Navs and tabs

Documentation and examples for how to use Bootstrap's included navigation components.

On this page

constructor
show
dispose
getInstance
getOrCreateInstance
Events

Base nav

Navigation available in Bootstrap share general markup and styles, from the base `.nav` class to the active and disabled states. Swap modifier classes to switch between each style.

The base `.nav` component is built with flexbox and provide a strong foundation for building all types of navigation components. It includes some style overrides (for working with lists), some link padding for larger hit areas, and basic disabled styling.

> The base `.nav` component does not include any `.active` state. The following examples include the class, mainly to demonstrate that this particular class does not trigger any special styling.
>
> To convey the active state to assistive technologies, use the `aria-current` attribute — using the `page` value for current page, or `true` for the current item in a set.

Active Link Link Disabled

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Classes are used throughout, so your markup can be super flexible. Use `<ul>`s like above, `<ol>` if the order of your items is important, or roll your own with a `<nav>` element. Because the `.nav` uses `display: flex`, the nav links behave the same as nav items would, but without the extra markup.

Active Link Link Disabled

```
<nav class="nav">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

Available styles

Change the style of `.navs` component with modifiers and utilities. Mix and match as needed, or build your own.

Horizontal alignment

Change the horizontal alignment of your nav with flexbox utilities. By default, navs are left-aligned, but you can easily change them to center or right aligned.

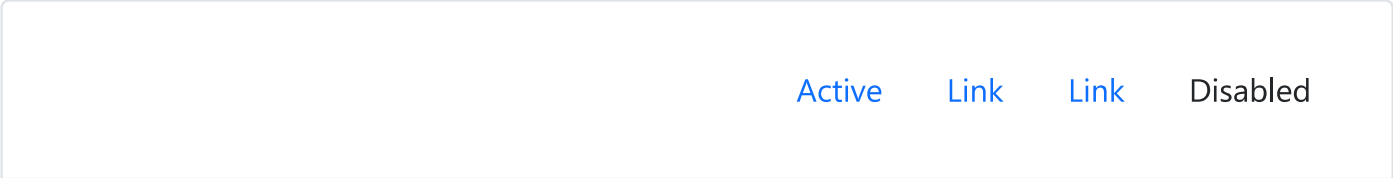
Centered with `.justify-content-center`:

Active Link Link Disabled

```
<ul class="nav justify-content-center">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
```

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

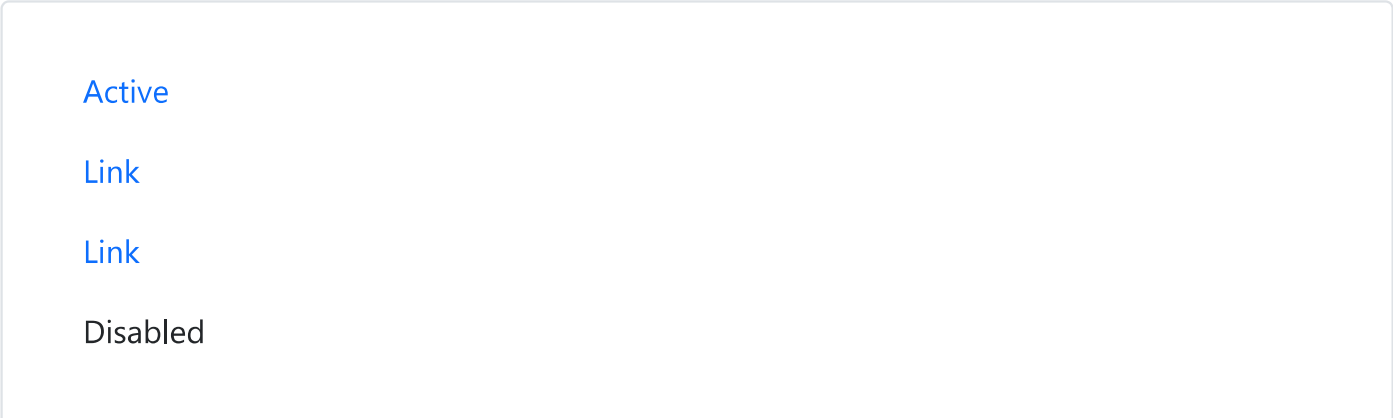
Right-aligned with `.justify-content-end`:




