

There's a newer version of Bootstrap!

Ctrl + /

v5.0 ▾



[View on GitHub](#)

# Modal

Use Bootstrap's JavaScript modal plugin to add dialogs to your site for lightboxes, user notifications, or completely custom content.



Your new development career awaits. Check out the latest listings.

ads via Carbon

## On this page

---

How it works

Examples

Modal components

Live demo

Static backdrop

Scrolling long content

Vertically centered

Tooltips and popovers

Using the grid

Varying modal content

Toggle between modals

Change animation

Remove animation

Dynamic heights

Accessibility

Embedding YouTube videos

Optional sizes

Fullscreen Modal

Sass

Variables

Loop

Usage

Via data attributes

Via JavaScript

Options

Methods

Passing options

toggle

show

hide

handleUpdate

dispose

getInstance

getOrCreateInstance

Events

## How it works

Before getting started with Bootstrap's modal component, be sure to read the following as our menu options have recently changed.

- Modals are built with HTML, CSS, and JavaScript. They're positioned over everything else in the document and remove scroll from the `<body>` so that modal content scrolls instead.
- Clicking on the modal "backdrop" will automatically close the modal.
- Bootstrap only supports one modal window at a time. Nested modals aren't supported as we believe them to be poor user experiences.
- Modals use `position: fixed`, which can sometimes be a bit particular about its rendering. Whenever possible, place your modal HTML in a top-level position to avoid potential interference from other elements. You'll likely run into issues when nesting a `.modal` within another fixed element.
- Once again, due to `position: fixed`, there are some caveats with using modals on mobile devices. [See our browser support docs](#) for details.
- Due to how HTML5 defines its semantics, [the autofocus HTML attribute](#) has no effect in Bootstrap modals. To achieve the same effect, use some custom JavaScript:

```
var myModal = document.getElementById('myModal')
var myInput = document.getElementById('myInput')

myModal.addEventListener('shown.bs.modal', function () {
  myInput.focus()
})
```

}}

The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the [reduced motion section of our accessibility documentation](#).

Keep reading for demos and usage guidelines.

## Examples

### Modal components

Below is a *static* modal example (meaning its `position` and `display` have been overridden). Included are the modal header, modal body (required for `padding`), and modal footer (optional). We ask that you include modal headers with dismiss actions whenever possible, or provide another explicit dismiss action.

Modal title

Modal body text goes here.

CloseSave changes

```
<div class="modal" tabindex="-1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="
      </div>
      <div class="modal-body">
        <p>Modal body text goes here.</p>
      </div>
      <div class="modal-footer">
```

```

        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Clos
        <button type="button" class="btn btn-primary">Save changes</button>
    </div>
</div>
</div>
...

```

## Live demo

Toggle a working modal demo by clicking the button below. It will slide down and fade in from the top of the page.

Launch demo modal

```

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

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModa
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Clos
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>

```

## Static backdrop

When backdrop is set to static, the modal will not close when clicking outside it. Click the button below to try it.

Launch static backdrop modal

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target=
  Launch static backdrop modal
</button>

<!-- Modal -->
<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboa
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Clos
        <button type="button" class="btn btn-primary">Understood</button>
      </div>
    </div>
  </div>
</div>
```

## Scrolling long content

When modals become too long for the user's viewport or device, they scroll independent of the page itself. Try the demo below to see what we mean.

Launch demo modal

You can also create a scrollable modal that allows scroll the modal body by adding `.modal-dialog-scrollable` to `.modal-dialog`.

Launch demo modal

```
<!-- Scrollable modal -->
<div class="modal-dialog modal-dialog-scrollable">
  ...
</div>
```

## Vertically centered

Add `.modal-dialog-centered` to `.modal-dialog` to vertically center the modal.

Vertically centered modal

Vertically centered scrollable modal

```
<!-- Vertically centered modal -->
<div class="modal-dialog modal-dialog-centered">
  ...
</div>

<!-- Vertically centered scrollable modal -->
<div class="modal-dialog modal-dialog-centered modal-dialog-scrollable">
  ...
</div>
```

## Tooltips and popovers

[Tooltips](#) and [popovers](#) can be placed within modals as needed. When modals are closed, any tooltips and popovers within are also automatically dismissed.

Launch demo modal

```
<div class="modal-body">
  <h5>Popover in a modal</h5>
  <p>This <a href="#" role="button" class="btn btn-secondary popover-test" title="Po
  <hr>
  <h5>Tooltips in a modal</h5>
```

