There's a newer version of Bootstrap!

Search docs... Ctrl + /

v5.0

View on GitHub

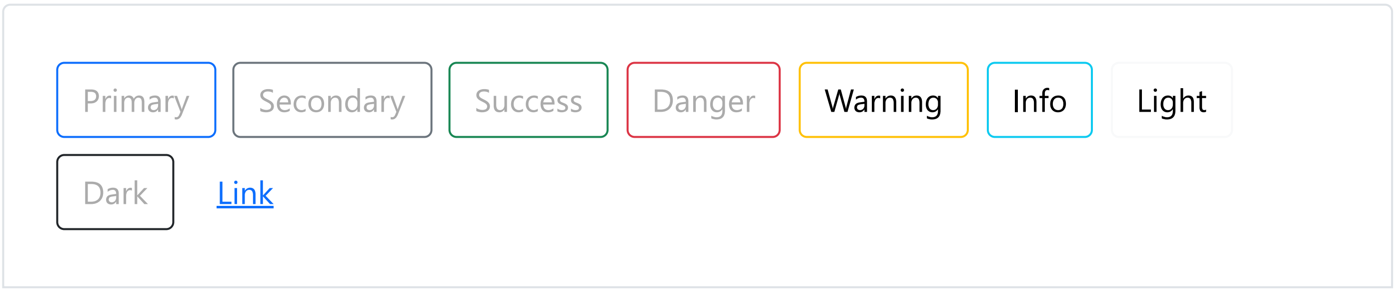
Buttons

Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

On this page

Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.



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

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

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the `.visually-hidden` class.

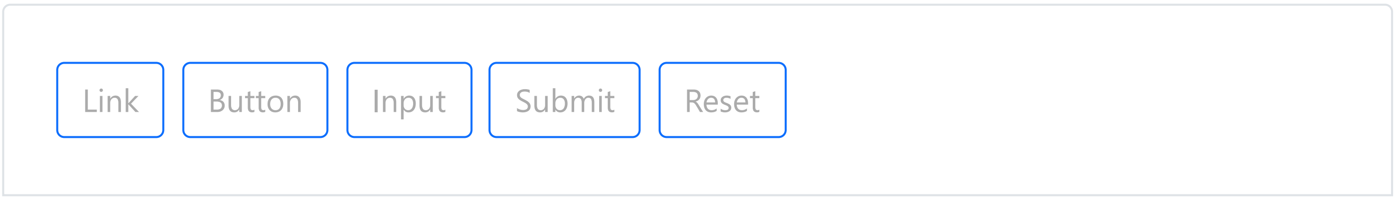
Disable text wrapping

If you don’t want the button text to wrap, you can add the `.text-nowrap` class to the button. In Sass, you can set `$btn-white-space: nowrap` to disable text wrapping for each button.

Button tags

The `.btn` classes are designed to be used with the `<button>` element. However, you can also use these classes on `<a>` or `<input>` elements (though some browsers may apply a slightly different rendering).

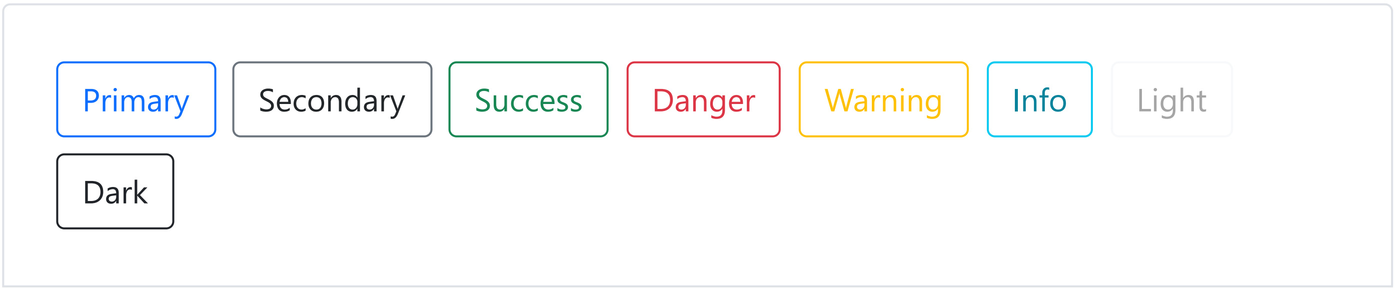
When using button classes on `<a>` elements that are used to trigger in-page functionality (like collapsing content), rather than linking to new pages or sections within the current page, these links should be given a `role="button"` to appropriately convey their purpose to assistive technologies such as screen readers.



