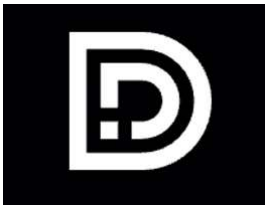
Forms

Examples and usage guidelines for form control styles, layout options, and custom components for creating a wide variety of forms.



On this page

Form control

Style textual inputs and textareas with support for multiple states.

Select

Improve browser default select elements with a custom initial appearance.

Checks & radios

Use our custom radio buttons and checkboxes in forms for selecting input options.

Range

Replace browser default range inputs with our custom version.

Input group

Attach labels and buttons to your inputs for increased semantic value.

Floating labels
Create beautifully simple form labels that float over your input fields.

Layout
Create inline, horizontal, or complex grid-based layouts with your forms.

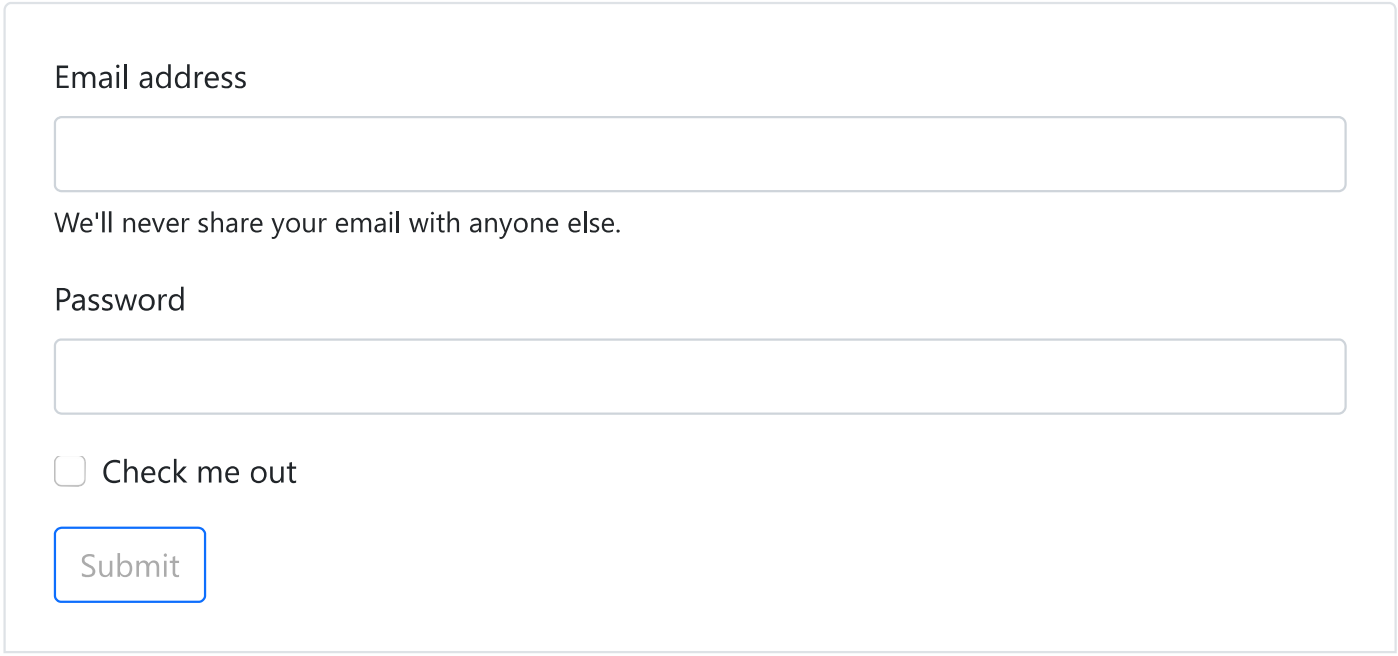
Validation
Validate your forms with custom or native validation behaviors and styles.

Overview

Bootstrap's form controls expand on our Rebooted form styles with classes. Use these classes to opt into their customized displays for a more consistent rendering across browsers and devices.

Be sure to use an appropriate `type` attribute on all inputs (e.g., `email` for email address or `number` for numerical information) to take advantage of newer input controls like email verification, number selection, and more.

Here's a quick example to demonstrate Bootstrap's form styles. Keep reading for documentation on required classes, form layout, and more.




```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedb
    <div id="emailHelp" class="form-text">We'll never share your email with anyone e
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
```

```
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
```

Form text

Block-level or inline-level form text can be created using `.form-text`.

Associating form text with form controls

Form text should be explicitly associated with the form control it relates to using the `aria-describedby` attribute. This will ensure that assistive technologies—such as screen readers—will announce this form text when the user focuses or enters the control.

Form text below inputs can be styled with `.form-text`. If a block-level element will be used, a top margin is added for easy spacing from the inputs above.

Password

Your password must be 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji.

```
<label for="inputPassword5" class="form-label">Password</label>
<input type="password" id="inputPassword5" class="form-control" aria-describedby="pa
<div id="passwordHelpBlock" class="form-text">
  Your password must be 8-20 characters long, contain letters and numbers, and must
</div>
```

Inline text can use any typical inline HTML element (be it a `<span>`, `<small>`, or something else) with nothing more than the `.form-text` class.

