element should have the `.popover` class.
title	string | function	''	Default title value if `title` attribute isn't present. If a function is given, it will be called with its `this` reference set to the element that the popover is attached to.
trigger	string	'click'	How popover is triggered - click | hover | focus | manual. You may pass multiple triggers; separate them with a space.
viewport	string | object	{ selector: 'body', padding: 0 }	Keeps the popover within the bounds of this element. Example: `viewport: '#viewport'` or `{ "selector": "#viewport", "padding": 0 }`

Data attributes for individual popovers

Options for individual popovers can alternatively be specified through the use of data attributes, as explained above.

Methods

$().popover(options)

Initializes popovers for an element collection.

.popover('show')

Reveals an element's popover.

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

.popover('hide')

Hides an element's popover.

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

.popover('toggle')

Toggles an element's popover.

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

.popover('destroy')

Hides and destroys an element's popover.

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

Events

Event Type	Description
show.bs.popover	This event fires immediately when the `show` instance method is called.
shown.bs.popover	This event is fired when the popover has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.popover	This event is fired immediately when the `hide` instance method has been called.
hidden.bs.popover	This event is fired when the popover has finished being hidden from the

	user (will wait for CSS transitions to complete).

```
$('#myPopover').on('hidden.bs.popover', function () {
  // do something…
})
```

警告框 alert.js

警告框实例

Add dismiss functionality to all alert messages with this plugin.

Å®ŽÄ¾‹Ï¼Š

Holy guacamole! Best check yo self, you're not looking too good. ×

Oh snap! You got an error! ×

Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.

Take this action Or do this

用法

Just add `data-dismiss="alert"` to your close button to automatically give an alert close functionality. Closing an alert removes it from the DOM.

```
<button type="button" class="close" data-dismiss="alert"><span aria-
hidden="true">&times;</span><span class="sr-only">Close</span></button>
```

```
<button type="button" class="close" data-dismiss="alert">
  <span aria-hidden="true">&times;</span>
  <span class="sr-only">Close</span>
</button>
```

To have your alerts use animation when closing, make sure they have the `.fade` and `.in` classes already applied to them.

Methods

$().alert()

Makes an alert listen for click events on descendant elements which have the `data-dismiss="alert"` attribute. (Not necessary when using the data-api's auto-initialization.)

$().alert('close')

Closes an alert by removing it from the DOM. If the `.fade` and `.in` classes are present on the element, the alert will fade out before it is removed.

Events

Bootstrap's alert plugin exposes a few events for hooking into alert functionality.

Event Type	Description
close.bs.alert	This event fires immediately when the `close` instance method is called.
closed.bs.alert	This event is fired when the alert has been closed (will wait for CSS transitions to complete).

```
$('#myAlert').on('closed.bs.alert', function () {
  // do something…
})
```

Buttons button.js

Do more with buttons. Control button states or create groups of buttons for more components like toolbars.

Cross-browser compatibility

Firefox persists form control states (disabledness and checkedness) across page loads (https://github.com/twbs/bootstrap/issues/793). A workaround for this is to use `autocomplete="off"`. See Mozilla bug #654072 (https://bugzilla.mozilla.org/show_bug.cgi?id=654072).

Stateful

Add `data-loading-text="Loading..."` to use a loading state on a button.

> **Use whichever state you like!**
> For the sake of this demonstration, we are using `data-loading-text` and `$().button('loading')`, but that's not the only state you can use. See more on this below in the `$().button(string)` documentation.

Å®ŽÄ¾‹Ï¼Š

Loading state

```
<button type="button" id="myButton" data-loading-text="Loading..." class="btn btn-
primary" autocomplete="off">
  Loading state
</button>

<script>
  $('#myButton').on('click', function () {
    var $btn = $(this).button('loading')
    // business logic...
    $btn.button('reset')
  })
</script>
```

Single toggle

Add `data-toggle="button"` to activate toggling on a single button.

Å®ŽÄ¾‹Ï¼Š

Single toggle

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

Checkbox / Radio

Add `data-toggle="buttons"` to a `.btn-group` containing checkbox or radio inputs to enable toggling in their respective styles.