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

Outline buttons

In need of a button, but not the hefty background colors they bring? Replace the default modifier classes with the `.btn-outline-*` ones to remove all background images and colors on any button.

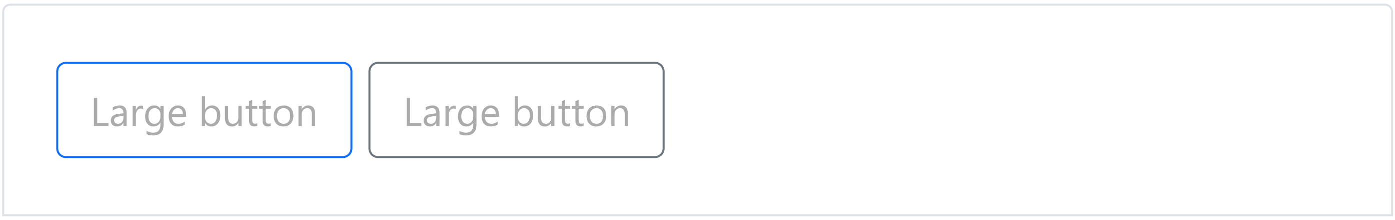


```
<button type="button" class="btn btn-outline-primary">Primary</button>
<button type="button" class="btn btn-outline-secondary">Secondary</button>
<button type="button" class="btn btn-outline-success">Success</button>
<button type="button" class="btn btn-outline-danger">Danger</button>
<button type="button"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

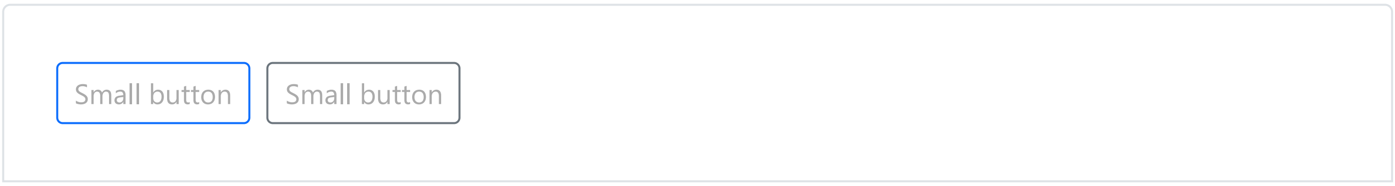
Some of the button styles use a relatively light foreground color, and should only be used on a dark background in order to have sufficient contrast.

Sizes

Fancy larger or smaller buttons? Add `.btn-lg` or `.btn-sm` for additional sizes.



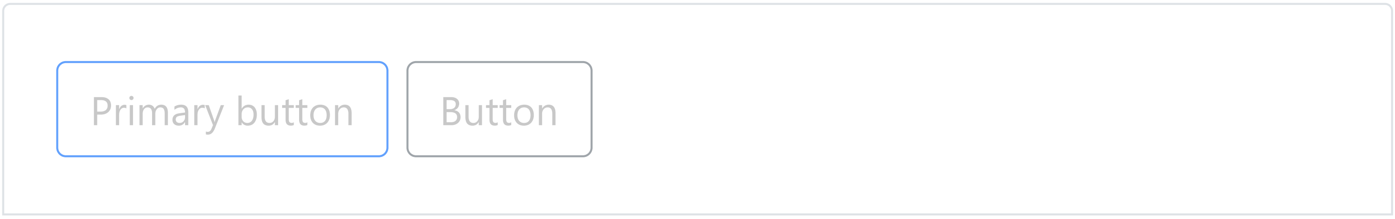
```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-secondary btn-lg">Large button</button>
```



