Search...

Forms

View on GitHub

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

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.

Email address

We'll never share your email with anyone else.

Password

☐ Check me out

Submit

Copy

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp">
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with
anyone else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="form-group 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>
```

Form controls

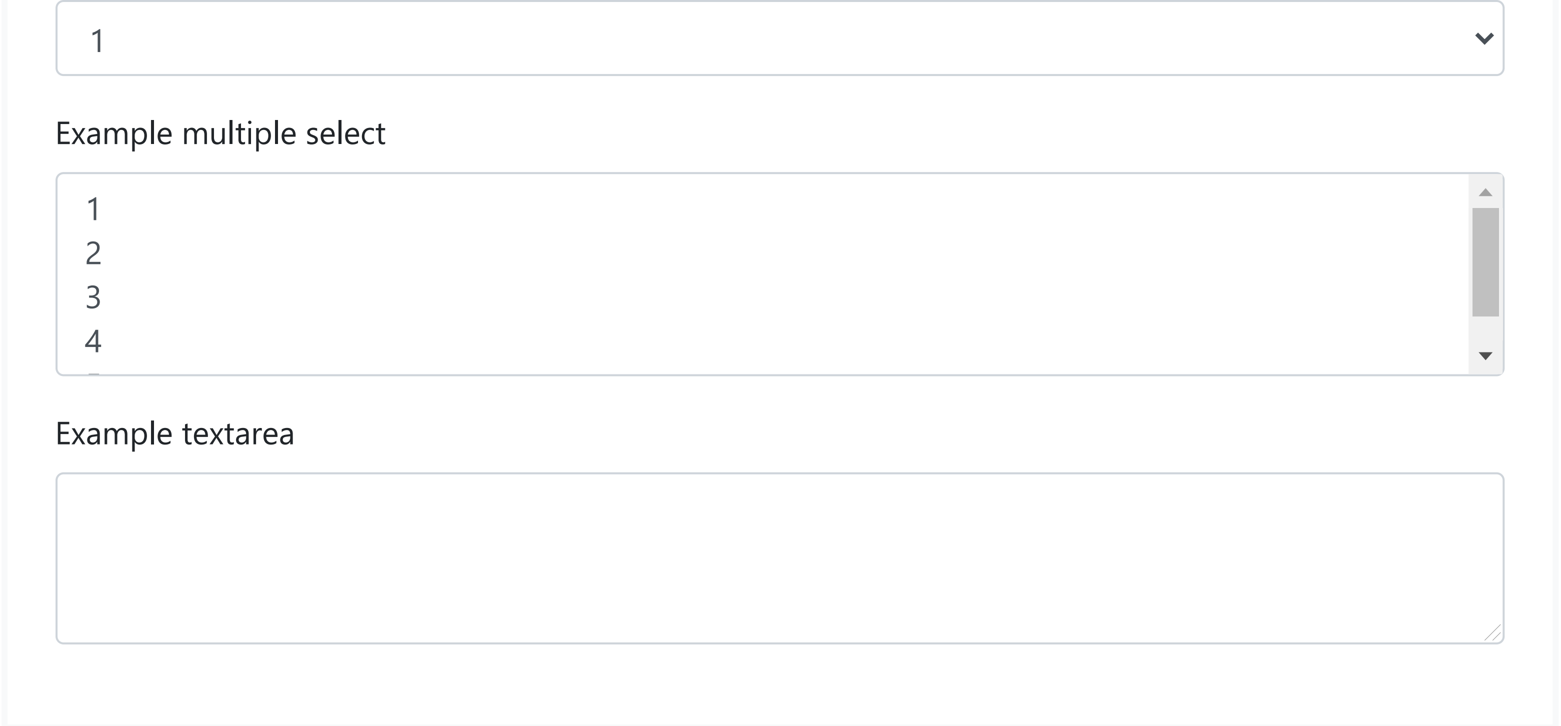
Textual form controls—like `<input>`s, `<select>`s, and `<textarea>`s—are styled with the `.form-control` class. Included are styles for general appearance, focus state, sizing, and more.

Be sure to explore our custom forms to further style `<select>`s.

Email address

name@example.com

Example select


1

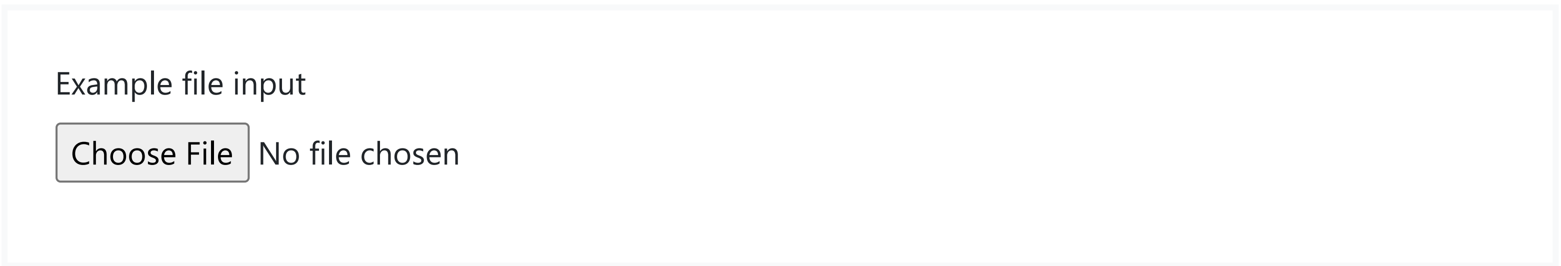
Example multiple select

1
2
3
4

Example textarea

```
<form>
  <div class="form-group">
    <label for="exampleFormControlInput1">Email address</label>
    <input type="email" class="form-control" id="exampleFormControlInput1"
placeholder="name@example.com">
  </div>
  <div class="form-group">
    <label for="exampleFormControlSelect1">Example select</label>
    <select class="form-control" id="exampleFormControlSelect1">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
      <option>5</option>
    </select>
  </div>
  <div class="form-group">
    <label for="exampleFormControlSelect2">Example multiple select</label>
    <select multiple class="form-control" id="exampleFormControlSelect2">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
      <option>5</option>
    </select>
  </div>
  <div class="form-group">
    <label for="exampleFormControlTextarea1">Example textarea</label>
    <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
  </div>
</form>
```

For file inputs, swap the `.form-control` for `.form-control-file`.

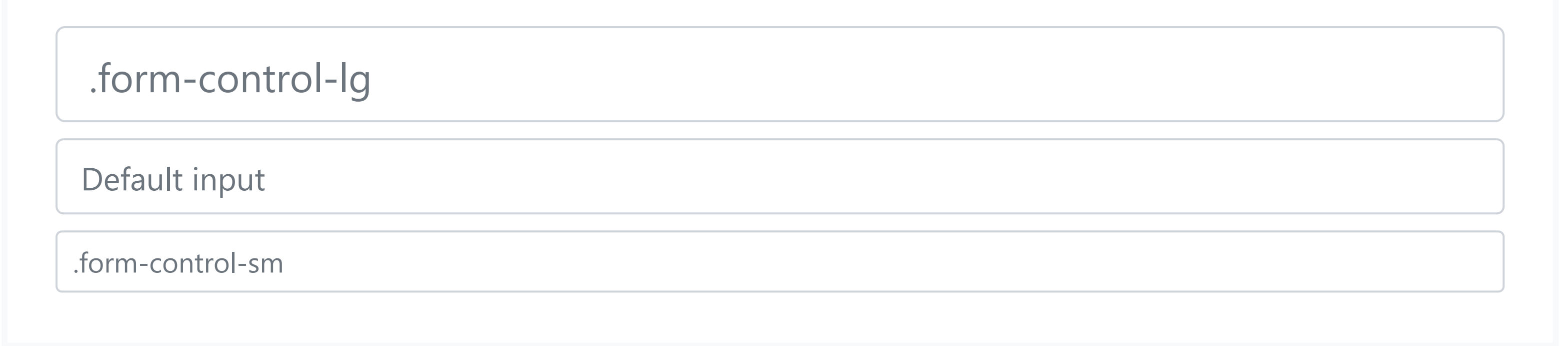

Example file input

Choose File No file chosen

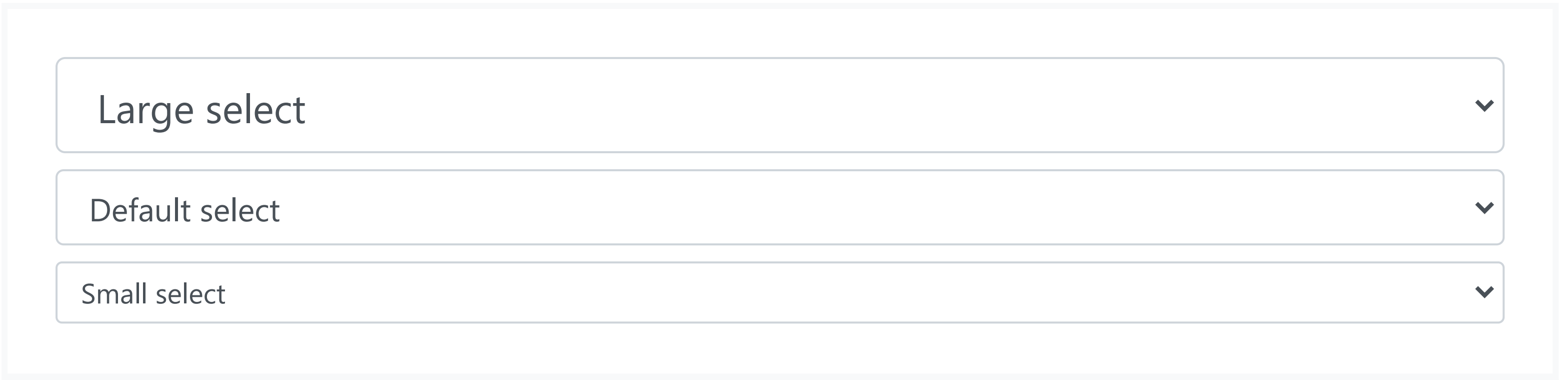
```
<form>
  <div class="form-group">
    <label for="exampleFormControlFile1">Example file input</label>
    <input type="file" class="form-control-file" id="exampleFormControlFile1">
  </div>
</form>
```

Sizing

Set heights using classes like `.form-control-lg` and `.form-control-sm`.



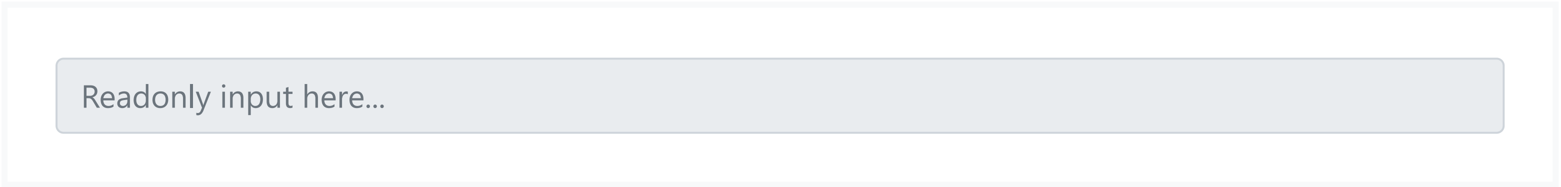
```html
<input class="form-control form-control-lg" type="text" placeholder=".form-control-lg">
<input class="form-control" type="text" placeholder="Default input">
<input class="form-control form-control-sm" type="text" placeholder=".form-control-sm">
```