```
<p><a href="#" class="tooltip-test" title="Tooltip">This link</a> and <a href="#"
```

## Using the grid

Utilize the Bootstrap grid system within a modal by nesting `.container-fluid` within the `.modal-body`. Then, use the normal grid system classes as you would anywhere else.

Launch demo modal

```
<div class="modal-body">
  <div class="container-fluid">
    <div class="row">
      <div class="col-md-4">.col-md-4</div>
      <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-md-3 ms-auto">.col-md-3 .ms-auto</div>
      <div class="col-md-2 ms-auto">.col-md-2 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-md-6 ms-auto">.col-md-6 .ms-auto</div>
    </div>
    <div class="row">
      <div class="col-sm-9">
        Level 1: .col-sm-9
        <div class="row">
          <div class="col-8 col-sm-6">
            Level 2: .col-8 .col-sm-6
          </div>
          <div class="col-4 col-sm-6">
            Level 2: .col-4 .col-sm-6
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

## Varying modal content

Have a bunch of buttons that all trigger the same modal with slightly different contents? Use `event.relatedTarget` and [HTML data-bs-\\* attributes](#) to vary the contents of the modal depending on which button was clicked.

Below is a live demo followed by example HTML and JavaScript. For more information, [read the modal events docs](#) for details on `relatedTarget`.

Open modal for @mdo

Open modal for @fat

Open modal for @getbootstrap

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target=
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target=
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target=

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModa
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">New message</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="
      </div>
      <div class="modal-body">
        <form>
          <div class="mb-3">
            <label for="recipient-name" class="col-form-label">Recipient:</label>
            <input type="text" class="form-control" id="recipient-name">
          </div>
          <div class="mb-3">
            <label for="message-text" class="col-form-label">Message:</label>
            <textarea class="form-control" id="message-text"></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Clos
        <button type="button" class="btn btn-primary">Send message</button>
      </div>
    </div>
  </div>
</div>
```

```
var exampleModal = document.getElementById('exampleModal')
exampleModal.addEventListener('show.bs.modal', function (event) {
```



```

// Button that triggered the modal
var button = event.relatedTarget
// Extract info from data-bs-* attributes
var recipient = button.getAttribute('data-bs-whatever')
// If necessary, you could initiate an AJAX request here
// and then do the updating in a callback.
//
// Update the modal's content.
var modalTitle = exampleModal.querySelector('.modal-title')
var modalBodyInput = exampleModal.querySelector('.modal-body input')

modalTitle.textContent = 'New message to ' + recipient
modalBodyInput.value = recipient
})

```

## Toggle between modals

Toggle between multiple modals with some clever placement of the `data-bs-target` and `data-bs-toggle` attributes. For example, you could toggle a password reset modal from within an already open sign in modal. **Please note multiple modals cannot be open at the same time**—this method simply toggles between two separate modals.

Open first modal

```

<div class="modal fade" id="exampleModalToggle" aria-hidden="true" aria-labelledby="
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalToggleLabel">Modal 1</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="
      </div>
      <div class="modal-body">
        Show a second modal and hide this one with the button below.
      </div>
      <div class="modal-footer">
        <button class="btn btn-primary" data-bs-target="#exampleModalToggle2" data-b
      </div>
    </div>
  </div>
</div>
<div class="modal fade" id="exampleModalToggle2" aria-hidden="true" aria-labelledby=
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">

```

```

    <h5 class="modal-title" id="exampleModalToggleLabel2">Modal 2</h5>
    <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="
</div>
<div class="modal-body">
    Hide this modal and show the first with the button below.
</div>
<div class="modal-footer">
    <button class="btn btn-primary" data-bs-target="#exampleModalToggle" data-bs
</div>
</div>
</div>
</div>

```

## Change animation

The `$modal-fade-transform` variable determines the transform state of `.modal-dialog` before the modal fade-in animation, the `$modal-show-transform` variable determines the transform of `.modal-dialog` at the end of the modal fade-in animation.

If you want for example a zoom-in animation, you can set `$modal-fade-transform: scale(.8)`.

## Remove animation

For modals that simply appear rather than fade in to view, remove the `.fade` class from your modal markup.