```
<button type="button" class="btn btn-primary btn-sm">Small button</button>
<button type="button" class="btn btn-secondary btn-sm">Small button</button>
```

Disabled state

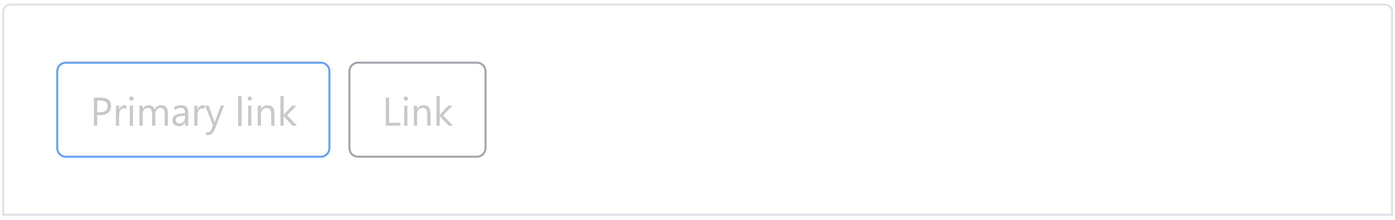
Make buttons look inactive by adding the `disabled` boolean attribute to any `<button>` element. Disabled buttons have `pointer-events: none` applied to, preventing hover and active states from triggering.