```html
<select class="form-control form-control-lg">
  <option>Large select</option>
</select>
<select class="form-control">
  <option>Default select</option>
</select>
<select class="form-control form-control-sm">
  <option>Small select</option>
</select>
```

Readonly

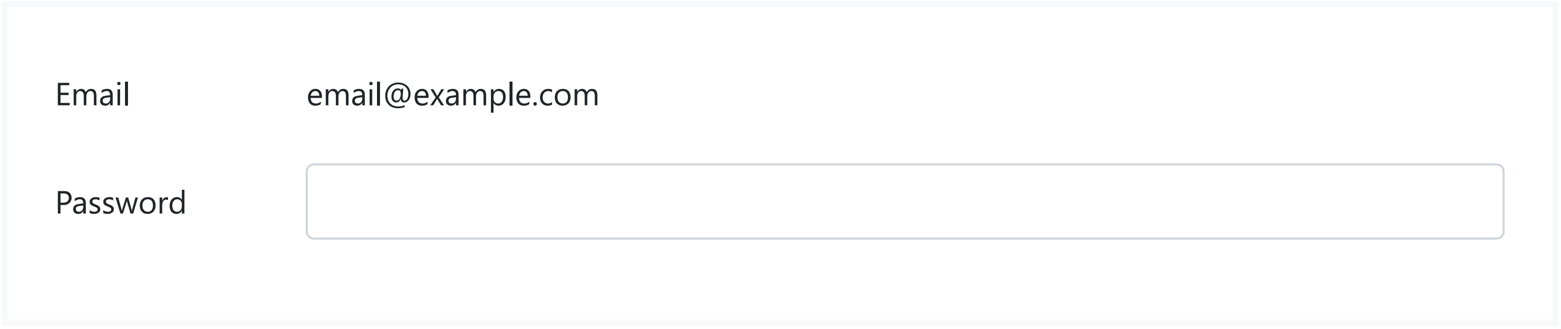
Add the `readonly` boolean attribute on an input to prevent modification of the input's value. Read-only inputs appear lighter (just like disabled inputs), but retain the standard cursor.



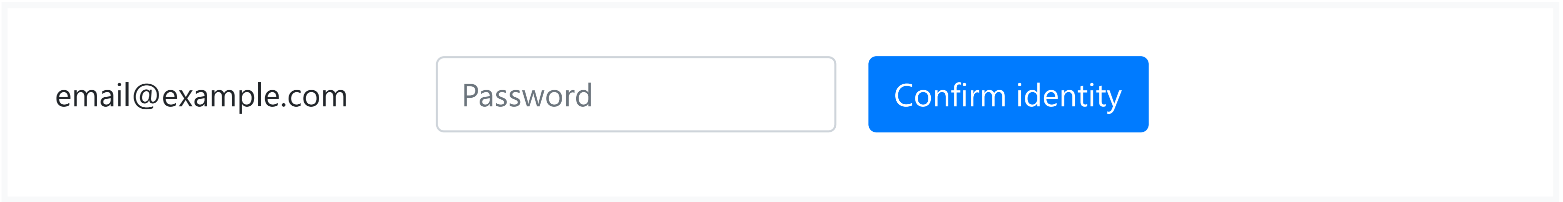
```html
<input class="form-control" type="text" placeholder="Readonly input here..." readonly>
```

Readonly plain text

If you want to have `<input readonly>` elements in your form styled as plain text, use the `.form-control-plaintext` class to remove the default form field styling and preserve the correct margin and padding.



```
<form>
  <div class="form-group row">
    <label for="staticEmail" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="text" readonly class="form-control-plaintext" id="staticEmail"
value="email@example.com">
    </div>
  </div>
  <div class="form-group row">
    <label for="inputPassword" class="col-sm-2 col-form-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword">
    </div>
  </div>
</form>
```

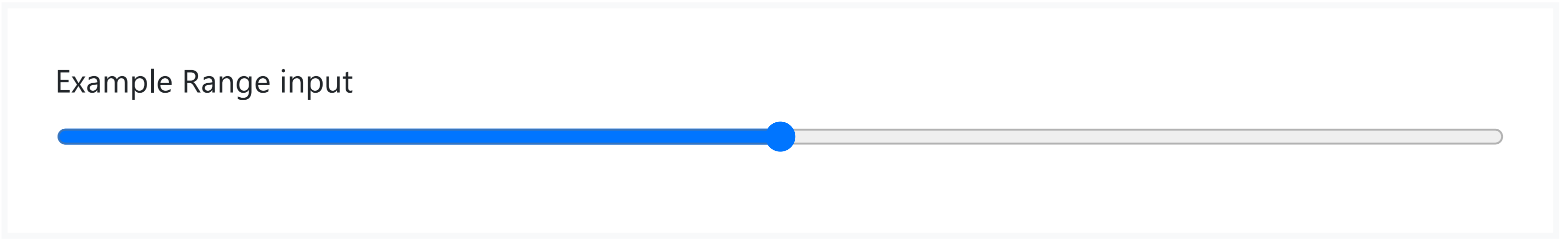


Copy

```
<form class="form-inline">
  <div class="form-group mb-2">
    <label for="staticEmail2" class="sr-only">Email</label>
    <input type="text" readonly class="form-control-plaintext" id="staticEmail2"
value="email@example.com">
  </div>
  <div class="form-group mx-sm-3 mb-2">
    <label for="inputPassword2" class="sr-only">Password</label>
    <input type="password" class="form-control" id="inputPassword2" placeholder="Password">
  </div>
  <button type="submit" class="btn btn-primary mb-2">Confirm identity</button>
</form>
```

Range Inputs

Set horizontally scrollable range inputs using `.form-control-range`.



Copy

```
<form>
  <div class="form-group">
    <label for="formControlRange">Example Range input</label>
    <input type="range" class="form-control-range" id="formControlRange">
  </div>
</form>
```

Checkboxes and radios

Default checkboxes and radios are improved upon with the help of `.form-check`, **a single class for both input types that improves the layout and behavior of their HTML elements**. Checkboxes are for selecting one or several options in a list, while radios are for selecting one option from many.

Disabled checkboxes and radios are supported. The `disabled` attribute will apply a lighter color to help indicate the input's state.

Checkboxes and radio buttons support HTML-based form validation and provide concise, accessible labels. As such, our `<input>`s and `<label>`s are sibling elements as opposed to an `<input>` within a `<label>`. This is slightly more verbose as you must specify `id` and `for` attributes to relate the `<input>` and `<label>`.

Default (stacked)

By default, any number of checkboxes and radios that are immediate sibling will be vertically stacked and appropriately spaced with `.form-check`.

☐ Default checkbox
☐ Disabled checkbox

Copy

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="defaultCheck1">
  <label class="form-check-label" for="defaultCheck1">
    Default checkbox
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="defaultCheck2" disabled>
  <label class="form-check-label" for="defaultCheck2">
    Disabled checkbox
  </label>
</div>
```

● Default radio
○ Second default radio
○ Disabled radio

Copy

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios" id="exampleRadios1"
value="option1" checked>
  <label class="form-check-label" for="exampleRadios1">
    Default radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios" id="exampleRadios2"
value="option2">
  <label class="form-check-label" for="exampleRadios2">
    Second default radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios" id="exampleRadios3"
value="option3" disabled>
  <label class="form-check-label" for="exampleRadios3">
    Disabled radio
  </label>
</div>
```

Inline

Group checkboxes or radios on the same horizontal row by adding `.form-check-inline` to any `.form-check`.

☐ 1 ☐ 2 ☐ 3 (disabled)

Copy

```
<div class="form-check form-check-inline">
  <input class="form-check-input" type="checkbox" id="inlineCheckbox1" value="option1">
  <label class="form-check-label" for="inlineCheckbox1">1</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="checkbox" id="inlineCheckbox2" value="option2">
  <label class="form-check-label" for="inlineCheckbox2">2</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="checkbox" id="inlineCheckbox3" value="option3"
disabled>
  <label class="form-check-label" for="inlineCheckbox3">3 (disabled)</label>
</div>
```

○ 1 ○ 2 ○ 3 (disabled)

Copy

```
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" name="inlineRadioOptions" id="inlineRadio1"
value="option1">
  <label class="form-check-label" for="inlineRadio1">1</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" name="inlineRadioOptions" id="inlineRadio2"
value="option2">
  <label class="form-check-label" for="inlineRadio2">2</label>
</div>
<div class="form-check form-check-inline">
  <input class="form-check-input" type="radio" name="inlineRadioOptions" id="inlineRadio3"
value="option3" disabled>
  <label class="form-check-label" for="inlineRadio3">3 (disabled)</label>
</div>
```

Without labels

Add `.position-static` to inputs within `.form-check` that don't have any label text. Remember to still provide some form of accessible name for assistive technologies (for instance, using `aria-label`).

☐
○

Copy

```
<div class="form-check">
  <input class="form-check-input position-static" type="checkbox" id="blankCheckbox"
value="option1" aria-label="...">
</div>
<div class="form-check">
  <input class="form-check-input position-static" type="radio" name="blankRadio"
id="blankRadio1" value="option1" aria-label="...">
</div>
```

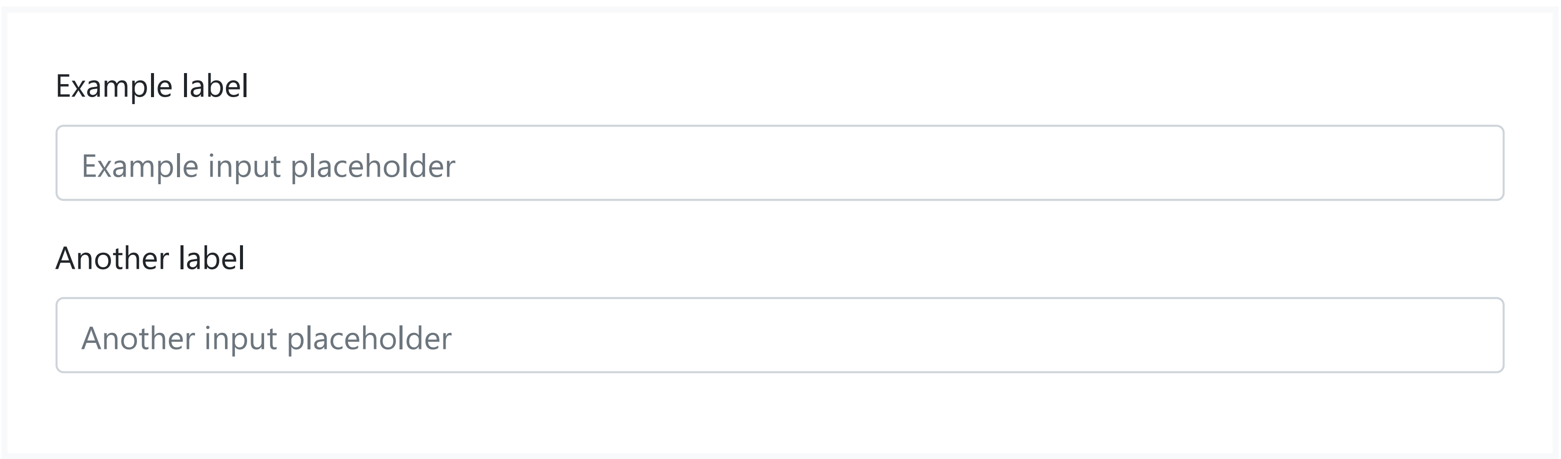
Layout

Since Bootstrap applies `display: block` and `width: 100%` to almost all our form controls, forms will by default stack vertically. Additional classes can be used to vary this layout on a per-form basis.

Form groups

The `.form-group` class is the easiest way to add some structure to forms. It provides a flexible class that encourages proper grouping of labels, controls, optional help text, and form validation messaging. By default it only applies `margin-bottom`, but it picks up additional styles in `.form-inline` as needed. Use it with

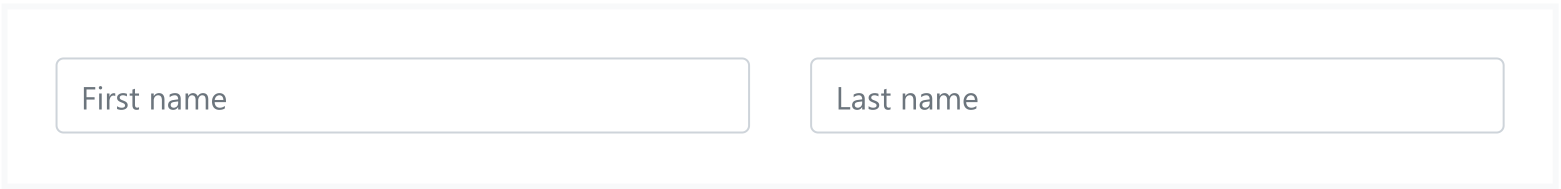
`<fieldset>`s, `<div>`s, or nearly any other element.



