JavaScript

- Help users fill in additional form controls
- Ensure users can submit a form without leaving a page
- Validation with JavaScript
  - Ensure error messages align with your site style and tone
  - Ensure users are notified about errors in real time
  - Ensure users can see the password they entered
- Resources

You can use JavaScript to respond to user interactions on your form, reveal additional form fields, submit a form, and much more.

Help users fill in additional form controls

Imagine that you built a survey form. After a user selects one option, you want to show an additional `<input>` to ask a specific question related to the selection. How can you only show the relevant `<input>` element?

{% Codepen { user: 'web-dot-dev', id: '8e1e7a38790c75c267a978efa1d8e937', height: 350 } %}

You can use JavaScript to reveal an `<input>` only when the associated `<input type="radio">` is currently selected.

```
if (event.target.checked) {
    // show additional field
} else {
   // hide additional field
}
```

{% Aside 'caution' %} Make sure your form is still usable if JavaScript isn't available. The core experience should be the same for all users, with JavaScript used only as an enhancement. {% endAside %}

Let's look at the JavaScript code for the demo. Have you noticed the `aria-controls`, and `aria-expanded` attributes? Use these ARIA attributes to help screen reader users understand when an additional form control is shown or hidden.

{% Aside 'caution' %} The `aria-expanded` attribute for `<input type="radio">` isn't correctly announced by every screen reader. {% endAside %}

Ensure users can submit a form without leaving a page

Imagine you have a comment form. When a reader adds a comment and submits the form, it would be ideal if they could immediately see the comment without a page refresh.

To achieve this, listen to the `onsubmit` event, then use `event.preventDefault()` to prevent the default behavior, and send the `FormData` using the Fetch API. {% BrowserCompat 'api.fetch' %}

{% Aside %} `FormData` is a set of key/value pairs representing form fields and their values. You can add all your form fields to the `FormData` object, or only send some form fields and their values. {% endAside %}

{% Codepen { user: 'web-dot-dev', id: 'b499a861d7b1fee74464a690cf0ff97e', height: 400, tab: 'js,result' } %}

Your backend script can check if a `POST` request appears to be from the browser (using the `action` attribute of a form element, where `method="post"`) or from JavaScript, such as a `fetch()` request.

```
if (req.xhr || req.headers.accept.indexOf('json') !== -1) {
    // return JSON
} else {
    // return HTML
}
```

Always notify screen reader users about dynamic content changes. Add an element with the `aria-live="polite"` attribute to your HTML, and update the content of the element after a change. For example, update the text to 'Your comment was successfully posted', after a user submits a comment.

Learn more about ARIA live regions.

Validation with JavaScript

Ensure error messages align with your site style and tone

The wording of default error messages differs between browsers. How can you make sure the same message is shown to all users, and that the message aligns with your site's style and tone? Use the `setCustomValidity()` method of the Constraint Validation API to define your own error messages.

{% Codepen { user: 'web-dot-dev', id: '7ea31257d7cd8fc28792c7f5cdaba97b', height: 300, tab: 'js,result' } %}

{% Aside %} Make sure to translate and localize your error messages if you have a multi-language website. {% endAside %}

Ensure users are notified about errors in real time

The built-in HTML features for form validation are great for notifying users about invalid form fields before the data is sent to your backend. Wouldn't it be great to notify users as soon as they leave a form field?

{% Codepen { user: 'web-dot-dev', id: 'b7ed22a0539f9beef4dc03380f51f224', height: 300, tab: 'js,result' } %}

Listen for the `blur` event which fires when an element loses focus, and use the `ValidityState` interface to detect if a form control is invalid.

Ensure users can see the password they entered

The text entered for `<input type="password">` is obscured by default, to respect the privacy of users. Help users to enter their password, by showing a `<button>` to toggle the visibility of the entered text.

{% Codepen { user: 'web-dot-dev', id: 'bd8577c5380c436dba2788c7a2c8652a', height: 300 } %}

Try out the demo. Toggle the visibility of the entered text, by using the **Show Password** `<button>`. How does this work? Clicking on **Show Password**, changes the `type` attribute of the password field from `type="password"` to `type="text"`, and the `<button>` text changes to 'Hide Password'.

{% Aside 'codelab' %} Try to improve the **Show Password** `<button>`. Where would you position the `<button>`? Could you use only an icon and still make sure the `<button>` is accessible? {% endAside %}

It's important to make the **Show Password** button accessible. Connect the `<button>` with the `<input type="password">` using the `aria-controls` attribute.

{% Aside 'caution' %} The `aria-controls` attribute is not supported by every screen reader, but for those that do support it, it's a great enhancement. {% endAside %}

To notify screen reader users if the password is currently shown or hidden, use a hidden element with `aria-live="polite"`, and change its text accordingly. It's important to enable screen reader users to know when a password is displayed and visible to someone else looking at their screen.

```
<span class="visually-hidden" aria-live="polite">
    <!-- Dynamically change this text with JavaScript -->
</span>
```

{% Aside %} Microsoft Edge shows a built-in password reveal control for `<input type="password">`. To prevent having two reveal buttons, hide the built-in control with:

```
::-ms-reveal {
  display: none
}
```

Learn more about customizing the password reveal button. {% endAside %}

Learn more about implementing a show password option.

Resources

- FormData
- Constraint Validation API
- `<input type="password">`

□□□ #2
□ □□□ □□□ 2026-10-31 22:50:19 UTC
□ □□□ □□□ 2026-10-31 22:50:21 UTC