Overview

Dropdowns are toggleable, contextual overlays for displaying lists of links and more. They're made interactive with the included Bootstrap dropdown JavaScript plugin. They're toggled by clicking, not by hovering; this is an intentional design decision.

Dropdowns are built on a third party library, Popper, which provides dynamic positioning and viewport detection. Be sure to include [popper.min.js]({{< param "cdn.popper" >}}) before Bootstrap's JavaScript or use `bootstrap.bundle.min.js` / `bootstrap.bundle.js` which contains Popper. Popper isn't used to position dropdowns in navbars though as dynamic positioning isn't required.

Accessibility

The WAI ARIA standard defines an actual `role="menu"` widget, but this is specific to application-like menus which trigger actions or functions. ARIA menus can only contain menu items, checkbox menu items, radio button menu items, radio button groups, and sub-menus.

Bootstrap's dropdowns, on the other hand, are designed to be generic and applicable to a variety of situations and markup structures. For instance, it is possible to create dropdowns that contain additional inputs and form controls, such as search fields or login forms. For this reason, Bootstrap does not expect (nor automatically add) any of the `role` and `aria-` attributes required for true ARIA menus. Authors will have to include these more specific attributes themselves.

However, Bootstrap does add built-in support for most standard keyboard menu interactions, such as the ability to move through individual `.dropdown-item` elements using the cursor keys and close the menu with the `ESC` key.

Examples

Wrap the dropdown's toggle (your button or link) and the dropdown menu within `.dropdown`, or another element that declares `position: relative;`. Dropdowns can be triggered from `<a>` or `<button>` elements to better fit your potential needs. The examples shown here use semantic `<ul>` elements where appropriate, but custom markup is supported.

Single button

Any single `.btn` can be turned into a dropdown toggle with some markup changes. Here's how you can put them to work with either `<button>` elements:

{{< example >}}

Dropdown button

- Action
- Another action
- Something else here

{{< /example >}}

And with `<a>` elements:

{{< example >}}

Dropdown link

- Action
- Another action

- Something else here

{{< /example >}}

The best part is you can do this with any button variant, too:

Primary
- Action
- Another action
- Something else here
- ---
- Separated link

Secondary
- Action
- Another action
- Something else here
- ---
- Separated link

Success
- Action
- Another action
- Something else here
- ---
- Separated link

Info
- Action
- Another action
- Something else here
- ---
- Separated link

Warning
- Action
- Another action
- Something else here
- ---
- Separated link

Danger
- Action
- Another action
- Something else here
- ---
- Separated link

```
<!-- Example single danger button -->
<div class="btn-group">
```

```
  <button type="button" class="btn btn-danger dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Action
  </button>
  <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>
</div>
```

Split button

Similarly, create split button dropdowns with virtually the same markup as single button dropdowns, but with the addition of `.dropdown-toggle-split` for proper spacing around the dropdown caret.

We use this extra class to reduce the horizontal `padding` on either side of the caret by 25% and remove the `margin-left` that's added for regular button dropdowns. Those extra changes keep the caret centered in the split button and provide a more appropriately sized hit area next to the main button.

Primary Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

Secondary Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

Success Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

Info Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

Warning Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

Danger Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

```
<!-- Example split danger button -->
<div class="btn-group">
  <button type="button" class="btn btn-danger">Action</button>
  <button type="button" class="btn btn-danger dropdown-toggle dropdown-toggle-split"
data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <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>
</div>
```

Sizing

Button dropdowns work with buttons of all sizes, including default and split dropdown buttons.

Large button

- Action
- Another action
- Something else here
- ---
- Separated link

Large split button Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

```
<!-- Large button groups (default and split) -->
<div class="btn-group">
  <button class="btn btn-secondary btn-lg dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-expanded="false">
    Large button
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
<div class="btn-group">
  <button class="btn btn-secondary btn-lg" type="button">
    Large split button
  </button>
  <button type="button" class="btn btn-lg btn-secondary dropdown-toggle dropdown-toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

Small button

- Action
- Another action
- Something else here
- ---
- Separated link

Small split button Toggle Dropdown

- Action
- Another action
- Something else here
- ---
- Separated link

```
<div class="btn-group">
  <button class="btn btn-secondary btn-sm dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-expanded="false">
    Small button
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
<div class="btn-group">
  <button class="btn btn-secondary btn-sm" type="button">
    Small split button
```

```
  </button>
  <button type="button" class="btn btn-sm btn-secondary dropdown-toggle dropdown-
toggle-split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

Dark dropdowns

Opt into darker dropdowns to match a dark navbar or custom style by adding `.dropdown-menu-dark` onto an existing `.dropdown-menu`. No changes are required to the dropdown items.