```
<form>
  <div class="form-group">
    <label for="formGroupExampleInput">Example label</label>
    <input type="text" class="form-control" id="formGroupExampleInput" placeholder="Example 
input placeholder">
  </div>
  <div class="form-group">
    <label for="formGroupExampleInput2">Another label</label>
    <input type="text" class="form-control" id="formGroupExampleInput2" placeholder="Another 
input placeholder">
  </div>
</form>
```

Form grid

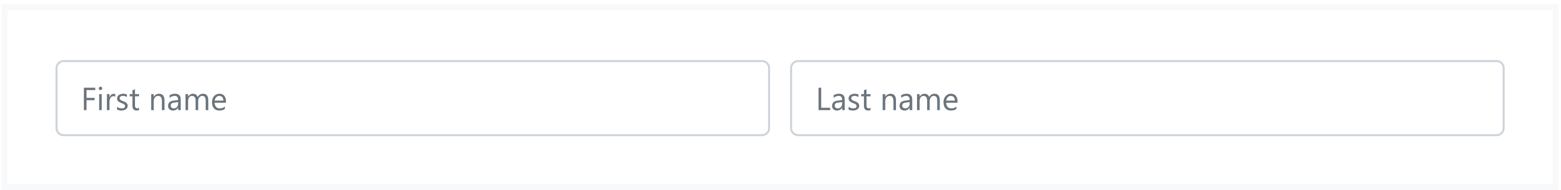
More complex forms can be built using our grid classes. Use these for form layouts that require multiple columns, varied widths, and additional alignment options.



```
<form>
  <div class="row">
    <div class="col">
      <input type="text" class="form-control" placeholder="First name">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="Last name">
    </div>
  </div>
</form>
```

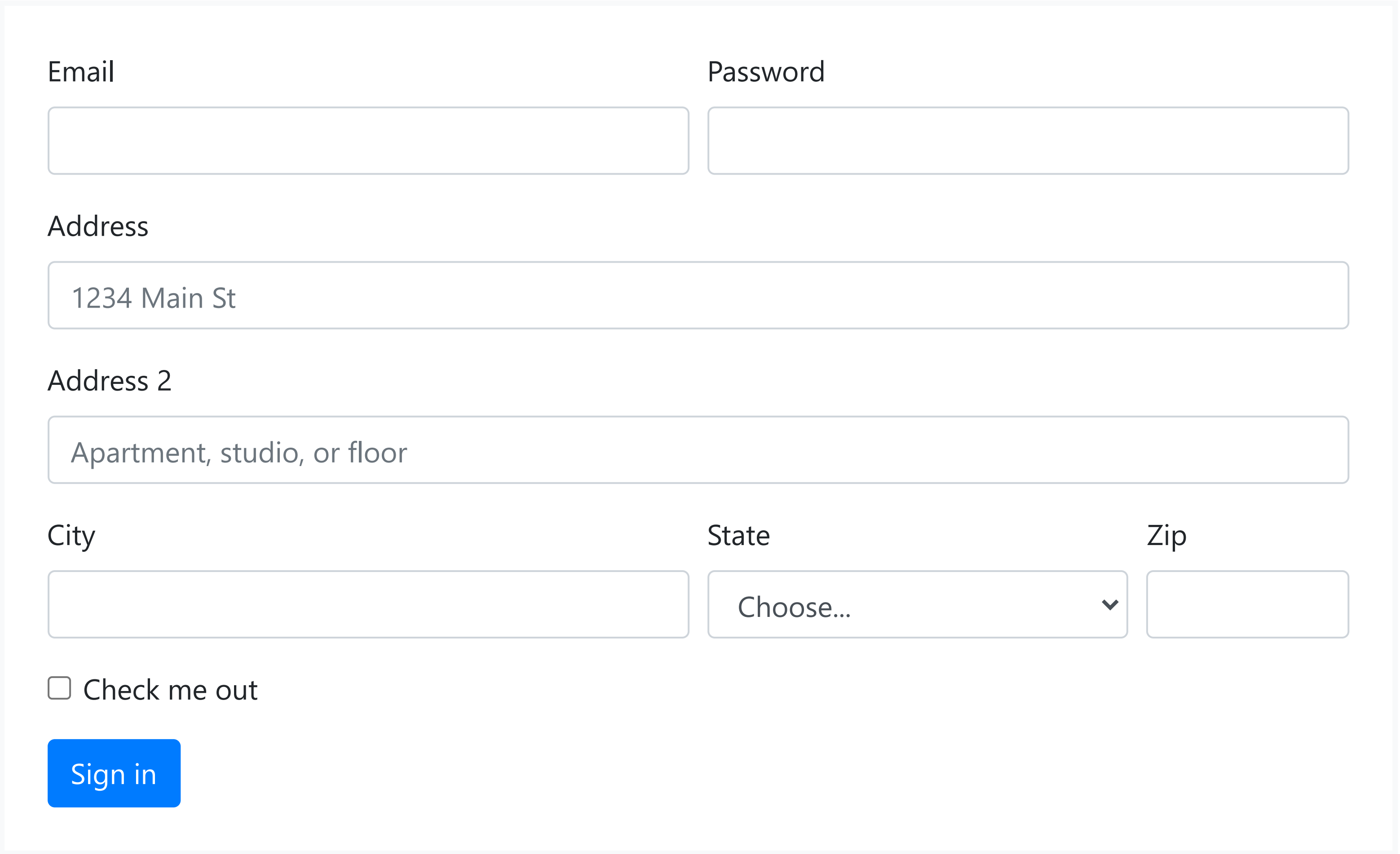
Form row

You may also swap `.row` for `.form-row`, a variation of our standard grid row that overrides the default column gutters for tighter and more compact layouts.



```
<form>
  <div class="form-row">
    <div class="col">
      <input type="text" class="form-control" placeholder="First name">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="Last name">
    </div>
  </div>
</form>
```

More complex layouts can also be created with the grid system.

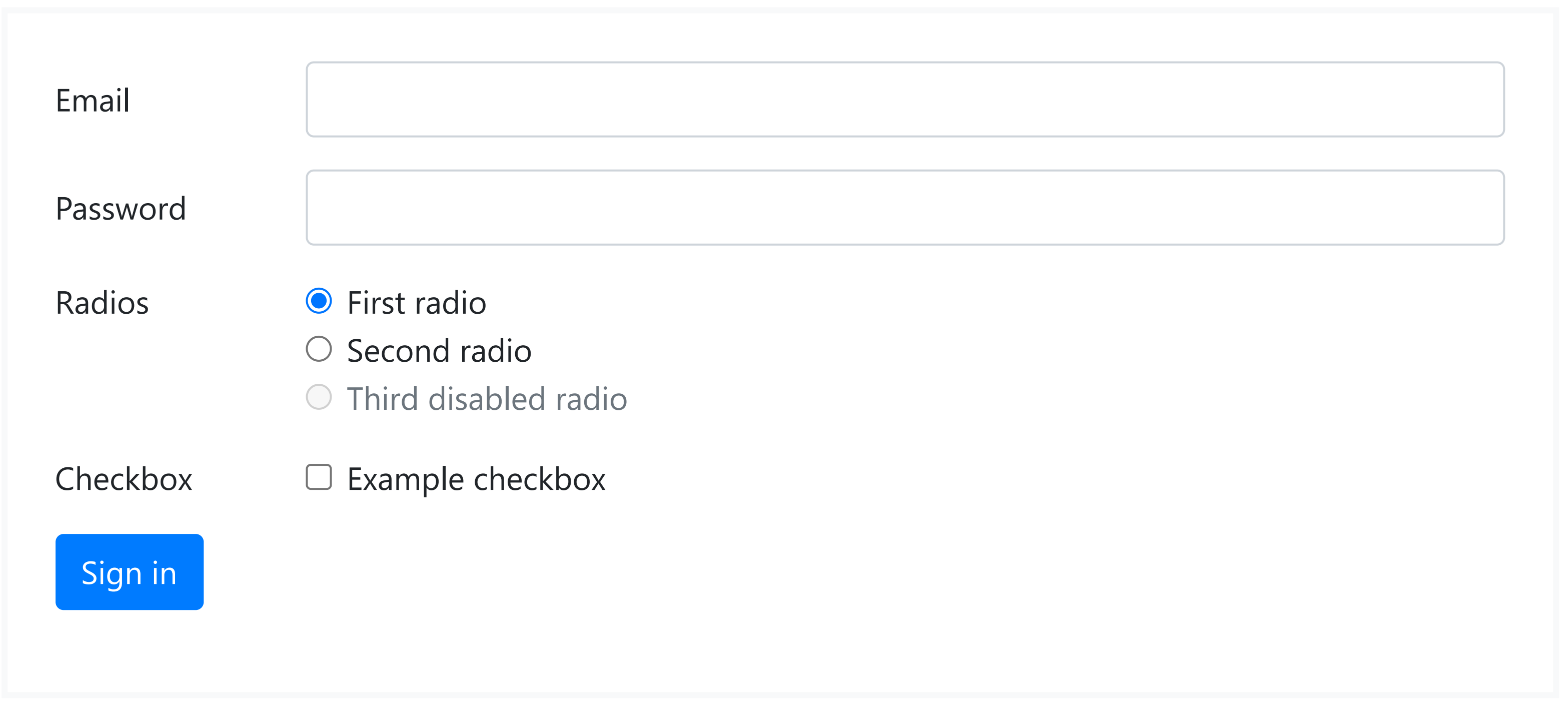


```html
<form>
  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="inputEmail4">Email</label>
      <input type="email" class="form-control" id="inputEmail4">
    </div>
    <div class="form-group col-md-6">
      <label for="inputPassword4">Password</label>
      <input type="password" class="form-control" id="inputPassword4">
    </div>
  </div>
  <div class="form-group">
    <label for="inputAddress">Address</label>
    <input type="text" class="form-control" id="inputAddress" placeholder="1234 Main St">
  </div>
  <div class="form-group">
    <label for="inputAddress2">Address 2</label>
    <input type="text" class="form-control" id="inputAddress2" placeholder="Apartment, studio, or floor">
  </div>
  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="inputCity">City</label>
      <input type="text" class="form-control" id="inputCity">
    </div>
    <div class="form-group col-md-4">
      <label for="inputState">State</label>
      <select id="inputState" class="form-control">
        <option selected>Choose...</option>
        <option>...</option>
      </select>
    </div>
    <div class="form-group col-md-2">
      <label for="inputZip">Zip</label>
      <input type="text" class="form-control" id="inputZip">
    </div>
  </div>
  <div class="form-group">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" id="gridCheck">
      <label class="form-check-label" for="gridCheck">
        Check me out
      </label>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

Horizontal form

Create horizontal forms with the grid by adding the `.row` class to form groups and using the `.col-*-*` classes to specify the width of your labels and controls. Be sure to add `.col-form-label` to your `<label>`s as well so they're vertically centered with their associated form controls.

At times, you maybe need to use margin or padding utilities to create that perfect alignment you need. For example, we've removed the `padding-top` on our stacked radio inputs label to better align the text baseline.