Preselected options need `.active`

For preselected options, you must add the `.active` class to the input's `label` yourself.

Visual checked state only updated on click

If the checked state of a checkbox button is updated without firing a `click` event on the button (e.g. via `<input type="reset">` or via setting the `checked` property of the input), you will need to toggle the `.active` class on the input's `label` yourself.

Å®ŽÄ¾‹Ï¼Š

Checkbox 1 (pre-checked) | Checkbox 2 | Checkbox 3

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary active">
    <input type="checkbox" autocomplete="off" checked> Checkbox 1 (pre-checked)
  </label>
  <label class="btn btn-primary">
    <input type="checkbox" autocomplete="off"> Checkbox 2
  </label>
  <label class="btn btn-primary">
    <input type="checkbox" autocomplete="off"> Checkbox 3
  </label>
</div>
```

Å®ŽÄ¾‹Ï¼Š

Radio 1 (preselected) | Radio 2 | Radio 3

```
<div class="btn-group" data-toggle="buttons">
  <label class="btn btn-primary active">
    <input type="radio" name="options" id="option1" autocomplete="off" checked> Radio
1 (preselected)
  </label>
  <label class="btn btn-primary">
    <input type="radio" name="options" id="option2" autocomplete="off"> Radio 2
  </label>
  <label class="btn btn-primary">
    <input type="radio" name="options" id="option3" autocomplete="off"> Radio 3
  </label>
</div>
```

Methods

$().button('toggle')

Toggles push state. Gives the button the appearance that it has been activated.

$().button('reset')

Resets button state - swaps text to original text.

$().button(string)

Swaps text to any data defined text state.

```
<button type="button" id="myStateButton" data-complete-text="finished!"
class="btn btn-primary" autocomplete="off">
  ...
</button>

<script>
  $('#myStateButton').on('click', function () {
    $(this).button('complete') // button text will be "finished!"
  })
</script>
```

Collapse collapse.js

About

Get base styles and flexible support for collapsible components like accordions and navigation.

Plugin dependency

Collapse requires the transitions plugin to be included in your version of Bootstrap.

Example accordion

Using the collapse plugin, we built a simple accordion by extending the panel component.

Å®ŽÄ¾‹Ï¼Š

Collapsible Group Item #1

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #2

Collapsible Group Item #3

```
<div class="panel-group" id="accordion" role="tablist" aria-multiselectable="true">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#accordion" href="#collapseOne" aria-
expanded="true" aria-controls="collapseOne">
          Collapsible Group Item #1
        </a>
      </h4>
    </div>
    <div id="collapseOne" class="panel-collapse collapse in" role="tabpanel">
      <div class="panel-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry
richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch.
Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua
put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim
keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea
proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-
table, raw denim aesthetic synth nesciunt you probably haven't heard of them
accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#accordion" href="#collapseTwo" aria-
expanded="false" aria-controls="collapseTwo">
          Collapsible Group Item #2
        </a>
      </h4>
    </div>
    <div id="collapseTwo" class="panel-collapse collapse" role="tabpanel">
      <div class="panel-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry
richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch.
Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua
put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim
keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea
proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-
table, raw denim aesthetic synth nesciunt you probably haven't heard of them
accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">
        <a data-toggle="collapse" data-parent="#accordion" href="#collapseThree"
aria-expanded="false" aria-controls="collapseThree">
          Collapsible Group Item #3
        </a>
      </h4>
    </div>
    <div id="collapseThree" class="panel-collapse collapse" role="tabpanel">
      <div class="panel-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry
richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch.
Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua
put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim
keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea
proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-
table, raw denim aesthetic synth nesciunt you probably haven't heard of them
accusamus labore sustainable VHS.
      </div>
    </div>
  </div>
</div>
```

You can also use the plugin without the accordion markup. Make a button toggle the expanding and collapsing of another element.