```
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Vertical

Stack your navigation by changing the flex item direction with the `.flex-column` utility. Need to stack them on some viewports but not others? Use the responsive versions (e.g., `.flex-sm-column`).




```
<ul class="nav flex-column">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
```

```
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled">Disabled</a>
    </li>
  </ul>
```

As always, vertical navigation is possible without `<ul>`s, too.

Active

Link

Link

Disabled

```
<nav class="nav flex-column">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

Tabs

Takes the basic nav from above and adds the `.nav-tabs` class to generate a tabbed interface. Use them to create tabbable regions with our tab JavaScript plugin.

Active Link Link Disabled

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
```

```
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled">Disabled</a>
    </li>
  </ul>
```

Pills

Take that same HTML, but use `.nav-pills` instead:

Active Link Link Disabled

```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

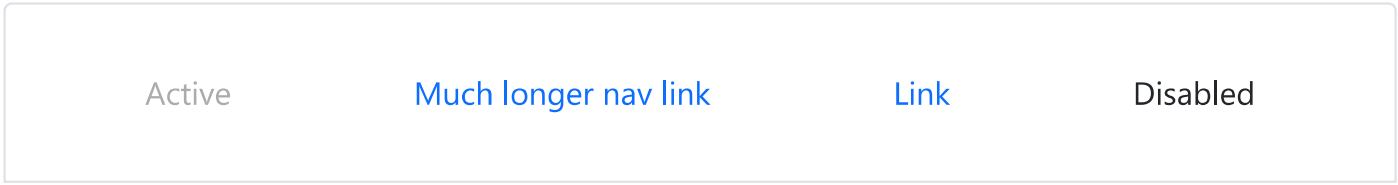
Fill and justify

Force your `.nav`'s contents to extend the full available width one of two modifier classes. To proportionately fill all available space with your `.nav-item`s, use `.nav-fill`. Notice that all horizontal space is occupied, but not every nav item has the same width.

Active Much longer nav link Link Disabled

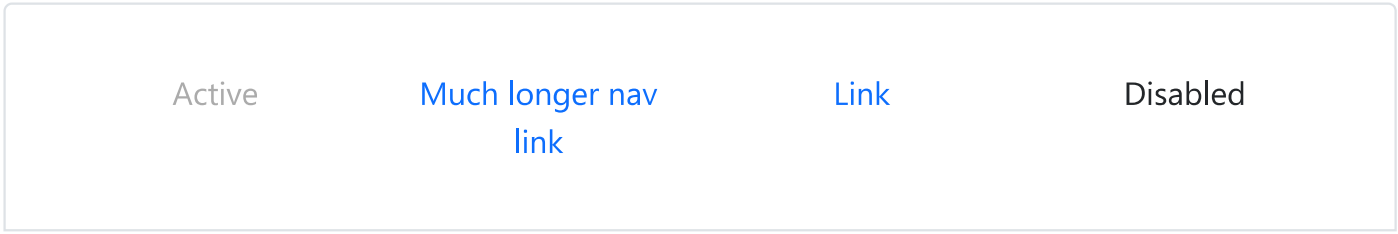
```
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Much longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