Email

Password

Radios
- ● First radio
- ○ Second radio
- ○ Third disabled radio

Checkbox
- ☐ Example checkbox

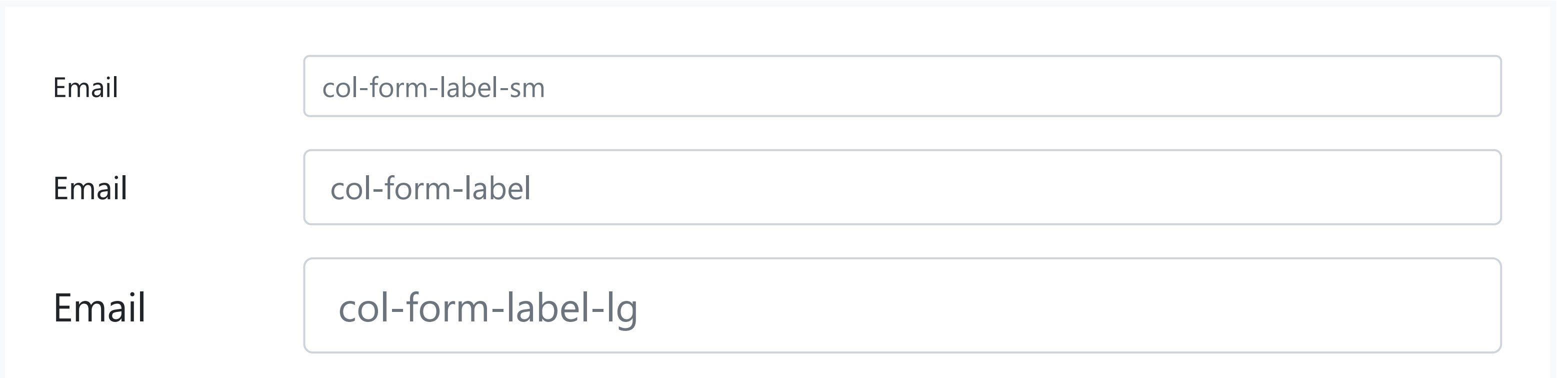
Sign in

Copy

```
<form>
  <div class="form-group row">
    <label for="inputEmail3" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3">
    </div>
  </div>
  <div class="form-group row">
    <label for="inputPassword3" class="col-sm-2 col-form-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3">
    </div>
  </div>
  <fieldset class="form-group">
    <div class="row">
      <legend class="col-form-label col-sm-2 pt-0">Radios</legend>
      <div class="col-sm-10">
        <div class="form-check">
          <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios1"
value="option1" checked>
          <label class="form-check-label" for="gridRadios1">
            First radio
          </label>
        </div>
        <div class="form-check">
          <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios2"
value="option2">
          <label class="form-check-label" for="gridRadios2">
            Second radio
          </label>
        </div>
        <div class="form-check disabled">
          <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios3"
value="option3" disabled>
          <label class="form-check-label" for="gridRadios3">
            Third disabled radio
          </label>
        </div>
      </div>
    </div>
  </fieldset>
  <div class="form-group row">
    <div class="col-sm-2">Checkbox</div>
    <div class="col-sm-10">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" id="gridCheck1">
        <label class="form-check-label" for="gridCheck1">
          Example checkbox
        </label>
      </div>
    </div>
  </div>
  <div class="form-group row">
    <div class="col-sm-10">
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
  </div>
</form>
```

Horizontal form label sizing

Be sure to use `.col-form-label-sm` or `.col-form-label-lg` to your `<label>`s or `<legend>`s to correctly follow the size of `.form-control-lg` and `.form-control-sm`.



Email col-form-label-sm

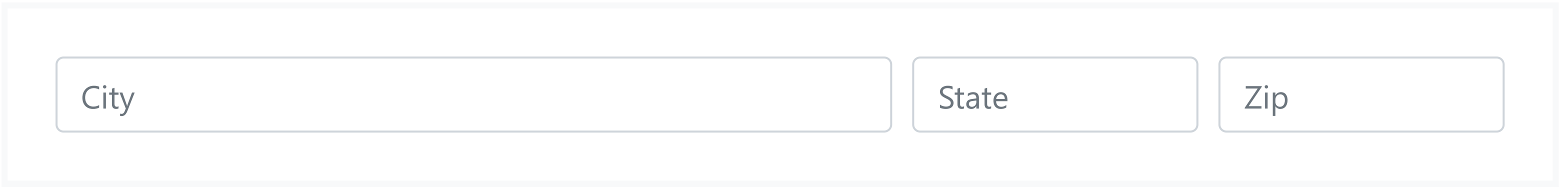
Email col-form-label

Email col-form-label-lg

```
<form>
  <div class="form-group row">
    <label for="colFormLabelSm" class="col-sm-2 col-form-label col-form-label-sm">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control form-control-sm" id="colFormLabelSm" placeholder="col-form-label-sm">
    </div>
  </div>
  <div class="form-group row">
    <label for="colFormLabel" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="colFormLabel" placeholder="col-form-label">
    </div>
  </div>
  <div class="form-group row">
    <label for="colFormLabelLg" class="col-sm-2 col-form-label col-form-label-lg">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control form-control-lg" id="colFormLabelLg" placeholder="col-form-label-lg">
    </div>
  </div>
</form>
```

Column sizing

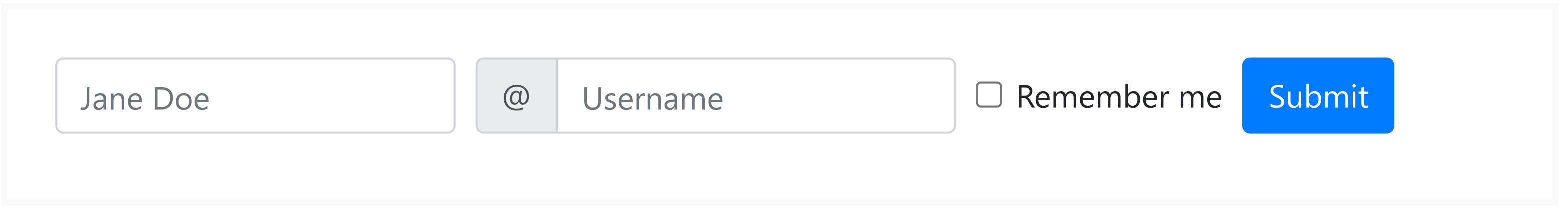
As shown in the previous examples, our grid system allows you to place any number of `.col`s within a `.row` or `.form-row`. They'll split the available width equally between them. You may also pick a subset of your columns to take up more or less space, while the remaining `.col`s equally split the rest, with specific column classes like `.col-7`.



```
<form>
  <div class="form-row">
    <div class="col-7">
      <input type="text" class="form-control" placeholder="City">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="State">
    </div>
    <div class="col">
      <input type="text" class="form-control" placeholder="Zip">
    </div>
  </div>
</form>
```

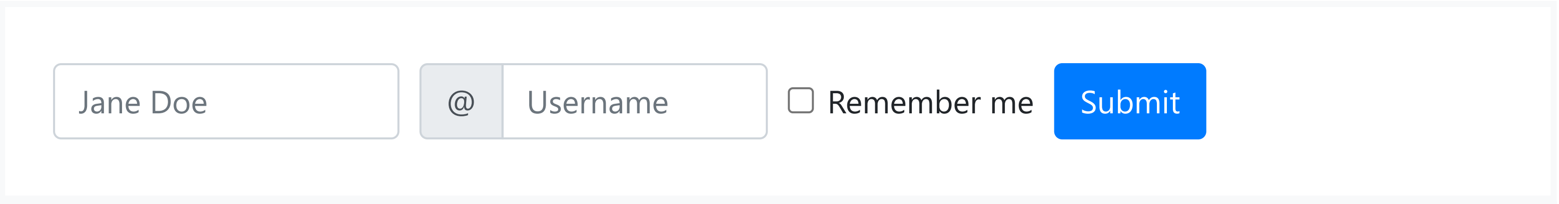
Auto-sizing

The example below uses a flexbox utility to vertically center the contents and changes `.col` to `.col-auto` so that your columns only take up as much space as needed. Put another way, the column sizes itself based on the contents.



```
<form>
  <div class="form-row align-items-center">
    <div class="col-auto">
      <label class="sr-only" for="inlineFormInput">Name</label>
      <input type="text" class="form-control mb-2" id="inlineFormInput" placeholder="Jane
Doe">
    </div>
    <div class="col-auto">
      <label class="sr-only" for="inlineFormInputGroup">Username</label>
      <div class="input-group mb-2">
        <div class="input-group-prepend">
          <div class="input-group-text">@</div>
        </div>
        <input type="text" class="form-control" id="inlineFormInputGroup"
placeholder="Username">
      </div>
    </div>
    <div class="col-auto">
      <div class="form-check mb-2">
        <input class="form-check-input" type="checkbox" id="autoSizingCheck">
        <label class="form-check-label" for="autoSizingCheck">
          Remember me
        </label>
      </div>
    </div>
    <div class="col-auto">
      <button type="submit" class="btn btn-primary mb-2">Submit</button>
    </div>
  </div>
</form>
```

You can then remix that once again with size-specific column classes.



```
<form>
  <div class="form-row align-items-center">
    <div class="col-sm-3 my-1">
      <label class="sr-only" for="inlineFormInputName">Name</label>
      <input type="text" class="form-control" id="inlineFormInputName" placeholder="Jane Doe">
    </div>
    <div class="col-sm-3 my-1">
      <label class="sr-only" for="inlineFormInputGroupUsername">Username</label>
      <div class="input-group">
        <div class="input-group-prepend">
          <div class="input-group-text">@</div>
        </div>
        <input type="text" class="form-control" id="inlineFormInputGroupUsername"
placeholder="Username">
      </div>
    </div>
    <div class="col-auto my-1">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" id="autoSizingCheck2">
        <label class="form-check-label" for="autoSizingCheck2">
          Remember me
        </label>
      </div>
    </div>
    <div class="col-auto my-1">
      <button type="submit" class="btn btn-primary">Submit</button>
    </div>
  </div>
</form>
```

And of course custom form controls are supported.

Choose... ☐ Remember my preference Submit

Copy

```
<form>
  <div class="form-row align-items-center">
    <div class="col-auto my-1">
      <label class="mr-sm-2 sr-only" for="inlineFormCustomSelect">Preference</label>
      <select class="custom-select mr-sm-2" id="inlineFormCustomSelect">
        <option selected>Choose...</option>
        <option value="1">One</option>
        <option value="2">Two</option>
        <option value="3">Three</option>
      </select>
    </div>
    <div class="col-auto my-1">
      <div class="custom-control custom-checkbox mr-sm-2">
        <input type="checkbox" class="custom-control-input" id="customControlAutosizing">
        <label class="custom-control-label" for="customControlAutosizing">Remember my
preference</label>
      </div>
    </div>
    <div class="col-auto my-1">
      <button type="submit" class="btn btn-primary">Submit</button>
    </div>
  </div>
</form>
```

Inline forms

Use the `.form-inline` class to display a series of labels, form controls, and buttons on a single horizontal row. Form controls within inline forms vary slightly from their default states.

- Controls are `display: flex`, collapsing any HTML white space and allowing you to provide alignment control with spacing and flexbox utilities.
- Controls and input groups receive `width: auto` to override the Bootstrap default `width: 100%`.
- Controls **only appear inline in viewports that are at least 576px wide** to account for narrow viewports on mobile devices.

You may need to manually address the width and alignment of individual form controls with spacing utilities (as shown below). Lastly, be sure to always include a `<label>` with each form control, even if you need to hide it from non-screenreader visitors with `.sr-only`.