```
<button type="button" class="btn btn-danger" data-toggle="collapse" data-
target="#demo" aria-expanded="true" aria-controls="demo">
  simple collapsible
</button>

<div id="demo" class="collapse in">...</div>
```

Make expand/collapse controls accessible

Be sure to add `aria-expanded` to the control element. This attribute explicitly defines the current state of the collapsible element to screen readers and similar assistive technologies. If the collapsible element is closed by default, it should have a value of `aria-expanded="false"`. If you've set the collapsible element to be open by default using the `in` class, set `aria-expanded="true"` on the control instead. The plugin will automatically toggle this attribute based on whether or not the collapsible element has been opened or closed.

Additionally, if your control element is targetting a single collapsible element – i.e. the `data-target` attribute is pointing to an `id` selector – you may add an additional `aria-controls` attribute to the control element, containing the `id` of the collapsible element. Modern screen readers and similar assistive technologies make use of this attribute to provide users with additional shortcuts to navigate directly to the collapsible element itself.

Usage

The collapse plugin utilizes a few classes to handle the heavy lifting:

- `.collapse` hides the content
- `.collapse.in` shows the content
- `.collapsing` is added when the transition starts, and removed when it finishes

These classes can be found in `component-animations.less`.

Via data attributes

Just add `data-toggle="collapse"` and a `data-target` to the element to automatically assign control of a collapsible element. The `data-target` attribute accepts a CSS selector to apply the collapse to. Be sure to add the class `collapse` to the collapsible element. If you'd like it to default open, add the additional class `in` .

To add accordion-like group management to a collapsible control, add the data attribute `data-parent="#selector"` . Refer to the demo to see this in action.

Via JavaScript

Enable manually with:

```
$('.collapse').collapse()
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-` , as in `data-parent=""` .

Name	type	default	description
parent	selector	false	If a selector is provided, then all collapsible elements under the specified parent will be closed when this collapsible item is shown. (similar to traditional accordion behavior - this is dependent on the `panel` class)
toggle	boolean	true	Toggles the collapsible element on invocation

Methods

.collapse(options)

Activates your content as a collapsible element. Accepts an optional options `object` .

```
$('#myCollapsible').collapse({
  toggle: false
})
```

.collapse('toggle')

Toggles a collapsible element to shown or hidden.

.collapse('show')

Shows a collapsible element.

.collapse('hide')

Hides a collapsible element.

Events

Bootstrap's collapse class exposes a few events for hooking into collapse functionality.

Event Type	Description
show.bs.collapse	This event fires immediately when the `show` instance method is called.
shown.bs.collapse	This event is fired when a collapse element has been made visible to the user (will wait for CSS transitions to complete).
hide.bs.collapse	This event is fired immediately when the `hide` method has been called.
hidden.bs.collapse	This event is fired when a collapse element has been hidden from the user (will wait for CSS transitions to complete).