Password Must be 8-20 characters long.

```
<div class="row g-3 align-items-center">
  <div class="col-auto">
    <label for="inputPassword6" class="col-form-label">Password</label>
  </div>
  <div class="col-auto">
    <input type="password" id="inputPassword6" class="form-control" aria-describedby
  </div>
  <div class="col-auto">
    <span id="passwordHelpInline" class="form-text">
      Must be 8-20 characters long.
    </span>
  </div>
</div>
```

Disabled forms

Add the `disabled` boolean attribute on an input to prevent user interactions and make it appear lighter.

```
<input class="form-control" id="disabledInput" type="text" placeholder="Disabled inp
```

Add the `disabled` attribute to a `<fieldset>` to disable all the controls within. Browsers treat all native form controls (`<input>`, `<select>`, and `<button>` elements) inside a `<fieldset disabled>` as disabled, preventing both keyboard and mouse interactions on them.

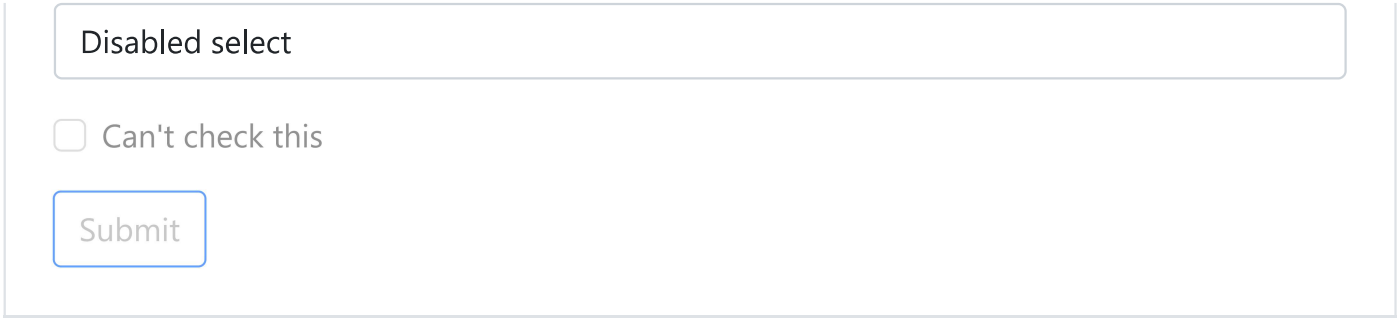
However, if your form also includes custom button-like elements such as `<a class="btn btn-*">...</a>`, these will only be given a style of `pointer-events: none`, meaning they are still focusable and operable using the keyboard. In this case, you must manually modify these controls by adding `tabindex="-1"` to prevent them from receiving focus and `aria-disabled="disabled"` to signal their state to assistive technologies.

Disabled fieldset example

Disabled input

Disabled input

Disabled select menu




```
<form>
  <fieldset disabled>
    <legend>Disabled fieldset example</legend>
    <div class="mb-3">
      <label for="disabledTextInput" class="form-label">Disabled input</label>
      <input type="text" id="disabledTextInput" class="form-control" placeholder="Di
    </div>
    <div class="mb-3">
      <label for="disabledSelect" class="form-label">Disabled select menu</label>
      <select id="disabledSelect" class="form-select">
        <option>Disabled select</option>
      </select>
    </div>
    <div class="mb-3">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" id="disabledFieldsetCheck" d
        <label class="form-check-label" for="disabledFieldsetCheck">
          Can't check this
        </label>
      </div>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </fieldset>
</form>
```

Accessibility

Ensure that all form controls have an appropriate accessible name so that their purpose can be conveyed to users of assistive technologies. The simplest way to achieve this is to use a `<label>` element, or—in the case of buttons—to include sufficiently descriptive text as part of the `<button>...</button>` content.

For situations where it's not possible to include a visible `<label>` or appropriate text content, there are alternative ways of still providing an accessible name, such as:

- `<label>` elements hidden using the `.visually-hidden` class
- Pointing to an existing element that can act as a label using `aria-labelledby`

- Providing a `title` attribute
- Explicitly setting the accessible name on an element using `aria-label`

If none of these are present, assistive technologies may resort to using the `placeholder` attribute as a fallback for the accessible name on `<input>` and `<textarea>` elements. The examples in this section provide a few suggested, case-specific approaches.

While using visually hidden content (`.visually-hidden`, `aria-label`, and even `placeholder` content, which disappears once a form field has content) will benefit assistive technology users, a lack of visible label text may still be problematic for certain users. Some form of visible label is generally the best approach, both for accessibility and usability.

Sass

Many form variables are set at a general level to be re-used and extended by individual form components. You'll see these most often as `$btn-input-*` and `$input-*` variables.

Variables

`$btn-input-*` variables are shared global variables between our buttons and our form components. You'll find these frequently reassigned as values to other component-specific variables.

```
$input-btn-padding-y:         .375rem;
$input-btn-padding-x:         .75rem;
$input-btn-font-family:       null;
$input-btn-font-size:         $font-size-base;
$input-btn-line-height:       $line-height-base;

$input-btn-focus-width:         .25rem;
$input-btn-focus-color-opacity: .25;
$input-btn-focus-color:         rgba($component-active-bg, $input-btn-focus-color-op
$input-btn-focus-blur:          0;
$input-btn-focus-box-shadow:    0 0 $input-btn-focus-blur $input-btn-focus-width $in

$input-btn-padding-y-sm:      .25rem;
$input-btn-padding-x-sm:      .5rem;
$input-btn-font-size-sm:      $font-size-sm;

$input-btn-padding-y-lg:      .5rem;
$input-btn-padding-x-lg:      1rem;
$input-btn-font-size-lg:      $font-size-lg;
```

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

Bootstrap

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 v5.1.3.

Analytics by Fathom.

Links

Home

Docs

Examples

Themes

Blog

Swag Store

Guides

Getting started

Starter template

Webpack

Parcel

Projects

Bootstrap 5

Bootstrap 4

Icons

RFS

npm starter

Community

Issues

Discussions

Corporate sponsors

Open Collective

Stack Overflow