When using a `<nav>`-based navigation, you can safely omit `.nav-item` as only `.nav-link` is required for styling `<a>` elements.




```
<nav class="nav nav-pills nav-fill">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Much longer nav link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

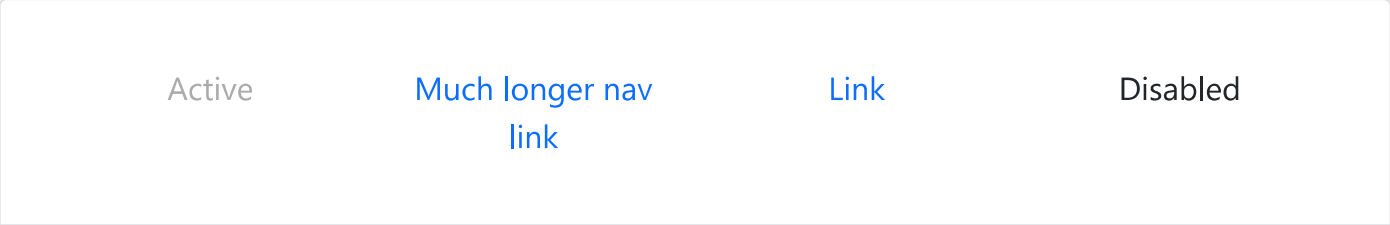
For equal-width elements, use `.nav-justified`. All horizontal space will be occupied by nav links, but unlike the `.nav-fill` above, every nav item will be the same width.




```
<ul class="nav nav-pills nav-justified">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Much longer nav link</a>
```

```
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled">Disabled</a>
    </li>
  </ul>
```

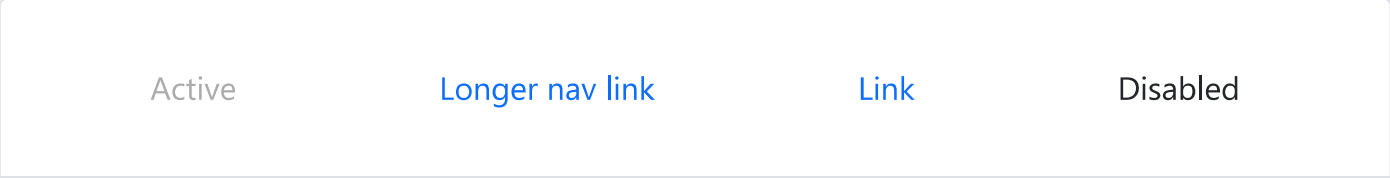
Similar to the `.nav-fill` example using a `<nav>`-based navigation.




```
<nav class="nav nav-pills nav-justified">
  <a class="nav-link active" aria-current="page" href="#">Active</a>
  <a class="nav-link" href="#">Much longer nav link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled">Disabled</a>
</nav>
```

Working with flex utilities

If you need responsive nav variations, consider using a series of flexbox utilities. While more verbose, these utilities offer greater customization across responsive breakpoints. In the example below, our nav will be stacked on the lowest breakpoint, then adapt to a horizontal layout that fills the available width starting from the small breakpoint.