Jane Doe @ Username ☐ Remember me Submit

Copy

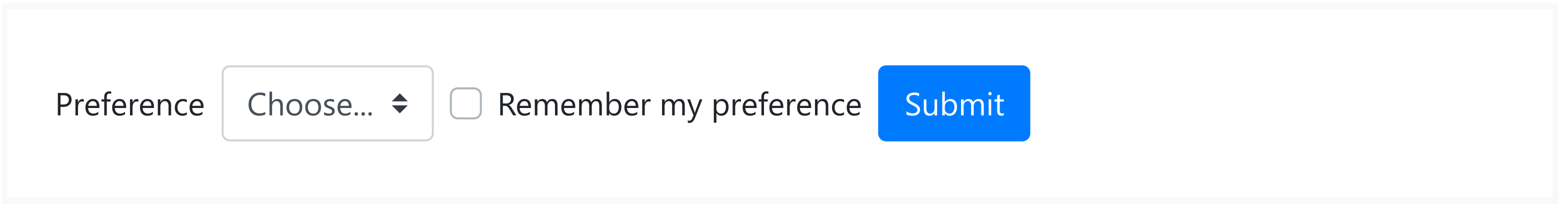
```
<form class="form-inline">
  <label class="sr-only" for="inlineFormInputName2">Name</label>
  <input type="text" class="form-control mb-2 mr-sm-2" id="inlineFormInputName2"
placeholder="Jane Doe">

  <label class="sr-only" for="inlineFormInputGroupUsername2">Username</label>
  <div class="input-group mb-2 mr-sm-2">
    <div class="input-group-prepend">
      <div class="input-group-text">@</div>
    </div>
    <input type="text" class="form-control" id="inlineFormInputGroupUsername2"
placeholder="Username">
  </div>

  <div class="form-check mb-2 mr-sm-2">
    <input class="form-check-input" type="checkbox" id="inlineFormCheck">
    <label class="form-check-label" for="inlineFormCheck">
      Remember me
    </label>
  </div>

  <button type="submit" class="btn btn-primary mb-2">Submit</button>
</form>
```

Custom form controls and selects are also supported.



Copy

```
<form class="form-inline">
  <label class="my-1 mr-2" for="inlineFormCustomSelectPref">Preference</label>
  <select class="custom-select my-1 mr-sm-2" id="inlineFormCustomSelectPref">
    <option selected>Choose...</option>
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </select>

  <div class="custom-control custom-checkbox my-1 mr-sm-2">
    <input type="checkbox" class="custom-control-input" id="customControlInline">
    <label class="custom-control-label" for="customControlInline">Remember my
preference</label>
  </div>

  <button type="submit" class="btn btn-primary my-1">Submit</button>
</form>
```

Alternatives to hidden labels

Assistive technologies such as screen readers will have trouble with your forms if you don't include a label for every input. For these inline forms, you can hide the labels using the `.sr-only` class. There are further alternative methods of providing a label for assistive technologies, such as the `aria-label`, `aria-labelledby` or `title` attribute. If none of these are present, assistive technologies may resort to using the `placeholder` attribute, if present, but note that use of `placeholder` as a replacement for other labelling methods is not advised.

Help text

Block-level help text in forms can be created using `.form-text` (previously known as `.help-block` in v3). Inline help text can be flexibly implemented using any inline HTML element and utility classes like `.text-muted`.

Associating help text with form controls

Help 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 help text when the user focuses or enters the control.

Help text below inputs can be styled with `.form-text`. This class includes `display: block` and adds some top margin 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.

Copy

```
<label for="inputPassword5">Password</label>
<input type="password" id="inputPassword5" class="form-control" aria-
describedby="passwordHelpBlock">
<small id="passwordHelpBlock" class="form-text text-muted">
  Your password must be 8-20 characters long, contain letters and numbers, and must not
contain spaces, special characters, or emoji.
</small>
```

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

Password Must be 8-20 characters long.

Copy

```
<form class="form-inline">
  <div class="form-group">
    <label for="inputPassword6">Password</label>
    <input type="password" id="inputPassword6" class="form-control mx-sm-3" aria-
describedby="passwordHelpInline">
    <small id="passwordHelpInline" class="text-muted">
      Must be 8-20 characters long.
    </small>
  </div>
</form>
```

Disabled forms

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

Copy

```
<input class="form-control" id="disabledInput" type="text" placeholder="Disabled input
here..." disabled>
```

Add the `disabled` attribute to a `<fieldset>` to disable all the controls within.

Disabled input

Disabled input

Disabled select menu

Disabled select

☐ Can't check this

Submit

Copy

```
<form>
  <fieldset disabled>
    <div class="form-group">
      <label for="disabledTextInput">Disabled input</label>
      <input type="text" id="disabledTextInput" class="form-control" placeholder="Disabled
input">
    </div>
    <div class="form-group">
      <label for="disabledSelect">Disabled select menu</label>
      <select id="disabledSelect" class="form-control">
        <option>Disabled select</option>
      </select>
    </div>
    <div class="form-group">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" id="disabledFieldsetCheck" disabled>
        <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>
```

Caveat with anchors

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-*">`, these will only be given a style of `pointer-events: none`. As noted in the section about disabled state for buttons (and specifically in the sub-section for anchor elements), this CSS property is not yet standardized and isn't fully supported in Internet Explorer 10. The anchor-based controls will also still be focusable and operable using the keyboard. 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.

Cross-browser compatibility

While Bootstrap will apply these styles in all browsers, Internet Explorer 11 and below don't fully support the `disabled` attribute on a `<fieldset>`. Use custom JavaScript to disable the fieldset in these browsers.

Validation

Provide valuable, actionable feedback to your users with HTML5 form validation–available in all our supported browsers. Choose from the browser default validation feedback, or implement custom messages with our built-in classes and starter JavaScript.

We are aware that currently the client-side custom validation styles and tooltips are not accessible, since they are not exposed to assistive technologies. While we work on a solution, we'd recommend either using the server-side option or the default browser validation method.

Input group validation

Input groups have difficulty with validation styles, unfortunately. Our recommendation is to place feedback messages as sibling elements of the `.input-group` that has `.is-{valid|invalid}`. Placing feedback messages within input groups breaks the `border-radius`. See this workaround.

How it works

Here’s how form validation works with Bootstrap:

- HTML form validation is applied via CSS’s two pseudo-classes, `:invalid` and `:valid`. It applies to `<input>`, `<select>`, and `<textarea>` elements.
- Bootstrap scopes the `:invalid` and `:valid` styles to parent `.was-validated` class, usually applied to the `<form>`. Otherwise, any required field without a value shows up as invalid on page load. This way, you may choose when to activate them (typically after form submission is attempted).
- To reset the appearance of the form (for instance, in the case of dynamic form submissions using AJAX), remove the `.was-validated` class from the `<form>` again after submission.
- As a fallback, `.is-invalid` and `.is-valid` classes may be used instead of the pseudo-classes for server side validation. They do not require a `.was-validated` parent class.
- Due to constraints in how CSS works, we cannot (at present) apply styles to a `<label>` that comes before a form control in the DOM without the help of custom JavaScript.
- All modern browsers support the constraint validation API, a series of JavaScript methods for validating form controls.
- Feedback messages may utilize the browser defaults (different for each browser, and unstylable via CSS) or our custom feedback styles with additional HTML and CSS.
- You may provide custom validity messages with `setCustomValidity` in JavaScript.

With that in mind, consider the following demos for our custom form validation styles, optional server side classes, and browser defaults.

Custom styles

For custom Bootstrap form validation messages, you’ll need to add the `novalidate` boolean attribute to your `<form>`. This disables the browser default feedback tooltips, but still provides access to the form validation APIs in JavaScript. Try to submit the form below; our JavaScript will intercept the submit button and relay feedback to you. When attempting to submit, you’ll see the `:invalid` and `:valid` styles applied to your form controls.

Custom feedback styles apply custom colors, borders, focus styles, and background icons to better communicate feedback. Background icons for `<select>`s are only available with `.custom-select`, and not `.form-control`.

First name

Mark

Last name

Otto

City

State

Choose...

Zip

☐ Agree to terms and conditions

Submit form

Copy

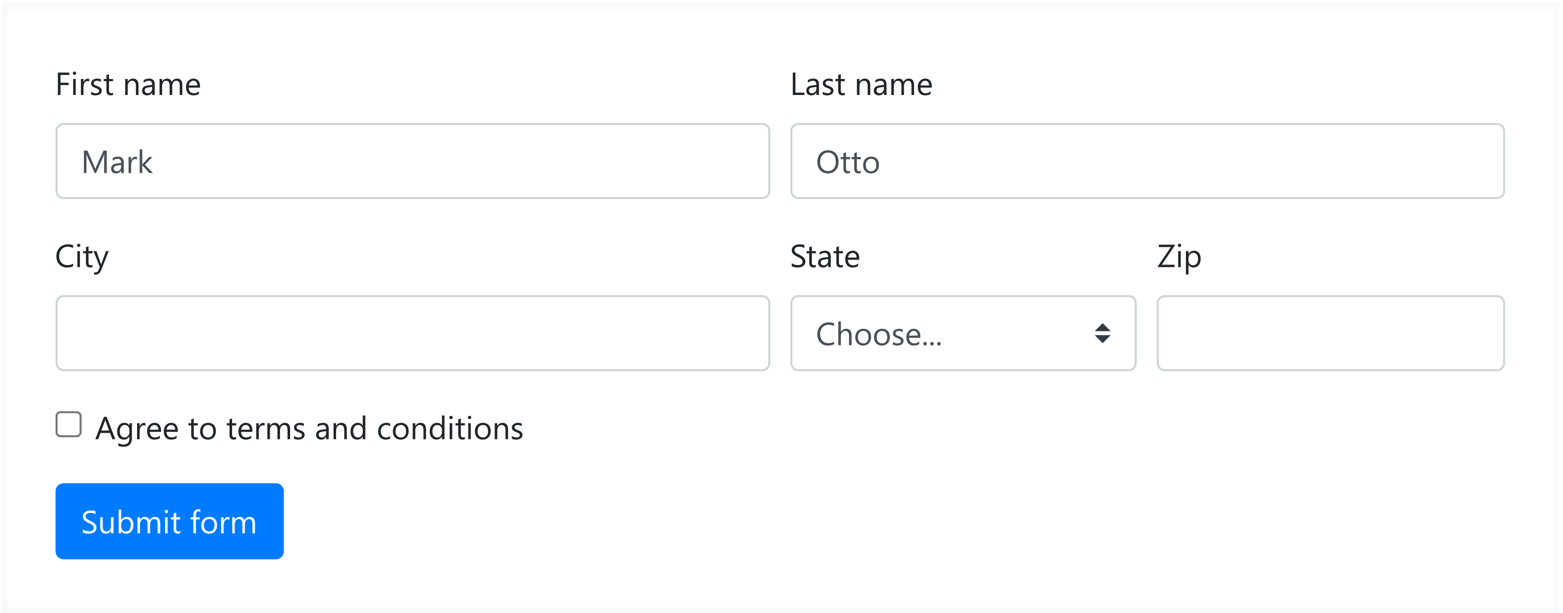
```
<form class="needs-validation" novalidate>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationCustom01">First name</label>
      <input type="text" class="form-control" id="validationCustom01" value="Mark" required>
      <div class="valid-feedback">
        Looks good!
      </div>
    </div>
    <div class="col-md-6 mb-3">
      <label for="validationCustom02">Last name</label>
      <input type="text" class="form-control" id="validationCustom02" value="Otto" required>
      <div class="valid-feedback">
        Looks good!
      </div>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationCustom03">City</label>
      <input type="text" class="form-control" id="validationCustom03" required>
      <div class="invalid-feedback">
        Please provide a valid city.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationCustom04">State</label>
      <select class="custom-select" id="validationCustom04" required>
        <option selected disabled value="">Choose...</option>
        <option>...</option>
      </select>
      <div class="invalid-feedback">
        Please select a valid state.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationCustom05">Zip</label>
      <input type="text" class="form-control" id="validationCustom05" required>
      <div class="invalid-feedback">
        Please provide a valid zip.
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" value="" id="invalidCheck" required>
      <label class="form-check-label" for="invalidCheck">
        Agree to terms and conditions
      </label>
      <div class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>

<script>
// Example starter JavaScript for disabling form submissions if there are invalid fields
(function() {
  'use strict';
  window.addEventListener('load', function() {
    // Fetch all the forms we want to apply custom Bootstrap validation styles to
    var forms = document.getElementsByClassName('needs-validation');
    // Loop over them and prevent submission
    var validation = Array.prototype.filter.call(forms, function(form) {
      form.addEventListener('submit', function(event) {
        if (form.checkValidity() === false) {
          event.preventDefault();
          event.stopPropagation();
        }
        form.classList.add('was-validated');
      }, false);
    });
  }, false);
})();
</script>
```

Browser defaults

Not interested in custom validation feedback messages or writing JavaScript to change form behaviors? All good, you can use the browser defaults. Try submitting the form below. Depending on your browser and OS, you'll see a slightly different style of feedback.