{{< example >}}

Dropdown button

- Action
- Another action
- Something else here
- ---
- Separated link

{{< /example >}}

And putting it to use in a navbar:

{{< example >}}

Navbar

- Dropdown
  - Action
  - Another action
  - Something else here

{{< /example >}}

Directions

{{< callout info >}}

RTL

Directions are mirrored when using Bootstrap in RTL, meaning `.dropstart` will appear on the right side. {{< /callout >}}

Dropup

Trigger dropdown menus above elements by adding `.dropup` to the parent element.

Dropup

- Action
- Another action

- Something else here
-

- Separated link

Split dropup Toggle Dropdown

- Action
- Another action
- Something else here
-

- Separated link

```
<!-- Default dropup button -->
<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropup
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropup button -->
<div class="btn-group dropup">
  <button type="button" class="btn btn-secondary">
    Split dropup
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-
split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropdown</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

Dropright

Trigger dropdown menus at the right of the elements by adding `.dropend` to the parent element.

Dropright

- Action
- Another action
- Something else here
-

- Separated link

Split dropend Toggle Dropright

- Action

- Another action
- Something else here
- ---
- Separated link

```
<!-- Default dropend button -->
<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropright
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropend button -->
<div class="btn-group dropend">
  <button type="button" class="btn btn-secondary">
    Split dropend
  </button>
  <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-
split" data-bs-toggle="dropdown" aria-expanded="false">
    <span class="visually-hidden">Toggle Dropright</span>
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>
```

Dropleft

Trigger dropdown menus at the left of the elements by adding `.dropstart` to the parent element.

Dropleft

- Action
- Another action
- Something else here
- ---
- Separated link

Toggle Dropleft

- Action
- Another action
- Something else here
- ---
- Separated link

Split dropstart

```
<!-- Default dropstart button -->
<div class="btn-group dropstart">
  <button type="button" class="btn btn-secondary dropdown-toggle" data-bs-
toggle="dropdown" aria-expanded="false">
    Dropstart
  </button>
  <ul class="dropdown-menu">
    <!-- Dropdown menu links -->
  </ul>
</div>