```
<nav class="nav nav-pills flex-column flex-sm-row">
  <a class="flex-sm-fill text-sm-center nav-link active" aria-current="page" href="#
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Longer nav 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">Disabled</a>
</nav>
```

Regarding accessibility

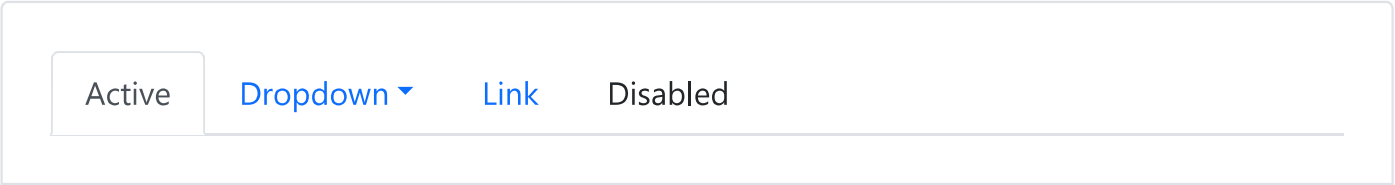
If you're using navs to provide a navigation bar, be sure to add a `role="navigation"` to the most logical parent container of the `<ul>`, or wrap a `<nav>` element around the whole navigation. Do not add the role to the `<ul>` itself, as this would prevent it from being announced as an actual list by assistive technologies.

Note that navigation bars, even if visually styled as tabs with the `.nav-tabs` class, should **not** be given `role="tablist"`, `role="tab"` or `role="tabpanel"` attributes. These are only appropriate for dynamic tabbed interfaces, as described in the WAI ARIA Authoring Practices. See JavaScript behavior for dynamic tabbed interfaces in this section for an example. The `aria-current` attribute is not necessary on dynamic tabbed interfaces since our JavaScript handles the selected state by adding `aria-selected="true"` on the active tab.

Using dropdowns

Add dropdown menus with a little extra HTML and the dropdowns JavaScript plugin.

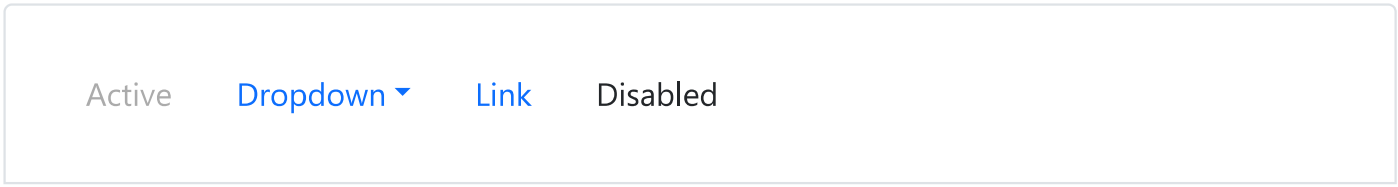
Tabs with dropdowns




```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="but
    <ul class="dropdown-menu">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
```

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

Pills with dropdowns




```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="but
    <ul class="dropdown-menu">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="#">Separated link</a></li>
    </ul>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

Sass

Variables

```
$nav-link-padding-y:                .5rem;
$nav-link-padding-x:                1rem;
$nav-link-font-size:                null;
$nav-link-font-weight:              null;
$nav-link-color:                    $link-color;
```

```
$nav-link-hover-color:              $link-hover-color;
$nav-link-transition:               color .15s ease-in-out, background-color .15s ea
$nav-link-disabled-color:           $gray-600;

$nav-tabs-border-color:             $gray-300;
$nav-tabs-border-width:             $border-width;
$nav-tabs-border-radius:            $border-radius;
$nav-tabs-link-hover-border-color:  $gray-200 $gray-200 $nav-tabs-border-color;
$nav-tabs-link-active-color:        $gray-700;
$nav-tabs-link-active-bg:           $body-bg;
$nav-tabs-link-active-border-color: $gray-300 $gray-300 $nav-tabs-link-active-bg;

$nav-pills-border-radius:           $border-radius;
$nav-pills-link-active-color:       $component-active-color;
```

JavaScript behavior

Use the tab JavaScript plugin—include it individually or through the compiled `bootstrap.js` file—to extend our navigational tabs and pills to create tabbable panes of local content.

Dynamic tabbed interfaces, as described in the WAI ARIA Authoring Practices, require `role="tablist"`, `role="tab"`, `role="tabpanel"`, and additional `aria-` attributes in order to convey their structure, functionality and current state to users of assistive technologies (such as screen readers). As a best practice, we recommend using `<button>` elements for the tabs, as these are controls that trigger a dynamic change, rather than links that navigate to a new page or location.

Note that dynamic tabbed interfaces should *not* contain dropdown menus, as this causes both usability and accessibility issues. From a usability perspective, the fact that the currently displayed tab's trigger element is not immediately visible (as it's inside the closed dropdown menu) can cause confusion. From an accessibility point of view, there is currently no sensible way to map this sort of construct to a standard WAI ARIA pattern, meaning that it cannot be easily made understandable to users of assistive technologies.