While these feedback styles cannot be styled with CSS, you can still customize the feedback text through JavaScript.



First name: Mark	Last name: Otto	
City:	State: Choose...	Zip:

☐ Agree to terms and conditions

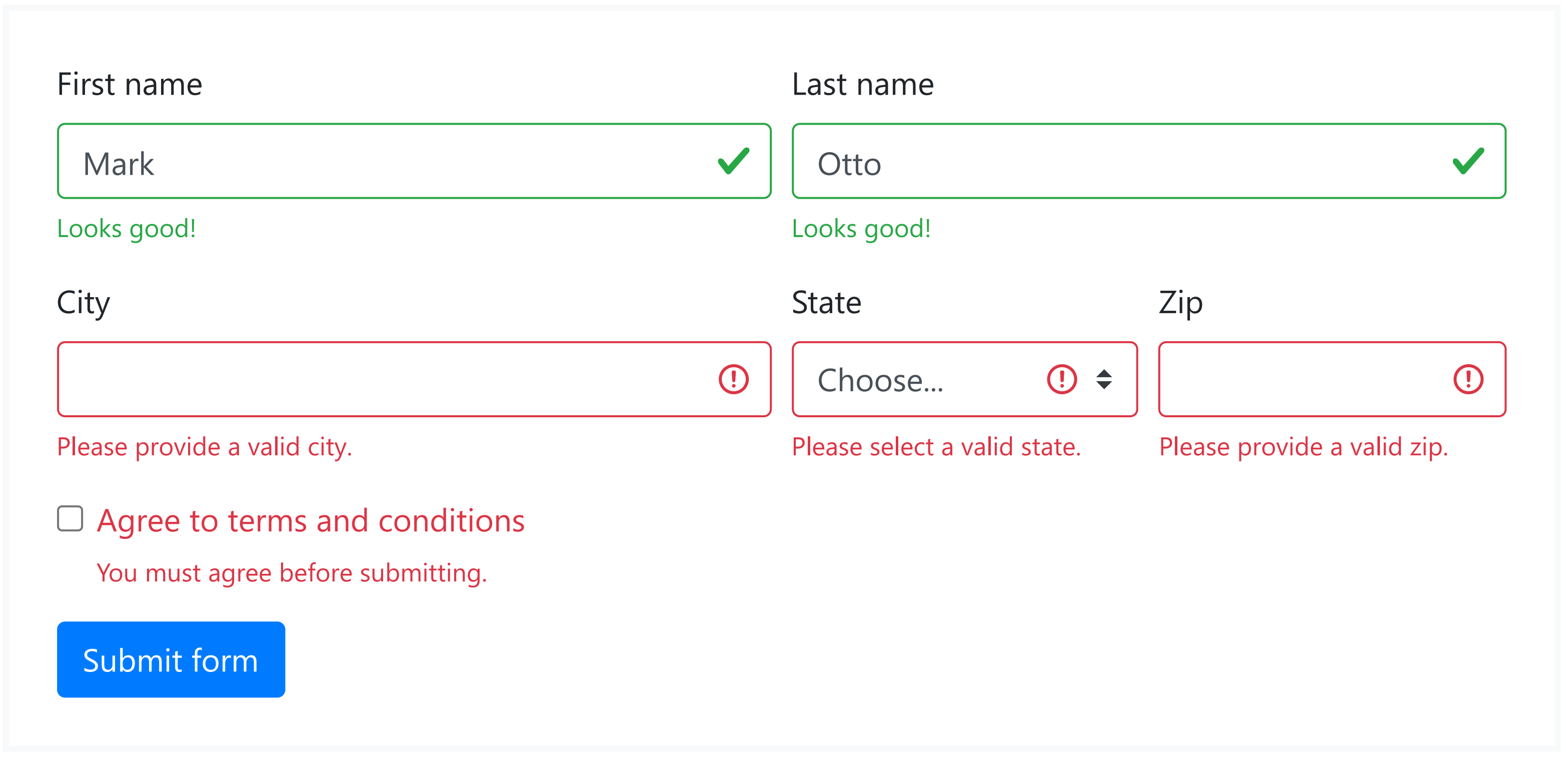
Copy

```
<form>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationDefault01">First name</label>
      <input type="text" class="form-control" id="validationDefault01" value="Mark" required>
    </div>
    <div class="col-md-6 mb-3">
      <label for="validationDefault02">Last name</label>
      <input type="text" class="form-control" id="validationDefault02" value="Otto" required>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationDefault03">City</label>
      <input type="text" class="form-control" id="validationDefault03" required>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationDefault04">State</label>
      <select class="custom-select" id="validationDefault04" required>
        <option selected disabled value="">Choose...</option>
        <option>...</option>
      </select>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationDefault05">Zip</label>
      <input type="text" class="form-control" id="validationDefault05" required>
    </div>
  </div>
  <div class="form-group">
    <div class="form-check">
      <input class="form-check-input" type="checkbox" value="" id="invalidCheck2" required>
      <label class="form-check-label" for="invalidCheck2">
        Agree to terms and conditions
      </label>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>
```

Server side

We recommend using client-side validation, but in case you require server-side validation, you can indicate invalid and valid form fields with `.is-invalid` and `.is-valid`. Note that `.invalid-feedback` is also supported with these classes.

For invalid fields, ensure that the invalid feedback/error message is associated with the relevant form field using `aria-describedby`. This attribute allows more than one `id` to be referenced, in case the field already points to additional form text.



First name	Last name	
Mark ✓	Otto ✓	
Looks good!	Looks good!	
City	State	Zip
	Choose...	
Please provide a valid city.	Please select a valid state.	Please provide a valid zip.

☐ Agree to terms and conditions

You must agree before submitting.

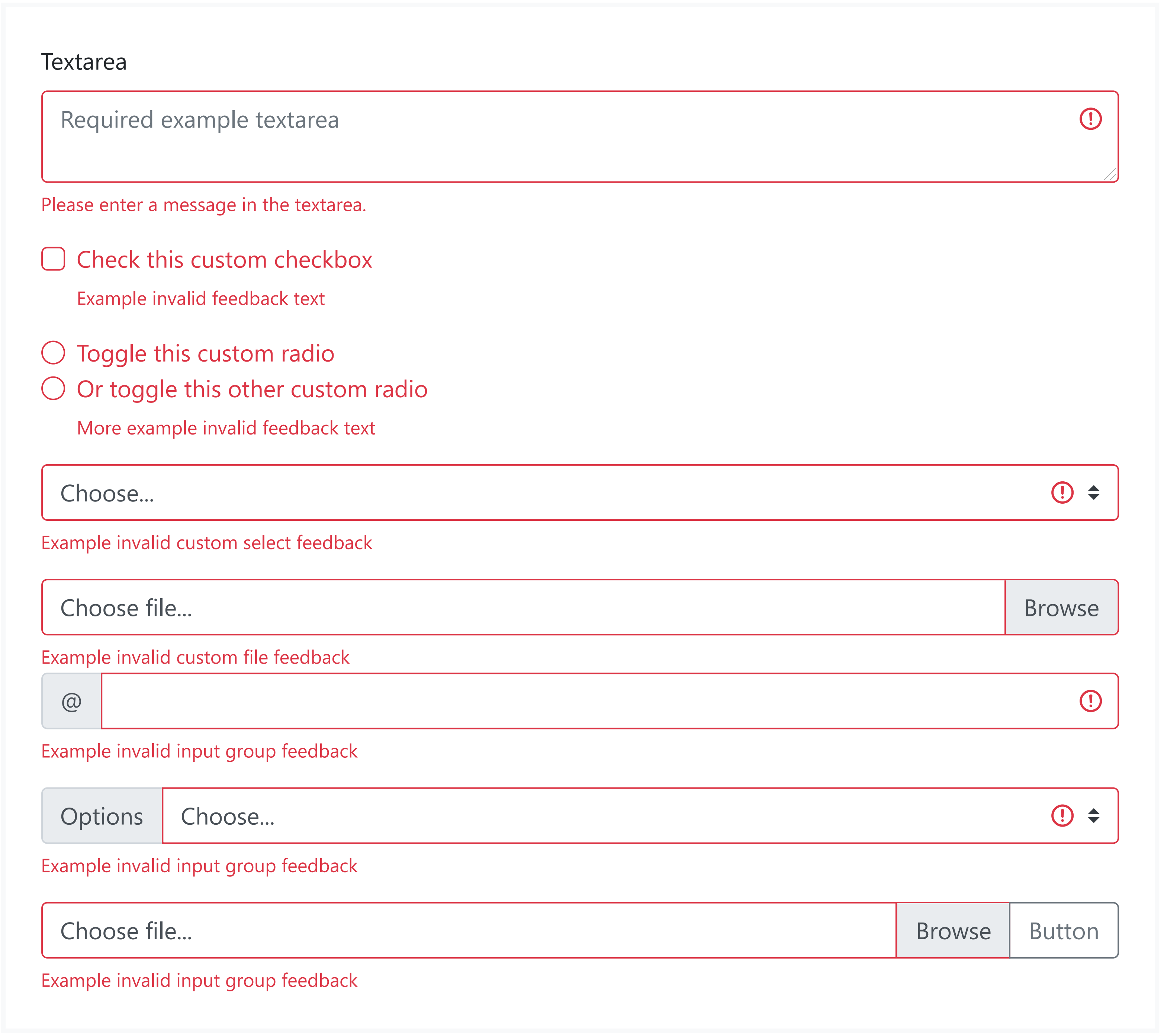
```html
<form>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationServer01">First name</label>
      <input type="text" class="form-control is-valid" id="validationServer01" value="Mark"
required>
      <div class="valid-feedback">
        Looks good!
      </div>
    </div>
    <div class="col-md-6 mb-3">
      <label for="validationServer02">Last name</label>
      <input type="text" class="form-control is-valid" id="validationServer02" value="Otto"
required>
      <div class="valid-feedback">
        Looks good!
      </div>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationServer03">City</label>
      <input type="text" class="form-control is-invalid" id="validationServer03" aria-
describedby="validationServer03Feedback" required>
      <div id="validationServer03Feedback" class="invalid-feedback">
        Please provide a valid city.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationServer04">State</label>
      <select class="custom-select is-invalid" id="validationServer04" aria-
describedby="validationServer04Feedback" required>
        <option selected disabled value="">Choose...</option>
        <option>...</option>
      </select>
      <div id="validationServer04Feedback" class="invalid-feedback">
        Please select a valid state.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationServer05">Zip</label>
      <input type="text" class="form-control is-invalid" id="validationServer05" aria-
describedby="validationServer05Feedback" required>
      <div id="validationServer05Feedback" class="invalid-feedback">
        Please provide a valid zip.
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="form-check">
      <input class="form-check-input is-invalid" type="checkbox" value="" id="invalidCheck3"
aria-describedby="invalidCheck3Feedback" required>
      <label class="form-check-label" for="invalidCheck3">
        Agree to terms and conditions
      </label>
      <div  id="invalidCheck3Feedback" class="invalid-feedback">
        You must agree before submitting.
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>
```

Supported elements

Validation styles are available for the following form controls and components:

- `<input>`s and `<textarea>`s with `.form-control`
- `<select>`s with `.form-control` or `.custom-select`
- `.form-check`s
- `.custom-checkbox`s and `.custom-radio`s

- `.custom-file`



Textarea

Required example textarea

Please enter a message in the textarea.

Check this custom checkbox

Example invalid feedback text

Toggle this custom radio

Or toggle this other custom radio

More example invalid feedback text

Choose...

Example invalid custom select feedback

Choose file... Browse

Example invalid custom file feedback

@

Example invalid input group feedback

Options Choose...

