{{< callout warning >}} 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. {{< /callout >}}

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 `.form-select`, and not `.form-control`.

{{< example >}}

First name Mark
Looks good!
Last name Otto
Looks good!
Username
@
Please choose a username.
City
Please provide a valid city.
State Choose...

Please select a valid state.
Zip
Please provide a valid zip.
Agree to terms and conditions
You must agree before submitting.
Submit form
{{< /example >}}

{{< example lang="js" show_preview="false" >}} {{< js.inline >}} {{- readFile (path.Join "site/static/docs" .Site.Params.docs_version "assets/js/validate-forms.js") -}} {{< /js.inline >}} {{< /example >}}

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.

{{< example >}}

First name Mark
Last name Otto
Username
@
City
State Choose...
Zip
Agree to terms and conditions
Submit form
{{< /example >}}

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` (noting that this attribute allows more than one `id` to be referenced, in case the field already points to additional form text).

To fix issues with border radii, input groups require an additional `.has-validation` class.

{{< example >}}

First name Mark
Looks good!
Last name Otto
Looks good!
Username
@

Please choose a username.
City
Please provide a valid city.
State Choose...
Please select a valid state.
Zip
Please provide a valid zip.
Agree to terms and conditions
You must agree before submitting.
Submit form
{{< /example >}}

Supported elements

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

- `<input>`s and `<textarea>`s with `.form-control` (including up to one `.form-control` in input groups)
- `<select>`s with `.form-select`
- `.form-check`s

{{< example >}}

Textarea Required example textarea
Please enter a message in the textarea.
Check this checkbox
Example invalid feedback text
Toggle this radio
Or toggle this other radio
More example invalid feedback text
Open this select menu
Example invalid select feedback
Choose File No file chosen
Example invalid form file feedback
Submit form
{{< /example >}}

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.

{{< example >}}

First name Mark
Looks good!
Last name Otto
Looks good!

Username
@
Please choose a unique and valid username.
City
Please provide a valid city.
State Choose...
Please select a valid state.
Zip
Please provide a valid zip.
Submit form
{{< /example >}}

Sass

Variables

{{< scss-docs name="form-feedback-variables" file="scss/_variables.scss" >}}

Mixins

Two mixins are combined together, through our loop, to generate our form validation feedback styles.

{{< scss-docs name="form-validation-mixins" file="scss/mixins/_forms.scss" >}}

Map

This is the validation Sass map from `_variables.scss`. Override or extend this to generate different or additional states.

{{< scss-docs name="form-validation-states" file="scss/_variables.scss" >}}

Maps of `$form-validation-states` can contain three optional parameters to override tooltips and focus styles.

Loop

Used to iterate over `$form-validation-states` map values to generate our validation styles. Any modifications to the above Sass map will be reflected in your compiled CSS via this loop.

{{< scss-docs name="form-validation-states-loop" file="scss/forms/_validation.scss" >}}

Customizing

Validation states can be customized via Sass with the `$form-validation-states` map. Located in our `_variables.scss` file, this Sass map is how we generate the default `valid` / `invalid` validation states. Included is a nested map for customizing each state's color, icon, tooltip color, and focus shadow. 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 `$form-validation-states` values without also modifying the `form-validation-state` mixin**.