Home Profile Contact

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-targe
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-target="#
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="contact-tab" data-bs-toggle="tab" data-bs-target="#
  </li>
</ul>
<div class="tab-content" id="myTabContent">
  <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-t
  <div class="tab-pane fade" id="contact" role="tabpanel" aria-labelledby="contact-t
</div>
```

To help fit your needs, this works with `<ul>`-based markup, as shown above, or with any arbitrary "roll your own" markup. Note that if you're using `<nav>`, you shouldn't add `role="tablist"` directly to it, as this would override the element's native role as a navigation landmark. Instead, switch to an alternative element (in the example below, a simple `<div>`) and wrap the `<nav>` around it.

Home Profile Contact

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<nav>
  <div class="nav nav-tabs" id="nav-tab" role="tablist">
    <button class="nav-link active" id="nav-home-tab" data-bs-toggle="tab" data-bs-t
    <button class="nav-link" id="nav-profile-tab" data-bs-toggle="tab" data-bs-targe
    <button class="nav-link" id="nav-contact-tab" data-bs-toggle="tab" data-bs-targe
  </div>
</nav>
<div class="tab-content" id="nav-tabContent">
  <div class="tab-pane fade show active" id="nav-home" role="tabpanel" aria-labelled
  <div class="tab-pane fade" id="nav-profile" role="tabpanel" aria-labelledby="nav-p
  <div class="tab-pane fade" id="nav-contact" role="tabpanel" aria-labelledby="nav-c
```

```
</div>
```

The tabs plugin also works with pills.

Home Profile Contact

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<ul class="nav nav-pills mb-3" id="pills-tab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="pills-home-tab" data-bs-toggle="pill" data-b
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-profile-tab" data-bs-toggle="pill" data-bs-ta
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="pills-contact-tab" data-bs-toggle="pill" data-bs-ta
  </li>
</ul>
<div class="tab-content" id="pills-tabContent">
  <div class="tab-pane fade show active" id="pills-home" role="tabpanel" aria-labell
  <div class="tab-pane fade" id="pills-profile" role="tabpanel" aria-labelledby="pil
  <div class="tab-pane fade" id="pills-contact" role="tabpanel" aria-labelledby="pil
</div>
```

And with vertical pills.

Home

Profile

Messages

Settings

This is some placeholder content the Home tab's associated content. Clicking another tab will toggle the visibility of this one for the next. The tab JavaScript swaps classes to control the content visibility and styling. You can use it with tabs, pills, and any other `.nav`-powered navigation.

```
<div class="d-flex align-items-start">
  <div class="nav flex-column nav-pills me-3" id="v-pills-tab" role="tablist" aria-o
    <button class="nav-link active" id="v-pills-home-tab" data-bs-toggle="pill" data
    <button class="nav-link" id="v-pills-profile-tab" data-bs-toggle="pill" data-bs-
    <button class="nav-link" id="v-pills-messages-tab" data-bs-toggle="pill" data-bs
    <button class="nav-link" id="v-pills-settings-tab" data-bs-toggle="pill" data-bs
  </div>
  <div class="tab-content" id="v-pills-tabContent">
    <div class="tab-pane fade show active" id="v-pills-home" role="tabpanel" aria-la
    <div class="tab-pane fade" id="v-pills-profile" role="tabpanel" aria-labelledby=
    <div class="tab-pane fade" id="v-pills-messages" role="tabpanel" aria-labelledby
    <div class="tab-pane fade" id="v-pills-settings" role="tabpanel" aria-labelledby
  </div>
</div>
```

Using data attributes

You can activate a tab or pill navigation without writing any JavaScript by simply specifying `data-bs-toggle="tab"` or `data-bs-toggle="pill"` on an element. Use these data attributes on `.nav-tabs` or `.nav-pills`.