```
<button type="button" class="btn btn-lg btn-primary" disabled>Primary button</button
<button type="button" class="btn btn-secondary btn-lg" disabled>Button</button>
```

Disabled buttons using the `<a>` element behave a bit different:

- `<a>`s don't support the `disabled` attribute, so you must add the `.disabled` class to make it visually appear disabled.
- Some future-friendly styles are included to disable all `pointer-events` on anchor buttons.
- Disabled buttons should include the `aria-disabled="true"` attribute to indicate the state of the element to assistive technologies.



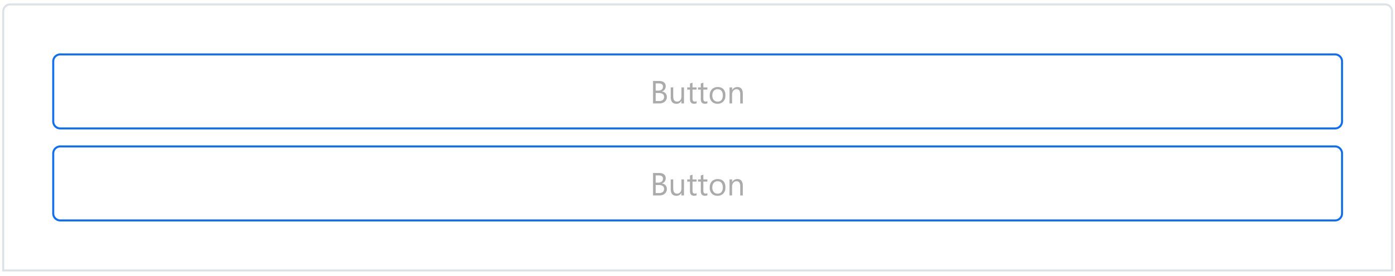
```
<a href="#" class="btn btn-primary btn-lg disabled" tabindex="-1" role="button" aria
<a href="#" class="btn btn-secondary btn-lg disabled" tabindex="-1" role="button" ar
```

Link functionality caveat

The `.disabled` class uses `pointer-events: none` to try to disable the link functionality of `<a>`s, but that CSS property is not yet standardized. In addition, even in browsers that do support `pointer-events: none`, keyboard navigation remains unaffected, meaning that sighted keyboard users and users of assistive technologies will still be able to activate these links. So to be safe, in addition to `aria-disabled="true"`, also include a `tabindex="-1"` attribute on these links to prevent them from receiving keyboard focus, and use custom JavaScript to disable their functionality altogether.

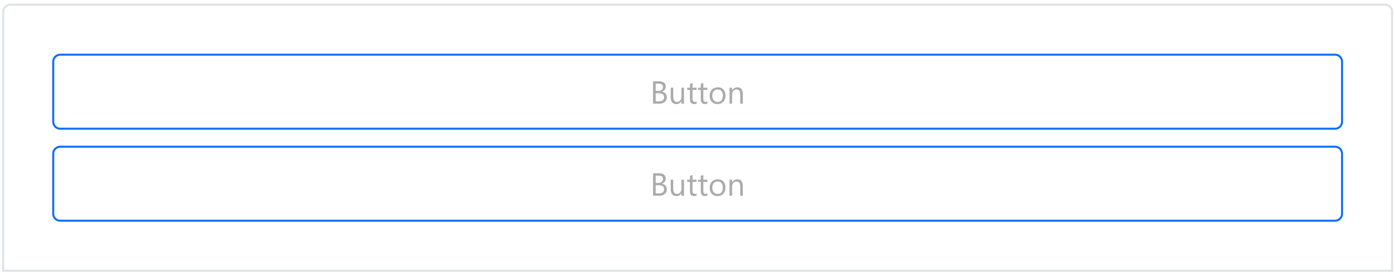
Block buttons

Create responsive stacks of full-width, "block buttons" like those in Bootstrap 4 with a mix of our display and gap utilities. By using utilities instead of button specific classes, we have much greater control over spacing, alignment, and responsive behaviors.



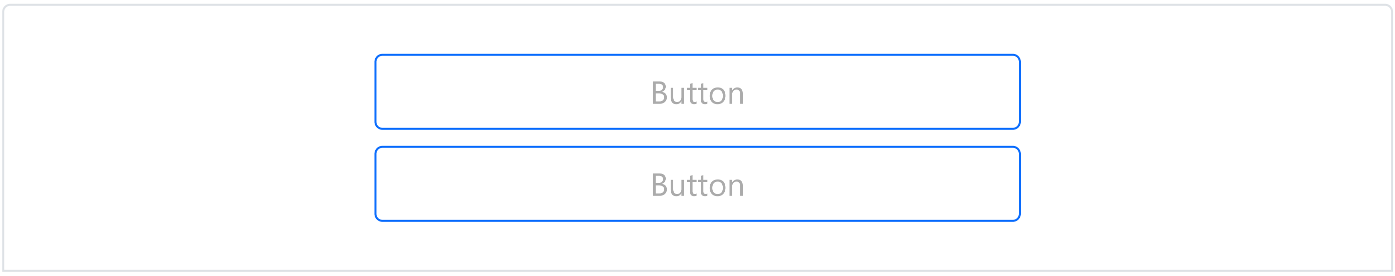
```
<div class="d-grid gap-2">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Here we create a responsive variation, starting with vertically stacked buttons until the `md` breakpoint, where `.d-md-block` replaces the `.d-grid` class, thus nullifying the `gap-2` utility. Resize your browser to see them change.