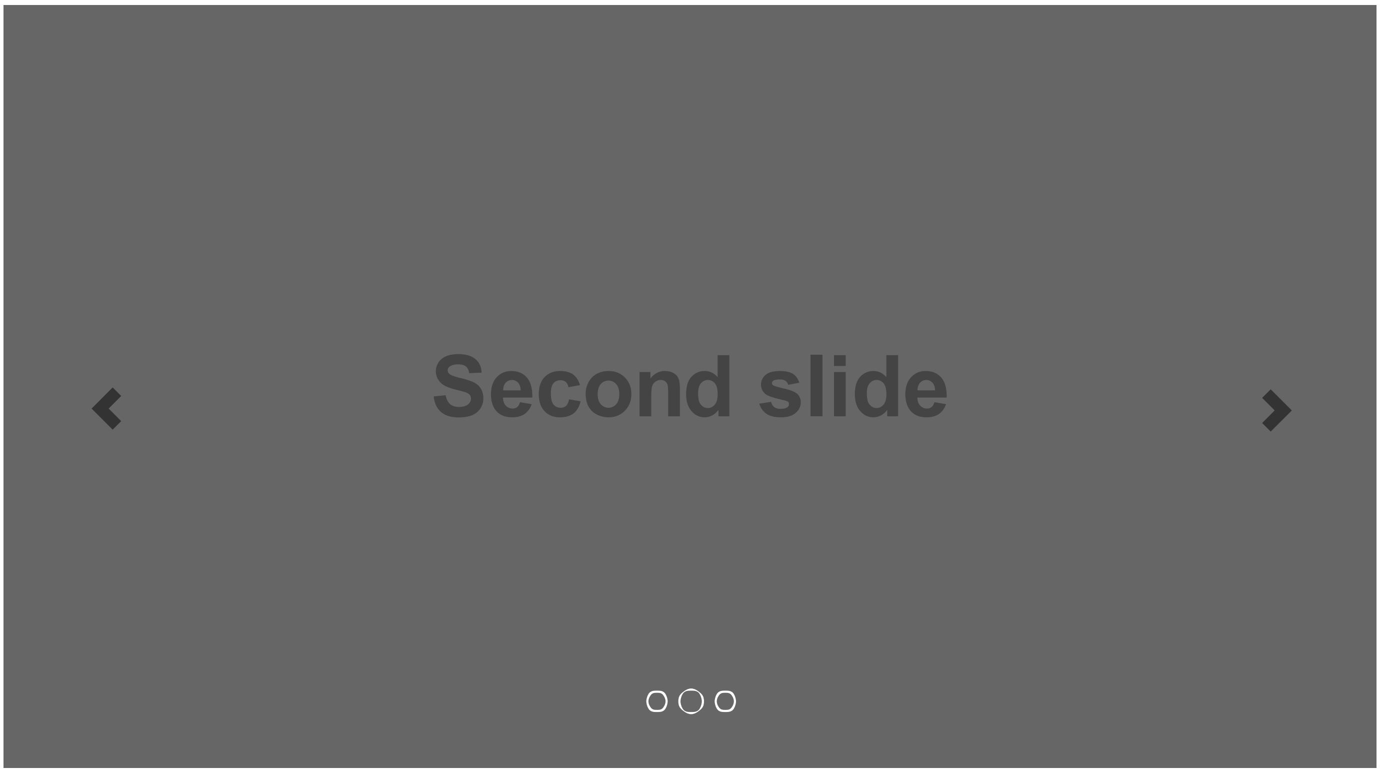
```
$('#myCollapsible').on('hidden.bs.collapse', function () {
  // do something…
})
```

Carousel carousel.js

A slideshow component for cycling through elemnts, like a carousel. **Nested carousels are not supported.**