Example invalid input group feedback

Choose file... Browse Button

Example invalid input group feedback

Copy

```html
<form class="was-validated">
  <div class="mb-3">
    <label for="validationTextarea">Textarea</label>
    <textarea class="form-control is-invalid" id="validationTextarea" placeholder="Required
example textarea" required></textarea>
    <div class="invalid-feedback">
      Please enter a message in the textarea.
    </div>
  </div>

  <div class="custom-control custom-checkbox mb-3">
    <input type="checkbox" class="custom-control-input" id="customControlValidation1"
required>
    <label class="custom-control-label" for="customControlValidation1">Check this custom
checkbox</label>
    <div class="invalid-feedback">Example invalid feedback text</div>
  </div>

  <div class="custom-control custom-radio">
    <input type="radio" class="custom-control-input" id="customControlValidation2"
name="radio-stacked" required>
    <label class="custom-control-label" for="customControlValidation2">Toggle this custom
radio</label>
  </div>
  <div class="custom-control custom-radio mb-3">
    <input type="radio" class="custom-control-input" id="customControlValidation3"
name="radio-stacked" required>
    <label class="custom-control-label" for="customControlValidation3">Or toggle this other
custom radio</label>
    <div class="invalid-feedback">More example invalid feedback text</div>
  </div>

  <div class="mb-3">
    <select class="custom-select" required>
      <option value="">Choose...</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </select>
    <div class="invalid-feedback">Example invalid custom select feedback</div>
  </div>

  <div class="custom-file mb-3">
    <input type="file" class="custom-file-input" id="validatedCustomFile" required>
    <label class="custom-file-label" for="validatedCustomFile">Choose file...</label>
    <div class="invalid-feedback">Example invalid custom file feedback</div>
  </div>

  <div class="mb-3">
    <div class="input-group is-invalid">
      <div class="input-group-prepend">
        <span class="input-group-text" id="validatedInputGroupPrepend">@</span>
      </div>
      <input type="text" class="form-control is-invalid" aria-
describedby="validatedInputGroupPrepend" required>
    </div>
    <div class="invalid-feedback">
      Example invalid input group feedback
    </div>
  </div>

  <div class="mb-3">
    <div class="input-group is-invalid">
      <div class="input-group-prepend">
        <label class="input-group-text" for="validatedInputGroupSelect">Options</label>
      </div>
      <select class="custom-select" id="validatedInputGroupSelect" required>
        <option value="">Choose...</option>
        <option value="1">One</option>
        <option value="2">Two</option>
        <option value="3">Three</option>
      </select>
    </div>
    <div class="invalid-feedback">
      Example invalid input group feedback
    </div>
  </div>

  <div class="input-group is-invalid">
    <div class="custom-file">
      <input type="file" class="custom-file-input" id="validatedInputGroupCustomFile"
required>
      <label class="custom-file-label" for="validatedInputGroupCustomFile">Choose file...
</label>
    </div>
    <div class="input-group-append">
       <button class="btn btn-outline-secondary" type="button">Button</button>
    </div>
  </div>
  <div class="invalid-feedback">
    Example invalid input group feedback
  </div>
</form>
```

Tooltips

If your form layout allows it, you can swap the `.{valid|invalid}-feedback` classes for `.{valid|invalid}-tooltip` classes to display validation feedback in a styled tooltip. Be sure to have a parent with `position: relative` on it for tooltip positioning. In the example below, our column classes have this already, but your project may require an alternative setup.

First name	Last name	
Mark	Otto	
City	State	Zip
	Choose...	

Submit form

Copy

```html
<form class="needs-validation" novalidate>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationTooltip01">First name</label>
      <input type="text" class="form-control" id="validationTooltip01" value="Mark" required>
      <div class="valid-tooltip">
        Looks good!
      </div>
    </div>
    <div class="col-md-6 mb-3">
      <label for="validationTooltip02">Last name</label>
      <input type="text" class="form-control" id="validationTooltip02" value="Otto" required>
      <div class="valid-tooltip">
        Looks good!
      </div>
    </div>
  </div>
  <div class="form-row">
    <div class="col-md-6 mb-3">
      <label for="validationTooltip03">City</label>
      <input type="text" class="form-control" id="validationTooltip03" required>
      <div class="invalid-tooltip">
        Please provide a valid city.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationTooltip04">State</label>
      <select class="custom-select" id="validationTooltip04" required>
        <option selected disabled value="">Choose...</option>
        <option>...</option>
      </select>
      <div class="invalid-tooltip">
        Please select a valid state.
      </div>
    </div>
    <div class="col-md-3 mb-3">
      <label for="validationTooltip05">Zip</label>
      <input type="text" class="form-control" id="validationTooltip05" required>
      <div class="invalid-tooltip">
        Please provide a valid zip.
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Submit form</button>
</form>
```

Customizing

Validation states can be customized via Sass with the `$form-validation-states` map. Located in our `_variables.scss` file, this Sass map is looped over to generate the default `valid`/`invalid` validation states. Included is a nested map for customizing each state's color and icon. While no other states are supported by browsers, those using custom styles can easily add more complex form feedback.

Please note that we do not recommend customizing these values without also modifying the `form-validation-state` mixin.

Copy

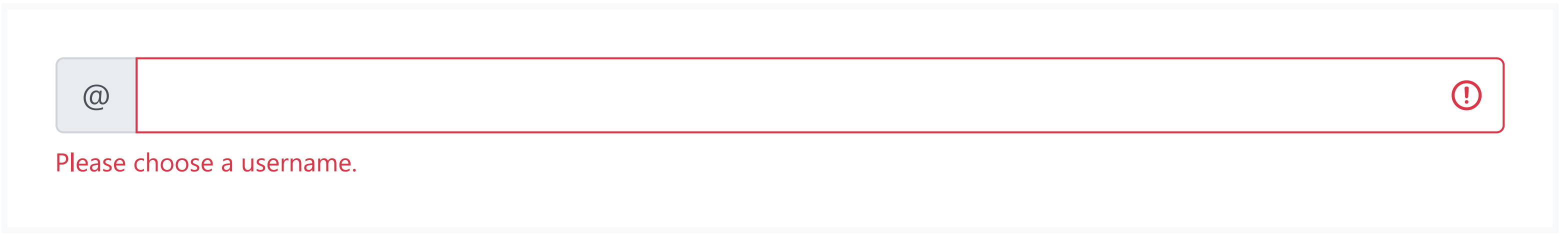
```scss
// Sass map from `_variables.scss`
// Override this and recompile your Sass to generate different states
$form-validation-states: map-merge(
  (
    "valid": (
      "color": $form-feedback-valid-color,
      "icon": $form-feedback-icon-valid
    ),
    "invalid": (
      "color": $form-feedback-invalid-color,
      "icon": $form-feedback-icon-invalid
    )
  ),
  $form-validation-states
);

// Loop from `_forms.scss`
// Any modifications to the above Sass map will be reflected in your compiled
// CSS via this loop.
@each $state, $data in $form-validation-states {
  @include form-validation-state($state, map-get($data, color), map-get($data, icon));
}
```

Input group validation workaround

We’re unable to resolve the broken `border-radius` of input groups with validation due to selector limitations, so manual overrides are required. When you’re using a standard input group and don’t customize the default border radius values, add `.rounded-right` to the elements with the broken `border-radius`.

```html
<div class="input-group">
  <div class="input-group-prepend">
    <span class="input-group-text">@</span>
  </div>
  <input type="text" class="form-control rounded-right" required>
  <div class="invalid-feedback">
    Please choose a username.
  </div>
</div>
```



Please choose a username.

When you are using a small or large input group or customizing the default `border-radius` values, add custom CSS to the element with the busted `border-radius`.

```css
/* Change values to match the radius of your form control */
.fix-rounded-right {
  border-top-right-radius: .2rem !important;
  border-bottom-right-radius: .2rem !important;
}
```

```html
<div class="input-group input-group-sm">
  <div class="input-group-prepend">
    <span class="input-group-text">@</span>
  </div>
  <input type="text" class="form-control fix-rounded-right" required>
  <div class="invalid-feedback">
    Please choose a username.
  </div>
</div>
```

@

Please choose a username.

Custom forms

For even more customization and cross browser consistency, use our completely custom form elements to replace the browser defaults. They're built on top of semantic and accessible markup, so they're solid replacements for any default form control.

Checkboxes and radios

Each checkbox and radio `<input>` and `<label>` pairing is wrapped in a `<div>` to create our custom control. Structurally, this is the same approach as our default `.form-check`.

We use the sibling selector (`~`) for all our `<input>` states—like `:checked`—to properly style our custom form indicator. When combined with the `.custom-control-label` class, we can also style the text for each item based on the `<input>`'s state.

We hide the default `<input>` with `opacity` and use the `.custom-control-label` to build a new custom form indicator in its place with `::before` and `::after`. Unfortunately we can't build a custom one from just the `<input>` because CSS's `content` doesn't work on that element.

In the checked states, we use **base64 embedded SVG icons** from Open Iconic. This provides us the best control for styling and positioning across browsers and devices.

Checkboxes

☐ Check this custom checkbox

Copy

```
<div class="custom-control custom-checkbox">
  <input type="checkbox" class="custom-control-input" id="customCheck1">
  <label class="custom-control-label" for="customCheck1">Check this custom checkbox</label>
</div>
```

Custom checkboxes can also utilize the `:indeterminate` pseudo class when manually set via JavaScript (there is no available HTML attribute for specifying it).

⊟ Check this custom checkbox

If you're using jQuery, something like this should suffice:

Copy

```
$('.your-checkbox').prop('indeterminate', true)
```

Radios

○ Toggle this custom radio
○ Or toggle this other custom radio

Copy

```
<div class="custom-control custom-radio">
  <input type="radio" id="customRadio1" name="customRadio" class="custom-control-input">
  <label class="custom-control-label" for="customRadio1">Toggle this custom radio</label>
</div>
<div class="custom-control custom-radio">
  <input type="radio" id="customRadio2" name="customRadio" class="custom-control-input">
  <label class="custom-control-label" for="customRadio2">Or toggle this other custom
radio</label>
</div>
```

Inline

○ Toggle this custom radio ○ Or toggle this other custom radio

Copy

```
<div class="custom-control custom-radio custom-control-inline">
  <input type="radio" id="customRadioInline1" name="customRadioInline1" class="custom-control-
input">
  <label class="custom-control-label" for="customRadioInline1">Toggle this custom
radio</label>
</div>
<div class="custom-control custom-radio custom-control-inline">
  <input type="radio" id="customRadioInline2" name="customRadioInline1" class="custom-control-
input">
  <label class="custom-control-label" for="customRadioInline2">Or toggle this other custom
radio</label>
</div>
```

Disabled

Custom checkboxes and radios can also be disabled. Add the `disabled` boolean attribute to the `<input>` and the custom indicator and label description will be automatically styled.

☐ Check this custom checkbox

○ Toggle this custom radio

Copy

```
<div class="custom-control custom-checkbox">
  <input type="checkbox" class="custom-control-input" id="customCheckDisabled1" disabled>
  <label class="custom-control-label" for="customCheckDisabled1">Check this custom
checkbox</label>
</div>

<div class="custom-control custom-radio">
  <input type="radio" name="radioDisabled" id="customRadioDisabled2" class="custom-control-
input" disabled>
  <label class="custom-control-label" for="customRadioDisabled2">Toggle this custom
radio</label>
</div>
```

Switches

A switch has the markup of a custom checkbox but uses the `.custom-switch` class to render a toggle switch. Switches also support the `disabled` attribute.