```
<!-- Nav tabs -->
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-targe
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-target="#
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="messages-tab" data-bs-toggle="tab" data-bs-target="
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-target="
  </li>
</ul>

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

```
</div>
```

Via JavaScript

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

```
var triggerTabList = [].slice.call(document.querySelectorAll('#myTab button'))
triggerTabList.forEach(function (triggerEl) {
  var tabTrigger = new bootstrap.Tab(triggerEl)

  triggerEl.addEventListener('click', function (event) {
    event.preventDefault()
    tabTrigger.show()
  })
})
```

You can activate individual tabs in several ways:

```
var triggerEl = document.querySelector('#myTab button[data-bs-target="#profile"]')
bootstrap.Tab.getInstance(triggerEl).show() // Select tab by name

var triggerFirstTabEl = document.querySelector('#myTab li:first-child button')
bootstrap.Tab.getInstance(triggerFirstTabEl).show() // Select first tab
```

Fade effect

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

```
<div class="tab-content">
  <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="
  <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-t
  <div class="tab-pane fade" id="messages" role="tabpanel" aria-labelledby="messages
  <div class="tab-pane fade" id="settings" role="tabpanel" aria-labelledby="settings
</div>
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

See our JavaScript documentation for more information.

constructor

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

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="home-tab" data-bs-toggle="tab" data-bs-targe
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="profile-tab" data-bs-toggle="tab" data-bs-target="#
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="messages-tab" data-bs-toggle="tab" data-bs-target="
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="settings-tab" data-bs-toggle="tab" data-bs-target="
  </li>
</ul>

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

<script>
  var firstTabEl = document.querySelector('#myTab li:last-child button')
  var firstTab = new bootstrap.Tab(firstTabEl)

  firstTab.show()
</script>
```

show

Selects the given tab and shows its associated pane. Any other tab that was previously selected becomes unselected and its associated pane is hidden. **Returns to the caller before the tab pane has actually been shown** (i.e. before the `shown.bs.tab` event occurs).

```
var someTabTriggerEl = document.querySelector('#someTabTrigger')
var tab = new bootstrap.Tab(someTabTriggerEl)

tab.show()
```

dispose

Destroys an element's tab.

getInstance

Static method which allows you to get the tab instance associated with a DOM element

```
var triggerEl = document.querySelector('#trigger')
var tab = bootstrap.Tab.getInstance(triggerEl) // Returns a Bootstrap tab instance
```

getOrCreateInstance

Static method which allows you to get the tab instance associated with a DOM element, or create a new one in case it wasn't initialized

```
var triggerEl = document.querySelector('#trigger')
var tab = bootstrap.Tab.getOrCreateInstance(triggerEl) // Returns a Bootstrap tab in
```

Events

When showing a new tab, the events fire in the following order:

1. `hide.bs.tab` (on the current active tab)
2. `show.bs.tab` (on the to-be-shown tab)
3. `hidden.bs.tab` (on the previous active tab, the same one as for the `hide.bs.tab` event)
4. `shown.bs.tab` (on the newly-active just-shown tab, the same one as for the `show.bs.tab` event)

If no tab was already active, then the `hide.bs.tab` and `hidden.bs.tab` events will not be fired.

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.
`hide.bs.tab`	This event fires when a new tab is to be shown (and thus the previous active tab is to be hidden). Use `event.target` and `event.relatedTarget` to target the current active tab and the new soon-to-be-active tab, respectively.
`hidden.bs.tab`	This event fires after a new tab is shown (and thus the previous active tab is hidden). Use `event.target` and `event.relatedTarget` to target the previous active tab and the new active tab, respectively.

```
var tabEl = document.querySelector('button[data-bs-toggle="tab"]')
tabEl.addEventListener('shown.bs.tab', function (event) {
  event.target // newly activated tab
  event.relatedTarget // previous active tab
})
```

Blog

Swag Store

Projects

Bootstrap 5

Bootstrap 4

Icons

RFS

npm starter

Community

Issues

Discussions

Corporate sponsors

Open Collective

Stack Overflow