Examples

Å®ŽÄ¾‹Ï¼Š



```
<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#carousel-example-generic" data-slide-to="0" class="active">
</li>
    <li data-target="#carousel-example-generic" data-slide-to="1"></li>
    <li data-target="#carousel-example-generic" data-slide-to="2"></li>
  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner" role="listbox">
    <div class="item active">
      <img src="..." alt="...">
      <div class="carousel-caption">
        ...
      </div>
    </div>
    <div class="item">
      <img src="..." alt="...">
      <div class="carousel-caption">
        ...
      </div>
    </div>
    ...
  </div>

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

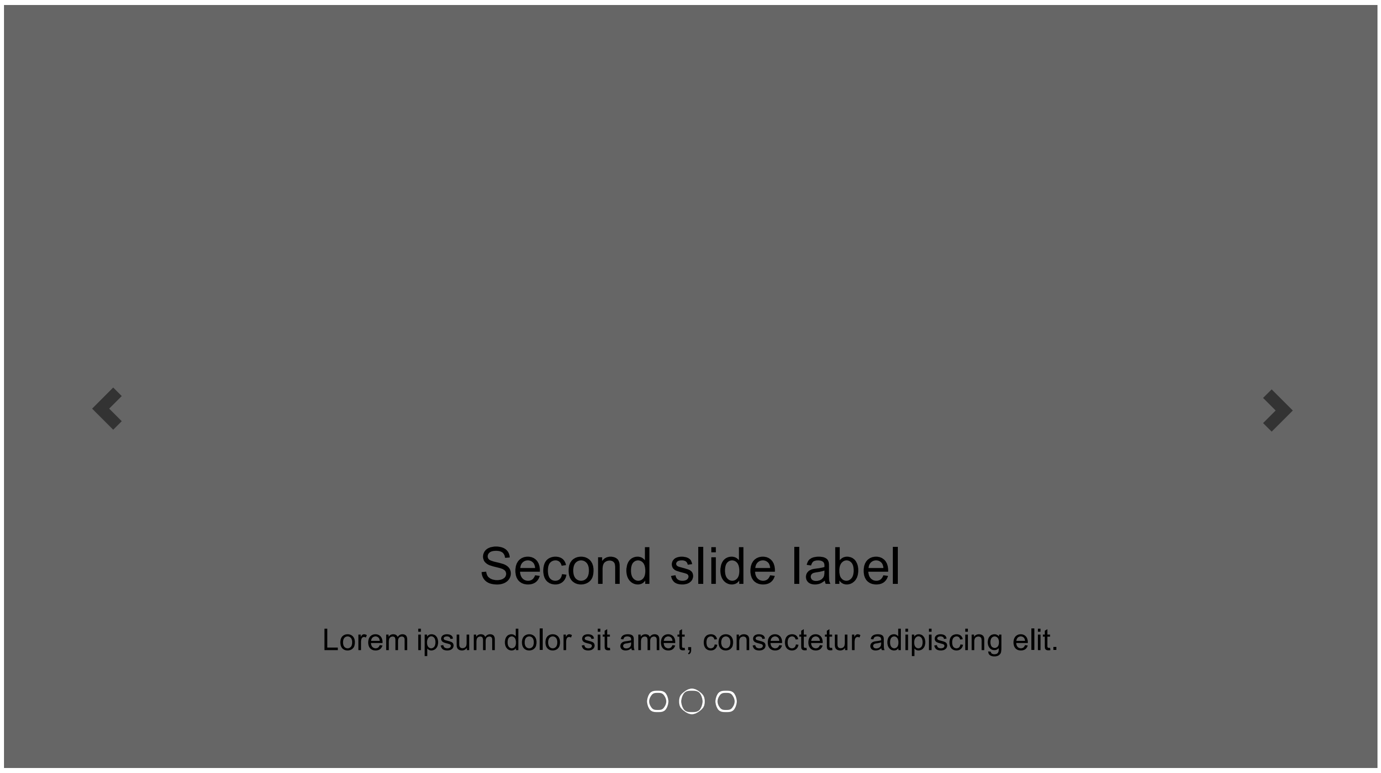
Transition animations not supported in Internet Explorer 8 & 9

Bootstrap exclusively uses CSS3 for its animations, but Internet Explorer 8 & 9 don't support the necessary CSS properties. Thus, there are no slide transition animations when using these browsers. We have intentionally decided not to include jQuery-based fallbacks for the transitions.

Optional captions

Add captions to your slides easily with the `.carousel-caption` element within any `.item`. Place just about any optional HTML within there and it will be automatically aligned and formatted.

Å®ŽÄ¾‹Ï¼Š



Second slide label

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
<div class="item">
  <img src="..." alt="...">
  <div class="carousel-caption">
    <h3>...</h3>
    <p>...</p>
  </div>
</div>
```

Accessibility issue

The carousel component is generally not compliant with accessibility standards. If you need to be compliant, please consider other options for presenting your content.

Usage

Multiple carousels

Carousels require the use of an `id` on the outermost container (the `.carousel`) for carousel controls to function properly. When adding multiple carousels, or when changing a carousel's `id`, be sure to update the relevant controls.

Via data attributes

Use data attributes to easily control the position of the carousel. `data-slide` accepts the keywords `prev` or `next`, which alters the slide position relative to its current position. Alternatively, use `data-slide-to` to pass a raw slide index to the carousel `data-slide-to="2"`, which shifts the slide position to a particular index beginning with `0`.

The `data-ride="carousel"` attribute is used to mark a carousel as animating starting at page load. **It cannot be used in combination with (redundant and unnecessary) explicit JavaScript initialization of the same carousel.**

Via JavaScript

Call carousel manually with:

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

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-`, as in `data-interval=""`.