Toggle this switch element

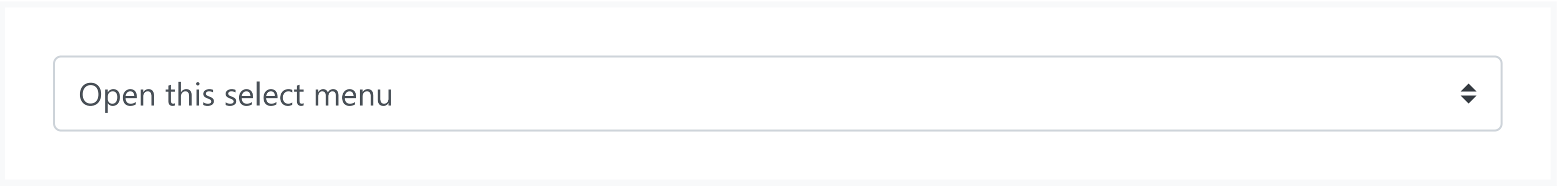
Disabled switch element

Copy

```
<div class="custom-control custom-switch">
  <input type="checkbox" class="custom-control-input" id="customSwitch1">
  <label class="custom-control-label" for="customSwitch1">Toggle this switch element</label>
</div>
<div class="custom-control custom-switch">
  <input type="checkbox" class="custom-control-input" disabled id="customSwitch2">
  <label class="custom-control-label" for="customSwitch2">Disabled switch element</label>
</div>
```

Select menu

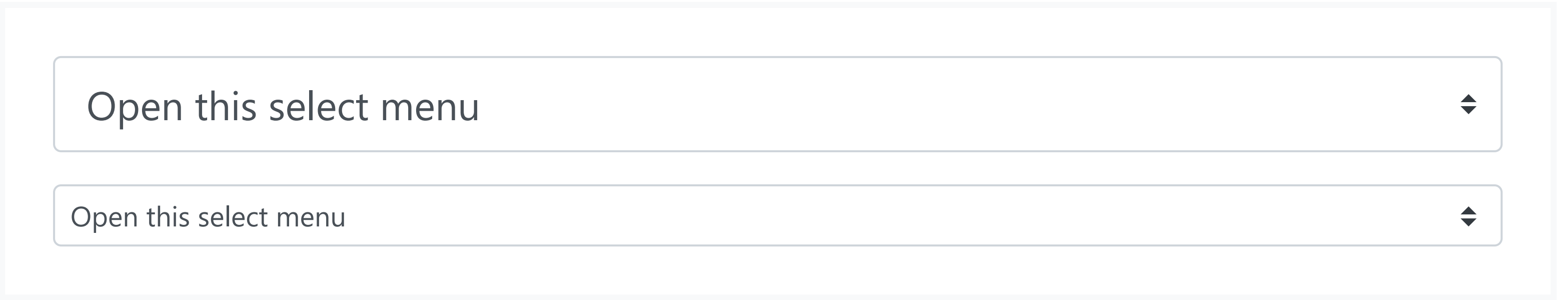
Custom `<select>` menus need only a custom class, `.custom-select` to trigger the custom styles. Custom styles are limited to the `<select>`'s initial appearance and cannot modify the `<option>`s due to browser limitations.



Copy

```
<select class="custom-select">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

You may also choose from small and large custom selects to match our similarly sized text inputs.

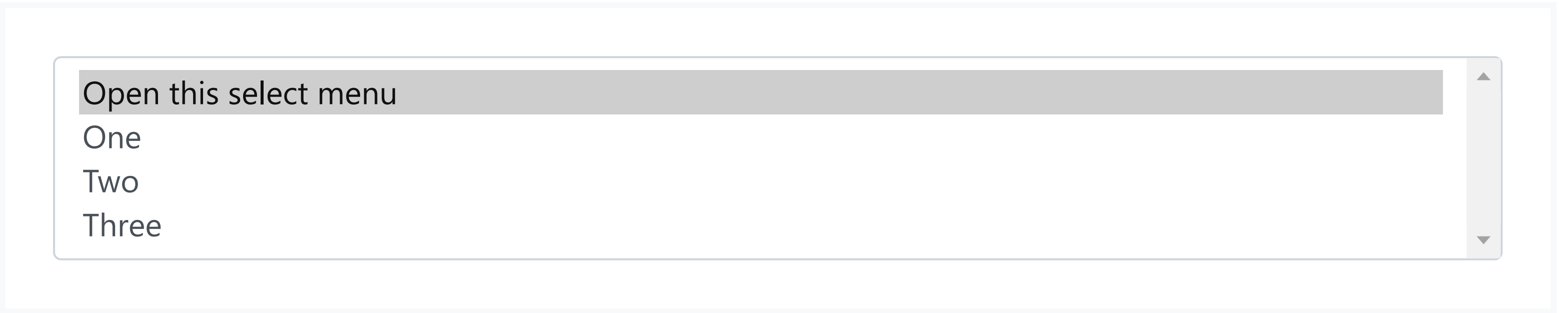


Copy

```
<select class="custom-select custom-select-lg mb-3">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>

<select class="custom-select custom-select-sm">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

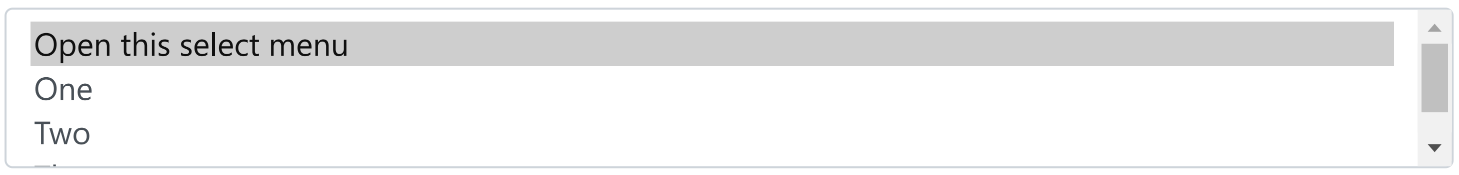
The `multiple` attribute is also supported:



Copy

```
<select class="custom-select" multiple>
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

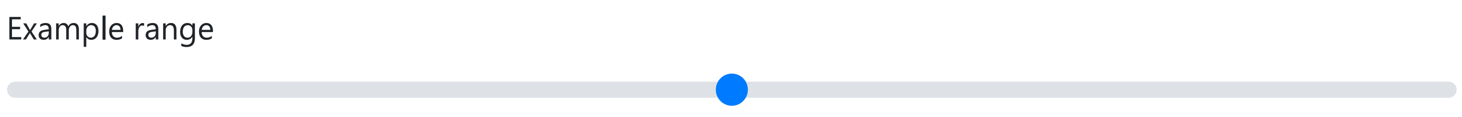
As is the `size` attribute:



```html
<select class="custom-select" size="3">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

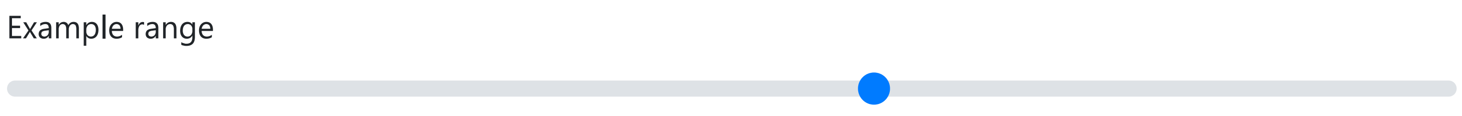
Range

Create custom `<input type="range">` controls with `.custom-range`. The track (the background) and thumb (the value) are both styled to appear the same across browsers. As only IE and Firefox support "filling" their track from the left or right of the thumb as a means to visually indicate progress, we do not currently support it.



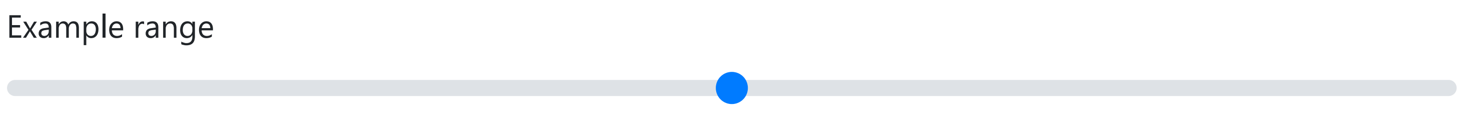
```html
<label for="customRange1">Example range</label>
<input type="range" class="custom-range" id="customRange1">
```

Range inputs have implicit values for `min` and `max`—`0` and `100`, respectively. You may specify new values for those using the `min` and `max` attributes.



```html
<label for="customRange2">Example range</label>
<input type="range" class="custom-range" min="0" max="5" id="customRange2">
```

By default, range inputs "snap" to integer values. To change this, you can specify a `step` value. In the example below, we double the number of steps by using `step="0.5"`.

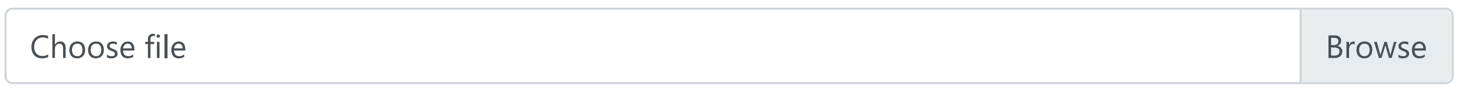


```html
<label for="customRange3">Example range</label>
<input type="range" class="custom-range" min="0" max="5" step="0.5" id="customRange3">
```

File browser

> The recommended plugin to animate custom file input: [bs-custom-file-input](), that's what we are using currently here in our docs.

The file input is the most gnarly of the bunch and requires additional JavaScript if you'd like to hook them up with functional *Choose file...* and selected file name text.



Copy

```
<div class="custom-file">
  <input type="file" class="custom-file-input" id="customFile">
  <label class="custom-file-label" for="customFile">Choose file</label>
</div>
```

We hide the default file `<input>` via `opacity` and instead style the `<label>`. The button is generated and positioned with `::after`. Lastly, we declare a `width` and `height` on the `<input>` for proper spacing for surrounding content.

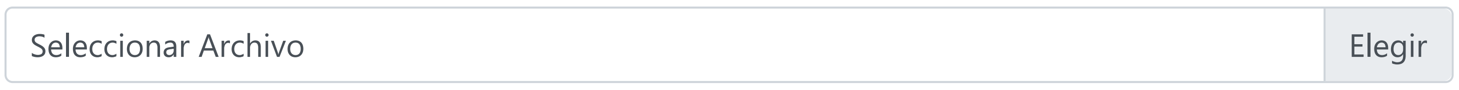
Translating or customizing the strings with SCSS

The `:lang()` pseudo-class is used to allow for translation of the "Browse" text into other languages. Override or add entries to the `$custom-file-text` Sass variable with the relevant language tag and localized strings. The English strings can be customized the same way. For example, here's how one might add a Spanish translation (Spanish's language code is `es`):

Copy

```
$custom-file-text: (
  en: "Browse",
  es: "Elegir"
);
```

Here's `lang(es)` in action on the custom file input for a Spanish translation:



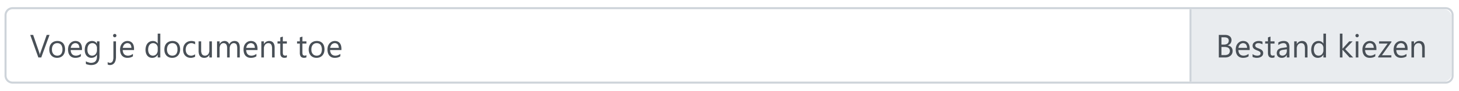
Copy

```
<div class="custom-file">
  <input type="file" class="custom-file-input" id="customFileLang" lang="es">
  <label class="custom-file-label" for="customFileLang">Seleccionar Archivo</label>
</div>
```

You'll need to set the language of your document (or subtree thereof) correctly in order for the correct text to be shown. This can be done using the `lang` attribute on the `<html>` element or the `Content-Language` HTTP header, among other methods.

Translating or customizing the strings with HTML

Bootstrap also provides a way to translate the "Browse" text in HTML with the `data-browse` attribute which can be added to the custom input label (example in Dutch):



Copy

```
<div class="custom-file">
  <input type="file" class="custom-file-input" id="customFileLangHTML">
  <label class="custom-file-label" for="customFileLangHTML" data-browse="Bestand kiezen">Voeg
je document toe</label>
</div>
```