```

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

```

## Dynamic heights

If the height of a modal changes while it is open, you should call `myModal.handleUpdate()` to readjust the modal's position in case a scrollbar appears.

## Accessibility

Be sure to add `aria-labelledby="..."`, referencing the modal title, to `.modal`. Additionally, you may give a description of your modal dialog with `aria-describedby` on `.modal`. Note that you don't need to add `role="dialog"` since we already add it via JavaScript.

# Embedding YouTube videos

Embedding YouTube videos in modals requires additional JavaScript not in Bootstrap to automatically stop playback and more. [See this helpful Stack Overflow post](#) for more information.

## Optional sizes

Modals have three optional sizes, available via modifier classes to be placed on a `.modal-dialog`. These sizes kick in at certain breakpoints to avoid horizontal scrollbars on narrower viewports.

Size	Class	Modal max-width
Small	<code>.modal-sm</code>	300px
Default	None	500px
Large	<code>.modal-lg</code>	800px
Extra large	<code>.modal-xl</code>	1140px

Our default modal without modifier class constitutes the “medium” size modal.

Extra large modal

Large modal

Small modal

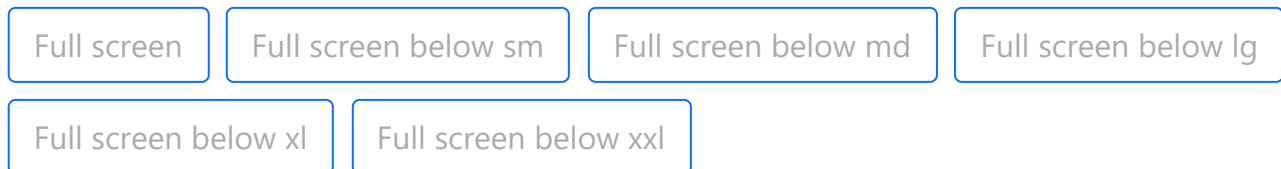
```
<div class="modal-dialog modal-xl">...</div>
<div class="modal-dialog modal-lg">...</div>
<div class="modal-dialog modal-sm">...</div>
```

## Fullscreen Modal

Another override is the option to pop up a modal that covers the user viewport, available via modifier classes that are placed on a `.modal-dialog`.

Class	Availability
<code>.modal-fullscreen</code>	Always
<code>.modal-fullscreen-sm-down</code>	Below 576px
<code>.modal-fullscreen-md-down</code>	Below 768px

Class	Availability
<code>.modal-fullscreen-lg-down</code>	Below 992px
<code>.modal-fullscreen-xl-down</code>	Below 1200px
<code>.modal-fullscreen-xxl-down</code>	Below 1400px



```
<!-- Full screen modal -->
<div class="modal-dialog modal-fullscreen-sm-down">
  ...
</div>
```

## Sass

## Variables

```
$modal-inner-padding: $spacer;

$modal-footer-margin-between: .5rem;

$modal-dialog-margin: .5rem;
$modal-dialog-margin-y-sm-up: 1.75rem;

$modal-title-line-height: $line-height-base;

$modal-content-color: null;
$modal-content-bg: $white;
$modal-content-border-color: rgba($black, .2);
$modal-content-border-width: $border-width;
$modal-content-border-radius: $border-radius-lg;
$modal-content-inner-border-radius: subtract($modal-content-border-radius, $modal-co
$modal-content-box-shadow-xs: $box-shadow-sm;
$modal-content-box-shadow-sm-up: $box-shadow;

$modal-backdrop-bg: $black;
```

```

$modal-backdrop-opacity: .5;
$modal-header-border-color: $border-color;
$modal-footer-border-color: $modal-header-border-color;
$modal-header-border-width: $modal-content-border-width;
$modal-footer-border-width: $modal-header-border-width;
$modal-header-padding-y: $modal-inner-padding;
$modal-header-padding-x: $modal-inner-padding;
$modal-header-padding: $modal-header-padding-y $modal-header-padding-x;

$modal-sm: 300px;
$modal-md: 500px;
$modal-lg: 800px;
$modal-xl: 1140px;

$modal-fade-transform: translate(0, -50px);
$modal-show-transform: none;
$modal-transition: transform .3s ease-out;

```

## Loop

[Responsive fullscreen modals](#) are generated via the `$breakpoints` map and a loop in `scss/_modal.scss`.

```

@each $breakpoint in map-keys($grid-breakpoints) {
  $infix: breakpoint-infix($breakpoint, $grid-breakpoints);
  $postfix: if($infix != "", $infix + "-down", "");

  @include media-breakpoint-down($breakpoint) {
    .modal-fullscreen#{$postfix} {
      width: 100vw;
      max-width: none;
      height: 100%;
      margin: 0;

      .modal-content {
        height: 100%;
        border: 0;
        @include border-radius(0);
      }