Name	type	default	description
interval	number	5000	The amount of time to delay between automatically cycling an item. If false, carousel will not automatically cycle.
pause	string	"hover"	Pauses the cycling of the carousel on mouseenter and resumes the cycling of the carousel on mouseleave.
wrap	boolean	true	Whether the carousel should cycle continuously or have hard stops.

Methods

.carousel(options)

Initializes the carousel with an optional options `object` and starts cycling through items.

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

.carousel('cycle')

Cycles through the carousel items from left to right.

.carousel('pause')

Stops the carousel from cycling through items.

.carousel(number)

Cycles the carousel to a particular frame (0 based, similar to an array).

.carousel('prev')

Cycles to the previous item.

.carousel('next')

Cycles to the next item.

Events

Bootstrap's carousel class exposes two events for hooking into carousel functionality.

Both events have the following additional properties:

- `direction` : The direction in which the carousel is sliding (either `"left"` or `"right"`).
- `relatedTarget` : The DOM element that is being slid into place as the active item.

Event Type	Description
slide.bs.carousel	This event fires immediately when the `slide` instance method is invoked.
slid.bs.carousel	This event is fired when the carousel has completed its slide transition.

```
$('#myCarousel').on('slide.bs.carousel', function () {
  // do something…
})
```

Affix affix.js

Example

The subnavigation on the right is a live demo of the affix plugin.

Usage

Use the affix plugin via data attributes or manually with your own JavaScript. **In both situations, you must provide CSS for the positioning and width of your affixed content.**

Positioning via CSS

The affix plugin toggles between three classes, each representing a particular state: `.affix`, `.affix-top`, and `.affix-bottom`. You must provide the styles for these classes yourself (independent of this plugin) to handle the actual positions.

Here's how the affix plugin works:

1. To start, the plugin adds `.affix-top` to indicate the element is in its top-most position. At this point no CSS positioning is required.
2. Scrolling past the element you want affixed should trigger the actual affixing. This is where `.affix` replaces `.affix-top` and sets `position: fixed;` (provided by Bootstrap's CSS).
3. If a bottom offset is defined, scrolling past it should replace `.affix` with `.affix-bottom`. Since offsets are optional, setting one requires you to set the appropriate CSS. In this case, add `position: absolute;` when necessary. The plugin uses the data attribute or JavaScript option to determine where to position the element from there.

Follow the above steps to set your CSS for either of the usage options below.

Via data attributes

To easily add affix behavior to any element, just add `data-spy="affix"` to the element you want to spy on. Use offsets to define when to toggle the pinning of an element.

```
<div data-spy="affix" data-offset-top="60" data-offset-bottom="200">
  ...
</div>
```

Via JavaScript

Call the affix plugin via JavaScript:

```
$('#myAffix').affix({
  offset: {
    top: 100,
    bottom: function () {
      return (this.bottom = $('.footer').outerHeight(true))
    }
  }
})
```

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-` , as in `data-offset-top="200"` .

Name	type	default	description
offset	number | function | object	10	Pixels to offset from screen when calculating position of scroll. If a single number is provided, the offset will be applied in both top and bottom directions. To provide a unique, bottom and top offset just provide an object `offset: { top: 10 }` or `offset: { top: 10, bottom: 5 }` . Use a function when you need to dynamically calculate an offset.
target	selector | node | jQuery element	the `window` object	Specifies the target element of the affix.

Events

Bootstrap's affix plugin exposes a few events for hooking into affix functionality.

Event Type	Description
affix.bs.affix	This event fires immediately before the element has been affixed.
affixed.bs.affix	This event is fired after the element has been affixed.
affix-top.bs.affix	This event fires immediately before the element has been affixed-top.
affixed-top.bs.affix	This event is fired after the element has been affixed-top.
affix-bottom.bs.affix	This event fires immediately before the element has been affixed-bottom.

affixed-bottom.bs.affix	This event is fired after the element has been affixed-bottom.