Overview

Bootstrap's form controls expand on [our Rebooted form styles]({{< docsref "/content/reboot#forms" >}}) 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.

{{< example >}}

Email address
We'll never share your email with anyone else.
Password
Check me out
Submit
{{< /example >}}

Form text

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

{{< callout warning >}}

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. {{< /callout >}}

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.

{{< example >}} Password

Your password must be 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji.
{{< /example >}}

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.

{{< example >}}

Password

Must be 8-20 characters long.
{{< /example >}}

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
input here..." disabled>
```

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.

{{< example >}}

Disabled fieldset example

Disabled input Disabled input

Disabled select menu Disabled select

Can't check this

Submit

{{< /example >}}

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]({{< docsref "/components/buttons" >}}) and our form components. You'll find these frequently reassigned as values to other component-specific variables.

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