      .modal-header {
        @include border-radius(0);
      }

      .modal-body {
        overflow-y: auto;
      }
    }
  }
}

```

```

    }

    .modal-footer {
      @include border-radius(0);
    }
  }
}
}

```

## Usage

The modal plugin toggles your hidden content on demand, via data attributes or JavaScript. It also overrides default scrolling behavior and generates a `.modal-backdrop` to provide a click area for dismissing shown modals when clicking outside the modal.

### Via data attributes

Activate a modal without writing JavaScript. Set `data-bs-toggle="modal"` on a controller element, like a button, along with a `data-bs-target="#foo"` or `href="#foo"` to target a specific modal to toggle.

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

### Via JavaScript

Create a modal with a single line of JavaScript:

```
var myModal = new bootstrap.Modal(document.getElementById('myModal'), options)
```

## Options

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

Name	Type	Default	Description
<code>backdrop</code>	boolean or the	<code>true</code>	Includes a modal-backdrop element. Alternatively, specify <code>static</code> for a backdrop which doesn't close the modal on click.

Name	Type	Default	Description
	string 'static'		
keyboard	boolean	true	Closes the modal when escape key is pressed
focus	boolean	true	Puts the focus on the modal when initialized.

## 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.](#)

## Passing options

Activates your content as a modal. Accepts an optional options **object**.

```
var myModal = new bootstrap.Modal(document.getElementById('myModal'), {
  keyboard: false
})
```

## toggle

Manually toggles a modal. **Returns to the caller before the modal has actually been shown or hidden** (i.e. before the `shown.bs.modal` or `hidden.bs.modal` event occurs).

```
myModal.toggle()
```

## show

Manually opens a modal. **Returns to the caller before the modal has actually been shown** (i.e. before the `shown.bs.modal` event occurs).

```
myModal.show()
```

Also, you can pass a DOM element as an argument that can be received in the modal events (as the `relatedTarget` property).

```
var modalToggle = document.getElementById('toggleMyModal') // relatedTarget
myModal.show(modalToggle)
```

## hide

Manually hides a modal. **Returns to the caller before the modal has actually been hidden** (i.e. before the `hidden.bs.modal` event occurs).

```
myModal.hide()
```

## handleUpdate

Manually readjust the modal's position if the height of a modal changes while it is open (i.e. in case a scrollbar appears).

```
myModal.handleUpdate()
```

## dispose


Destroys an element's modal. (Removes stored data on the DOM element)

```
myModal.dispose()
```

## getInstance

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

```
var myModalEl = document.getElementById('myModal')
var modal = bootstrap.Modal.getInstance(myModalEl) // Returns a Bootstrap modal inst
```



## getOrCreateInstance

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



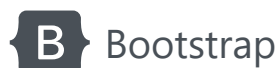
```
var myModalEl = document.querySelector('#myModal')
var modal = bootstrap.Modal.getOrCreateInstance(myModalEl) // Returns a Bootstrap mo
```

## Events

Bootstrap's modal class exposes a few events for hooking into modal functionality. All modal events are fired at the modal itself (i.e. at the `<div class="modal">`).

Event type	Description
<code>show.bs.modal</code>	This event fires immediately when the <code>show</code> instance method is called. If caused by a click, the clicked element is available as the <code>relatedTarget</code> property of the event.
<code>shown.bs.modal</code>	This event is fired when the modal has been made visible to the user (will wait for CSS transitions to complete). If caused by a click, the clicked element is available as the <code>relatedTarget</code> property of the event.
<code>hide.bs.modal</code>	This event is fired immediately when the <code>hide</code> instance method has been called.
<code>hidden.bs.modal</code>	This event is fired when the modal has finished being hidden from the user (will wait for CSS transitions to complete).
<code>hidePrevented.bs.modal</code>	This event is fired when the modal is shown, its backdrop is <code>static</code> and a click outside the modal or an escape key press is performed with the keyboard option or <code>data-bs-keyboard</code> set to <code>false</code> .

```
var myModalEl = document.getElementById('myModal')
myModalEl.addEventListener('hidden.bs.modal', function (event) {
  // do something...
})
```



Designed and built with all the love in the world by the Bootstrap team with the help of our contributors.

Code licensed MIT, docs CC BY 3.0.

Currently v5.0.2.

## Links

[Home](#)

[Docs](#)

[Examples](#)

[Themes](#)

[Blog](#)

[Swag Store](#)

## Projects

[Bootstrap 5](#)

[Bootstrap 4](#)

[Icons](#)

[RFS](#)

[npm starter](#)

## Guides

[Getting started](#)

[Starter template](#)

[Webpack](#)

[Parcel](#)

## Community

[Issues](#)

[Discussions](#)

[Corporate sponsors](#)

[Open Collective](#)

[Stack Overflow](#)