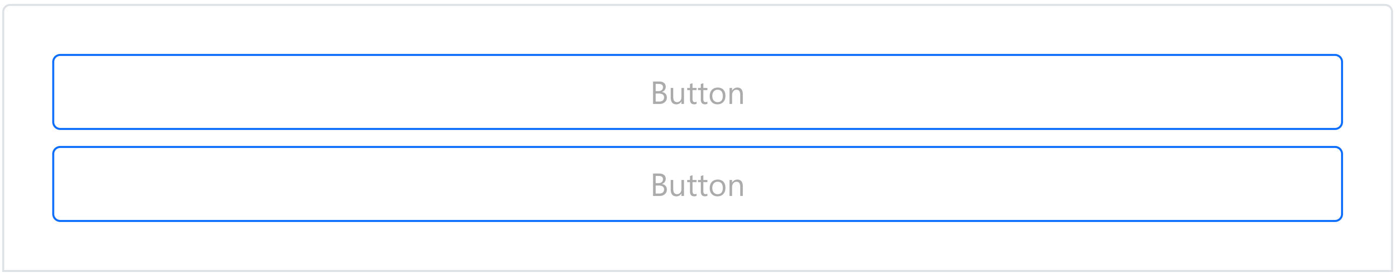
```
<div class="d-grid gap-2 d-md-block">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

You can adjust the width of your block buttons with grid column width classes. For example, for a half-width "block button", use `.col-6`. Center it horizontally with `.mx-auto`, too.



```
<div class="d-grid gap-2 col-6 mx-auto">
  <button class="btn btn-primary" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

Additional utilities can be used to adjust the alignment of buttons when horizontal. Here we've taken our previous responsive example and added some flex utilities and a margin utility on the button to right align the buttons when they're no longer stacked.



```
<div class="d-grid gap-2 d-md-flex justify-content-md-end">
  <button class="btn btn-primary me-md-2" type="button">Button</button>
  <button class="btn btn-primary" type="button">Button</button>
</div>
```

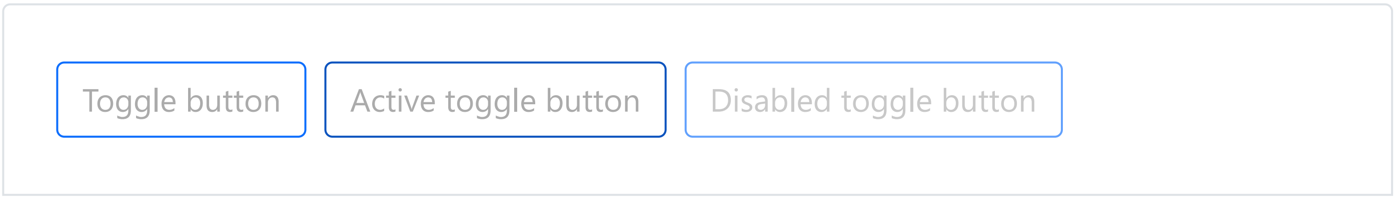
Button plugin

The button plugin allows you to create simple on/off toggle buttons.

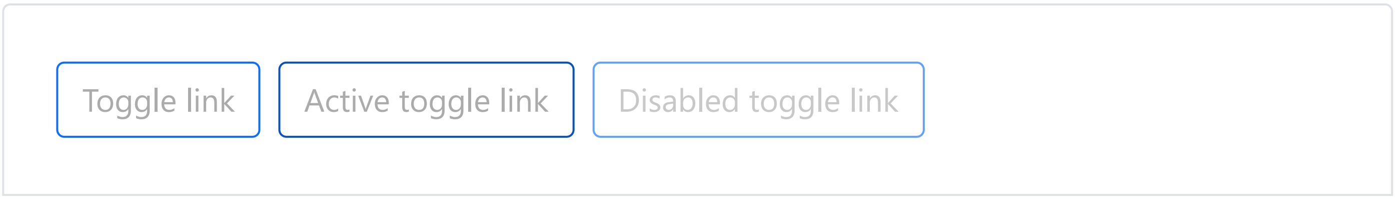
> Visually, these toggle buttons are identical to the checkbox toggle buttons. However, they are conveyed differently by assistive technologies: the checkbox toggles will be announced by screen readers as “checked”/“not checked” (since, despite their appearance, they are fundamentally still checkboxes), whereas these toggle buttons will be announced as “button”/“button pressed”. The choice between these two approaches will depend on the type of toggle you are creating, and whether or not the toggle will make sense to users when announced as a checkbox or as an actual button.