<!-- Split dropstart button -->
<div class="btn-group">
  <div class="btn-group dropstart" role="group">
    <button type="button" class="btn btn-secondary dropdown-toggle dropdown-toggle-
split" data-bs-toggle="dropdown" aria-expanded="false">
      <span class="visually-hidden">Toggle Dropstart</span>
    </button>
    <ul class="dropdown-menu">
      <!-- Dropdown menu links -->
    </ul>
  </div>
  <button type="button" class="btn btn-secondary">
    Split dropstart
  </button>
</div>
```

Menu items

You can use `<a>` or `<button>` elements as dropdown items.

{{< example >}}

Dropdown

- Action
- Another action
- Something else here

{{< /example >}}

You can also create non-interactive dropdown items with `.dropdown-item-text`. Feel free to style further with custom CSS or text utilities.

{{< example >}}

- Dropdown item text
- Action
- Another action
- Something else here

{{< /example >}}

Active

Add `.active` to items in the dropdown to **style them as active**. To convey the active state to assistive technologies, use the `aria-current` attribute — using the `page` value for the current page, or `true` for the current item in a set.

{{< example >}}

- Regular link
- Active link
- Another link

{{< /example >}}

Disabled

Add `.disabled` to items in the dropdown to **style them as disabled**.

{{< example >}}

- Regular link
- Disabled link
- Another link

{{< /example >}}

Menu alignment

By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. You can change this with the directional `.drop*` classes, but you can also control them with additional modifier classes.

Add `.dropdown-menu-end` to a `.dropdown-menu` to right align the dropdown menu. Directions are mirrored when using Bootstrap in RTL, meaning `.dropdown-menu-end` will appear on the left side.

{{< callout info >}} **Heads up!** Dropdowns are positioned thanks to Popper except when they are contained in a navbar. {{< /callout >}}

{{< example >}}

Right-aligned menu example

- Action
- Another action
- Something else here

{{< /example >}}

Responsive alignment

If you want to use responsive alignment, disable dynamic positioning by adding the `data-bs-display="static"` attribute and use the responsive variation classes.

To align **right** the dropdown menu with the given breakpoint or larger, add `.dropdown-menu{-sm|-md|-lg|-xl|-xxl}-end`.

{{< example >}}

Left-aligned but right aligned when large screen

- Action
- Another action

- Something else here

{{< /example >}}

To align **left** the dropdown menu with the given breakpoint or larger, add `.dropdown-menu-end` and `.dropdown-menu{-sm|-md|-lg|-xl|-xxl}-start`.

{{< example >}}

Right-aligned but left aligned when large screen

- Action
- Another action
- Something else here

{{< /example >}}

Note that you don't need to add a `data-bs-display="static"` attribute to dropdown buttons in navbars, since Popper isn't used in navbars.

Alignment options

Taking most of the options shown above, here's a small kitchen sink demo of various dropdown alignment options in one place.

{{< example >}}

Dropdown

- Menu item
- Menu item
- Menu item

Right-aligned menu

- Menu item
- Menu item
- Menu item

Left-aligned, right-aligned lg

- Menu item
- Menu item
- Menu item

Right-aligned, left-aligned lg

- Menu item
- Menu item
- Menu item

Dropstart

- Menu item
- Menu item
- Menu item

Dropend

- Menu item
- Menu item
- Menu item

Dropup

- Menu item
- Menu item
- Menu item

{{< /example >}}

Menu content

Headers

Add a header to label sections of actions in any dropdown menu.

{{< example >}}

- **DROPDOWN HEADER**
- Action
- Another action

{{< /example >}}

Dividers

Separate groups of related menu items with a divider.

{{< example >}}

- Action
- Another action
- Something else here
- ---
- Separated link

{{< /example >}}

Text

Place any freeform text within a dropdown menu with text and use [spacing utilities]({{< docsref "/utilities/spacing" >}}). Note that you'll likely need additional sizing styles to constrain the menu width.

{{< example >}}

Some example text that's free-flowing
within the dropdown menu.

And this is more example text.

{{< /example >}}

Forms

Put a form within a dropdown menu, or make it into a dropdown menu, and use [margin or padding utilities]({{< docsref "/utilities/spacing" >}}) to give it the negative space you require.

{{< example >}}

Email address email@example.com
Password Password
Remember me
Sign in

New around here? Sign up Forgot password?

{{< /example >}}

{{< example >}}

Email address email@example.com

Password Password

Remember me

Sign in

{{< /example >}}

Dropdown options

Use `data-bs-offset` or `data-bs-reference` to change the location of the dropdown.

{{< example >}}

Offset

- Action
- Another action
- Something else here

Reference Toggle Dropdown

- Action
- Another action
- Something else here
-

- Separated link

{{< /example >}}

Auto close behavior

By default, the dropdown menu is closed when clicking inside or outside the dropdown menu. You can use the `autoClose` option to change this behavior of the dropdown.

{{< example >}}

Default dropdown

- Menu item
- Menu item
- Menu item

Clickable outside

- Menu item
- Menu item
- Menu item

Clickable inside

- Menu item
- Menu item
- Menu item

Manual close

- Menu item
- Menu item

- Menu item

{{< /example >}}

Sass

Variables

Variables for all dropdowns:

{{< scss-docs name="dropdown-variables" file="scss/_variables.scss" >}}

Variables for the dark dropdown:

{{< scss-docs name="dropdown-dark-variables" file="scss/_variables.scss" >}}

Variables for the CSS-based carets that indicate a dropdown's interactivity:

{{< scss-docs name="caret-variables" file="scss/_variables.scss" >}}

Mixins

Mixins are used to generate the CSS-based carets and can be found in `scss/mixins/_caret.scss`.

{{< scss-docs name="caret-mixins" file="scss/mixins/_caret.scss" >}}

Usage

Via data attributes or JavaScript, the dropdown plugin toggles hidden content (dropdown menus) by toggling the `.show` class on the parent `.dropdown-menu`. The `data-bs-toggle="dropdown"` attribute is relied on for closing dropdown menus at an application level, so it's a good idea to always use it.

{{< callout info >}} On touch-enabled devices, opening a dropdown adds empty `mouseover` handlers to the immediate children of the `<body>` element. This admittedly ugly hack is necessary to work around a quirk in iOS' event delegation, which would otherwise prevent a tap anywhere outside of the dropdown from triggering the code that closes the dropdown. Once the dropdown is closed, these additional empty `mouseover` handlers are removed. {{< /callout >}}

Via data attributes

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

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

Via JavaScript

Call the dropdowns via JavaScript:

```
var dropdownElementList = [].slice.call(document.querySelectorAll('.dropdown-
toggle'))
var dropdownList = dropdownElementList.map(function (dropdownToggleEl) {
  return new bootstrap.Dropdown(dropdownToggleEl)
})
```

{{< callout info >}}

`data-bs-toggle="dropdown"` still required

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

Options

Options can be passed via data attributes or JavaScript. For data attributes, append the option name to `data-bs-`, as in `data-bs-offset=""`. Make sure to change the case type of the option name from camelCase to kebab-case when passing the options via data attributes. For example, instead of using `data-bs-autoClose="false"`, use `data-bs-auto-close="false"`.

Name	Type	Default	Description
`boundary`	string | element	`'clippingParents'`	Overflow constraint boundary of the dropdown menu (applies only to Popper's preventOverflow modifier). By default it's `'clippingParents'` and can accept an HTMLElement reference (via JavaScript only). For more information refer to Popper's detectOverflow docs.
`reference`	string | element | object	`'toggle'`	Reference element of the dropdown menu. Accepts the values of `'toggle'`, `'parent'`, an HTMLElement reference or an object providing `getBoundingClientRect`. For more information refer to Popper's constructor docs and virtual element docs.
`display`	string	`'dynamic'`	By default, we use Popper for dynamic positioning. Disable this with `static`.
`offset`	array | string | function	`[0, 2]`	Offset of the dropdown relative to its target. You can pass a string in data attributes with comma separated values like: `data-bs-offset="10,20"` When a function is used to determine the offset, it is called with an object containing the popper placement, the reference, and popper rects as its first argument. The triggering element DOM node is passed as the second argument. The function must return an array

			with two numbers: `[skidding, distance]`. For more information refer to Popper's offset docs.
`autoClose`	boolean | string	`true`	Configure the auto close behavior of the dropdown: • `true` - the dropdown will be closed by clicking outside or inside the dropdown menu. • `false` - the dropdown will be closed by clicking the toggle button and manually calling `hide` or `toggle` method. (Also will not be closed by pressing `esc` key) • `'inside'` - the dropdown will be closed (only) by clicking inside the dropdown menu. • `'outside'` - the dropdown will be closed (only) by clicking outside the dropdown menu.
`popperConfig`	null | object | function	`null`	To change Bootstrap's default Popper config, see Popper's configuration. When a function is used to create the Popper configuration, it's called with an object that contains the Bootstrap's default Popper configuration. It helps you use and merge the default with your own configuration. The function must return a configuration object for Popper.

Using function with `popperConfig`

```
var dropdown = new bootstrap.Dropdown(element, {
  popperConfig: function (defaultBsPopperConfig) {
    // var newPopperConfig = {...}
    // use defaultBsPopperConfig if needed...
    // return newPopperConfig
  }
})
```

Methods

Method	Description

`toggle`	Toggles the dropdown menu of a given navbar or tabbed navigation.
`show`	Shows the dropdown menu of a given navbar or tabbed navigation.
`hide`	Hides the dropdown menu of a given navbar or tabbed navigation.
`update`	Updates the position of an element's dropdown.
`dispose`	Destroys an element's dropdown. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the dropdown instance associated to a DOM element, you can use it like this: `bootstrap.Dropdown.getInstance(element)`
`getOrCreateInstance`	Static method which returns a dropdown instance associated to a DOM element or create a new one in case it wasn't initialized. You can use it like this: `bootstrap.Dropdown.getOrCreateInstance(element)`

Events

All dropdown events are fired at the toggling element and then bubbled up. So you can also add event listeners on the `.dropdown-menu` 's parent element. `hide.bs.dropdown` and `hidden.bs.dropdown` events have a `clickEvent` property (only when the original Event type is `click`) that contains an Event Object for the click event.

Method	Description
`show.bs.dropdown`	Fires immediately when the show instance method is called.
`shown.bs.dropdown`	Fired when the dropdown has been made visible to the user and CSS transitions have completed.
`hide.bs.dropdown`	Fires immediately when the hide instance method has been called.
`hidden.bs.dropdown`	Fired when the dropdown has finished being hidden from the user and CSS transitions have completed.

```
var myDropdown = document.getElementById('myDropdown')
myDropdown.addEventListener('show.bs.dropdown', function () {
  // do something...
})
```