Help users enter the right data in forms

- Why should you validate your forms?
- Help users from accidentally missing required fields
- Help the user enter the correct format
- Communicate your validation rules
- Pattern attribute
- Add styles
- How to style a required form field
- Style invalid form controls
- Validation with JavaScript {: #javascript}
  - Provide meaningful error messages
- Resources

Browsers have built-in features for validation to check that users have entered data in the correct format. You can activate these features by using the correct elements and attributes. On top of that, you can enhance form validation with CSS and JavaScript.

This module is about form validation on the frontend. You must also make sure to validate data before storing or sharing it on your backend server: find out more.

Why should you validate your forms?

You learned in the previous module how to help users avoid having to repeatedly re-enter information in forms. How can you help users enter data that's valid?

It's frustrating to fill out a form without knowing which fields are required, or the constraints of those fields. For example, you enter a username and submit a form—only to find out that usernames must have at least eight characters.

You can help users with that by defining validation rules and communicating them.

Help users from accidentally missing required fields

You can use HTML to specify the correct format and constraints for data entered in your forms. You also need to specify which fields are mandatory.

https://codepen.io/web-dot-dev/pen/1e3e72e4376292ab80ecd8955529806b

Try to submit this form without entering any data. Do you see an error message attached to the `<input>` telling you that the field is required?

This happens because of the `required` attribute.

```
<label for="name">Name (required)</label>
<input required type="text" id="name" name="name">
```

Different browsers use different text for error messages. The wording is also dependent on the browser language of the user. If you use Safari and your preferred browser language is German, you get a different error message than a user in Chrome with English as their chosen browser language.

Find out how to change the default error messages with JavaScript later in this module.

You already learned that you can use many more types, for example, `type="email"`. Let's have a look at a required email `<input>`.

https://codepen.io/web-dot-dev/pen/061add379d5159560d2bc6bc037d82da

Try to submit this form without entering any data. Is there any difference from the demo before? Now insert your name in the email field and try to submit. You see a different error message. How is that possible? You get a different message because the value you entered isn't a valid email address.

The `required` attribute tells the browser that the field is mandatory. The browser also tests if the entered data matches the format of the `type`. The email field shown in the example is only valid if

it's not empty and if the entered data is a valid email address.

Help the user enter the correct format

You learned how to make a field mandatory. How would you instruct the browser that a user must enter at least eight characters for a form field?

https://codepen.io/web-dot-dev/pen/0901d502bbe2cc7c4aa0bd73068ddd96

Give the demo a try. After your change, you should not be able to submit the form if you enter less than eight characters.

Toggle answer

```
<label for="password">Password (required)</label>
<input required minlength="8" type="password" id="password" name="password">
```

The name of the attribute is `minlength`. Set the value to `8` and you have the desired validation rule. If you want the opposite, use `maxlength`.

For numerical input types use `min` and `max` to achieve the same result.

Communicate your validation rules

```
<label for="password">Password (required)</label>
<input required minlength="8" type="password" id="password"
  name="password" aria-describedby="password-minlength">
<div id="password-minlength">Enter at least eight characters</div>
```

Make sure all users understand your validation rules. For this, connect the form control with an element that explains the rules. To do so, add an `aria-describedby` attribute to the element with the `id` of the form.

Pattern attribute

Sometimes you want to define more advanced validation rules. Again, you can use an HTML attribute. It's called `pattern`, and you can define a regular expression as the value.

```
<label for="animal">What is your favorite animal? (required)</label>
<input required pattern="[a-z]{2,20}" type="text" id="animal" name="animal">
```

Here, only lowercase letters are allowed; the user has to enter at least two characters, and not more than twenty.

How would you change the `pattern` to also allow uppercase letters? Try it out.

Toggle answer

The correct answer is `pattern="[a-zA-Z]{2,20}"`.

Add styles

You have now learned how to add validation in HTML. Wouldn't it be great if you could also style form controls based on the validation status? This is possible with CSS.

How to style a required form field

Show the user that a field is mandatory before they interact with your form.

https://codepen.io/web-dot-dev/pen/5a5de1e7dbdcdba86d19e30d42291a27

You can style `required` fields with the `:required` CSS pseudo class.

```
input:required {
  border: 2px solid;
}
```

Style invalid form controls

Do you remember what happens if data entered by the user is invalid? The error message attached to the form control appears. Wouldn't it be great to adapt the appearance of the element when this happens?

https://codepen.io/web-dot-dev/pen/d6bb0f27d1faac3ffd3b594a81fad396

You can use the `:invalid` pseudo-class to add styles to invalid form controls. In addition, there is also the `:valid` pseudo-class for styling valid form elements.

Sometimes designers color the invalid state red, using the `:invalid` pseudo-class. However, to communicate error or success you should never rely only on color. For people with red-green color blindness a green and a red border look almost the same. It's impossible for them to see if there was an error or success. Add text or an icon to make it obvious.

There are more ways to adapt your styles based on validation. In the module about CSS you will learn more about styling forms.

In practice `:invalid` is tricky to work with. Invalid form fields are already marked as `:invalid` before user interaction, which may confuse users. The `:user-invalid` pseudo-class solves this issue, as the styles are only applied after user interaction.

Learn more about `:user-invalid`.

Validation with JavaScript {: #javascript}

To further enhance validation of your forms you can use the JavaScript Constraint Validation API.

Provide meaningful error messages

You learned before that error messages are not identical in every browser. How can you show the same message to everyone?

To achieve this, use the `setCustomValidity()` method of the Constraint Validation API. Let's see how this works.

```
const nameInput = document.querySelector('[name="name"]');

nameInput.addEventListener('invalid', () => {
    nameInput.setCustomValidity('Please enter your name.');
 });
```

Query the element where you want to set the custom error message. Listen to the `invalid` event of your defined element. There you set the message with `setCustomValidity()`. This example shows the message `Please enter your name.` if the input is invalid.

https://codepen.io/web-dot-dev/pen/7ea31257d7cd8fc28792c7f5cdaba97b

Open the demo in different browsers, you should see the same message everywhere. Now, try to remove the JavaScript and try again. You see the default error messages again.

There is much more you can do with the Constraint Validation API. You'll find a detailed look at using validation with JavaScript in a later module.

How to validate in real-time You can add real-time validation in JavaScript by listening to the `onblur` event of a form control, and validate the input immediately when a user leaves a form field.

https://codepen.io/web-dot-dev/pen/b7ed22a0539f9beef4dc03380f51f224

Click the form field in the demo, enter "web" and click somewhere else on the page. You see the native error message for `minlength` below the form field.

Learn more about implementing real-time validation with JavaScript in an upcoming module.

Resources

- Disable the payment button once the form is submitted
- Add CSS to make the form work better

□□□ #3
□ □□□ □□□ 2026-10-31 22:50:44 UTC
□ cocobook-sync □□□ 2026-11-04 17:39:20 UTC