Toggle states

Add `data-bs-toggle="button"` to toggle a button’s `active` state. If you’re pre-toggling a button, you must manually add the `.active` class **and** `aria-pressed="true"` to ensure that it is conveyed appropriately to assistive technologies.



```
<button type="button" class="btn btn-primary" data-bs-toggle="button" autocomplete="
<button type="button" class="btn btn-primary active" data-bs-toggle="button" autocom
<button type="button" class="btn btn-primary" disabled data-bs-toggle="button" autoc
```



```
<a href="#" class="btn btn-primary" role="button" data-bs-toggle="button">Toggle lin
<a href="#" class="btn btn-primary active" role="button" data-bs-toggle="button" ari
<a href="#" class="btn btn-primary disabled" tabindex="-1" aria-disabled="true" role
```

Methods

You can create a button instance with the button constructor, for example:

```
var button = document.getElementById('myButton')
var bsButton = new bootstrap.Button(button)
```

Method	Description
`toggle`	Toggles push state. Gives the button the appearance that it has been activated.
`dispose`	Destroys an element's button. (Removes stored data on the DOM element)
`getInstance`	Static method which allows you to get the button instance associated to a DOM element, you can use it like this: `bootstrap.Button.getInstance(element)`
`getOrCreateInstance`	Static method which returns a button instance associated to a DOM element or create a new one in case it wasn't initialised. You can use it like this: `bootstrap.Button.getOrCreateInstance(element)`

For example, to toggle all buttons

```
var buttons = document.querySelectorAll('.btn')
buttons.forEach(function (button) {
  var button = new bootstrap.Button(button)
  button.toggle()
})
```

Sass

Variables

```
$btn-padding-y:             $input-btn-padding-y;
$btn-padding-x:             $input-btn-padding-x;
$btn-font-family:           $input-btn-font-family;
$btn-font-size:             $input-btn-font-size;
$btn-line-height:           $input-btn-line-height;
$btn-white-space:           null; // Set to `nowrap` to prevent text wrapping

$btn-padding-y-sm:          $input-btn-padding-y-sm;
$btn-padding-x-sm:          $input-btn-padding-x-sm;
$btn-font-size-sm:          $input-btn-font-size-sm;

$btn-padding-y-lg:          $input-btn-padding-y-lg;
$btn-padding-x-lg:          $input-btn-padding-x-lg;
```

```
$btn-font-size-lg:            $input-btn-font-size-lg;

$btn-border-width:            $input-btn-border-width;

$btn-font-weight:             $font-weight-normal;
$btn-box-shadow:              inset 0 1px 0 rgba($white, .15), 0 1px 1px rgba($black
$btn-focus-width:             $input-btn-focus-width;
$btn-focus-box-shadow:        $input-btn-focus-box-shadow;
$btn-disabled-opacity:        .65;
$btn-active-box-shadow:       inset 0 3px 5px rgba($black, .125);

$btn-link-color:              $link-color;
$btn-link-hover-color:        $link-hover-color;
$btn-link-disabled-color:     $gray-600;

// Allows for customizing button radius independently from global border radius
$btn-border-radius:           $border-radius;
$btn-border-radius-sm:        $border-radius-sm;
$btn-border-radius-lg:        $border-radius-lg;

$btn-transition:              color .15s ease-in-out, background-color .15s ease-in-

$btn-hover-bg-shade-amount:       15%;
$btn-hover-bg-tint-amount:        15%;
$btn-hover-border-shade-amount:   20%;
$btn-hover-border-tint-amount:    10%;
$btn-active-bg-shade-amount:      20%;
$btn-active-bg-tint-amount:       20%;
$btn-active-border-shade-amount:  25%;
```

Mixins

There are three mixins for buttons: button and button outline variant mixins (both based on `$theme-colors`), plus a button size mixin.

```
@mixin button-variant(
  $background,
  $border,
  $color: color-contrast($background),
  $hover-background: if($color == $color-contrast-light, shade-color($background, $b
  $hover-border: if($color == $color-contrast-light, shade-color($border, $btn-hover
  $hover-color: color-contrast($hover-background),
  $active-background: if($color == $color-contrast-light, shade-color($background, $
  $active-border: if($color == $color-contrast-light, shade-color($border, $btn-acti
  $active-color: color-contrast($active-background),
  $disabled-background: $background,
```

```scss
  $disabled-border: $border,
  $disabled-color: color-contrast($disabled-background)
) {
  color: $color;
  @include gradient-bg($background);
  border-color: $border;
  @include box-shadow($btn-box-shadow);

  &:hover {
    color: $hover-color;
    @include gradient-bg($hover-background);
    border-color: $hover-border;
  }

  .btn-check:focus + &,
  &:focus {
    color: $hover-color;
    @include gradient-bg($hover-background);
    border-color: $hover-border;
    @if $enable-shadows {
      @include box-shadow($btn-box-shadow, 0 0 0 $btn-focus-width rgba(mix($color, $
    } @else {
      // Avoid using mixin so we can pass custom focus shadow properly
      box-shadow: 0 0 0 $btn-focus-width rgba(mix($color, $border, 15%), .5);
    }
  }

  .btn-check:checked + &,
  .btn-check:active + &,
  &:active,
  &.active,
  .show > &.dropdown-toggle {
    color: $active-color;
    background-color: $active-background;
    // Remove CSS gradients if they're enabled
    background-image: if($enable-gradients, none, null);
    border-color: $active-border;

    &:focus {
      @if $enable-shadows {
        @include box-shadow($btn-active-box-shadow, 0 0 0 $btn-focus-width rgba(mix(
      } @else {
        // Avoid using mixin so we can pass custom focus shadow properly
        box-shadow: 0 0 0 $btn-focus-width rgba(mix($color, $border, 15%), .5);
      }
    }
  }

  &:disabled,
  &.disabled {
    color: $disabled-color;
    background-color: $disabled-background;
    // Remove CSS gradients if they're enabled
    background-image: if($enable-gradients, none, null);
    border-color: $disabled-border;
  }
```

```scss
@mixin button-outline-variant(
  $color,
  $color-hover: color-contrast($color),
  $active-background: $color,
  $active-border: $color,
  $active-color: color-contrast($active-background)
) {
  color: $color;
  border-color: $color;

  &:hover {
    color: $color-hover;
    background-color: $active-background;
    border-color: $active-border;
  }

  .btn-check:focus + &,
  &:focus {
    box-shadow: 0 0 0 $btn-focus-width rgba($color, .5);
  }

  .btn-check:checked + &,
  .btn-check:active + &,
  &:active,
  &.active,
  &.dropdown-toggle.show {
    color: $active-color;
    background-color: $active-background;
    border-color: $active-border;

    &:focus {
      @if $enable-shadows {
        @include box-shadow($btn-active-box-shadow, 0 0 0 $btn-focus-width rgba($col
      } @else {
        // Avoid using mixin so we can pass custom focus shadow properly
        box-shadow: 0 0 0 $btn-focus-width rgba($color, .5);
      }
    }
  }

  &:disabled,
  &.disabled {
    color: $color;
    background-color: transparent;
  }
```

```
@mixin button-size($padding-y, $padding-x, $font-size, $border-radius) {
  padding: $padding-y $padding-x;
  @include font-size($font-size);
  // Manually declare to provide an override to the browser default
  @include border-radius($border-radius, 0);
}
```

Loops

Button variants (for regular and outline buttons) use their respective mixins with our `$theme-colors` map to generate the modifier classes in `scss/_buttons.scss`.

```
@each $color, $value in $theme-colors {
  .btn-#{$color} {
    @include button-variant($value, $value);
  }
}

@each $color, $value in $theme-colors {
  .btn-outline-#{$color} {
    @include button-outline-variant($value);
  }
}
```