You Don't Need JS for That

Aubrey Sambor

About me

Aubrey Sambor

- Lead Front-end Developer
- Over 25 years of development experience
- Loves CSS, knitting, beer, and coffee

Find me on...

- Mastodon labyrinth.social/@starshaped
- Website aubreysambor.com
- Drupal.org starshaped

What will I talk about today?

9 new HTML and CSS features that don't use Javascript (and will make your life easier):

1. Dialogs and popovers
2. Animations
3. Form improvements

Let's go!

Dialogs and popovers

1. The <dialog> element

- Used to create modal or non-modal dialog boxes (but is a modal dialog by default)
- Baseline widely available (can be used today!)
- Accessible out of the box!

Modal vs. Non-modal

Modal

- Can't be light dismissed (clicking outside of the dialog to close it)
- Traps focus
- Used for important alerts that usually need to be confirmed

Non-modal

- Can be light dismissed
- Shouldn't interrupt user's tasks on the page
- Used for less important notifications such as a toast

The <dialog> element in action

```
01. <dialog id="my-fancy-dialog">
02.   <p>Dialog content</p>
03.   <form method="dialog">
04.     <button>Close</button>
05.   </form>
06. </dialog>
```

2. The Popover API

- A way to display popover content on a webpage
- Can be invoked using either Javascript or HTML commands (obviously, I'll be focusing on the HTML commands!)
- Popovers are *always* non-modal!
- Used for dismissible alerts, toast messages, tooltips, and much more
- Baseline 2025 so can be used in all modern browsers

Popovers in action

```
01. <button popovertarget="mypopover">Toggle popover</button>
02. <div id="mypopover" popover>Popover content</div>
```

How do you, you know, *open* a dialog element?

3. The Invoker Commands API

- Provides a way to assign behaviors to buttons without the use of Javascript!
- Can toggle dialogs and popovers just by adding attributes to your HTML
- Baseline 2025 so can be used in all modern browsers

Invoker Commands

`commandfor`

Attribute on a `button` element that controls an element with the same ID

`command`

Specifies an action to be performed on an element controlled by a button with the `commandfor` attribute

<command> options

- `show-modal`: Command to show a modal `<dialog>` element.
- `close`: Closes a modal `<dialog>`
- `toggle-popover`: Opens and closes a popover. (open-popover and close-popover also exist as commands.)

Note that a command to show a *non*-modal dialog does not exist!

The Invoker Commands API in action

```
<button commandfor="my-fancy-dialog" command="show-modal">
  Open dialog</button>
<dialog id="my-fancy-dialog">
  <p>Dialog content</p>
  <button commandfor="my-fancy-dialog" command="close">
    Close dialog</button>
</dialog>
```

4. Interest invokers

- Similar to the Invoker Commands API, though these are invoked by hovering over the button ("indicating interest") instead of clicking
- Can only be used on elements with the `popover` attribute
- Differs from the Invoker Commands API in that the invoker can be a `button`, `a`, or `area` element
- Supported only in Chromium browsers for now

Interest invokers in action

```
01. <button interestfor="mypopover">Show popover</button>
02. <div id="mypopover" popover>Popover content</div>
```

5. Anchor Positioning

- A way to anchor one element to another, such as a dropdown
- Baseline 2026, newly available across all major browsers
- Place `anchor-name` on the anchor element, and `position-anchor` on the anchored content.
- Two different techniques:
  - `position-area`
  - `anchor()`

Anchor positioning basics

```
<p class="anchor">I am an anchor!</p>
<div class="content">I am the anchored text!</div>

.anchor {
  anchor-name: --my-fun-anchor;
}
.content {
  position-anchor: --my-fun-anchor;
  position: absolute;
}
```

position-area

- Positions the anchored element relative to a 3x3 grid surrounding the anchor.
- Can use logical properties as well as physical properties

Examples

```
position-area: top right; <!--physical properties-->
position-area: block-end inline-end; <!--logical properties-->
```

position-area in action

```
01. .anchor {
02.   anchor-name: --my-fun-anchor;
03. }
04. .content {
05.   position-anchor: --my-fun-anchor;
06.   position: absolute;
07.   position-area: block-end inline-end;
08.   position-try: block-end inline-start;
09. }
```

Wait. What is position-try?

position-try

Allows the element to 'try' another position if the original one doesn't work

Example

```
position-try: block-end inline-start;
```

anchor()

- Positions the anchored element relative to the edges of the anchor.
- Can also use logical properties or physical properties for this

Example

```
.content {
  top: anchor(bottom);
  left: anchor(left);
}
```

Animations

6. View Transitions

- Adds a transition when navigating from one page to the other
- Only takes 3 lines of CSS
- View transitions within the same page require Javascript so I won't be covering that here
- Cross-document view transitions supported in all major browsers except Firefox (womp womp)

View transitions in action

```
@view-transition {
  navigation: auto;
}
```

Seriously, this is all the code you need!

7. interpolate-size

- Finally, a way to animate width and/or height of an HTML element!
- Mainly used to open/close or show/hide a container such as a `<details>` element
- Supported in Chromium browsers only for now 😭

interpolate-size in action

```
:root {
  interpolate-size: allow-keywords;
}
```

`allow-keywords`: This needs to be set to enable animations between two sizes (usually between 0 and a width or height)

Animating a <details> element

```
details::details-content {
  block-size: 0;
  overflow: clip;
  transition: content-visibility 400ms, block-size 400ms;
  transition-behavior: allow-discrete;
}
details[open]::details-content {
  block-size: auto;
}
```

Animating a <details> element

There is a *lot* going on here!

- `::details-content`: A pseudo-element that targets the contents of a `<details>` element.
- `content-visibility` and `block-size`: Sets the smooth animations for opening and closing the `<details>` element.
- `transition-behavior: allow-discrete`: Allows the transition of 'discrete' animated properties (such as width and height).

Form improvements

8. field-sizing

- Controls the size of form elements such as `<input>`, `<textarea>`, and `<select>`
- Resizes `<textarea>` elements as you type, no Javascript required!
- Just 3 lines of code!
- Baseline 2026, supported in all modern browsers

field-sizing in action

```
input,
select,
textarea {
  field-sizing: content;
}
```

9. Customizable select

- A new way to style the `<select>` element without having to use a Javascript library!
- Allows the ability to style the select area, list of select options, the select arrow, and more!
- Supported in Chromium browsers only

The <selectedcontent> element

`<selectedcontent>` styles the currently selected option.

```
<select>
  <button>
    <selectedcontent></selectedcontent>
  </button>
  <option value="">Default value</option>
  ...
</select>
```

Making a <select> customizable

- `::picker(select)`: Targets the contents of the `<select>` element
- `appearance: base-select`: Sets the `<select>` element to be customizable

```
select,
::picker(select) {
  appearance: base-select;
}
```

::checkmark and :checked styles

Set within an `<option>` element!

```
&::checkmark {
  content: "☑";
}
&:checked {
  font-weight: bold;
}
```

Styling the picker arrow

```
select::picker-icon {
  color: #ff1493;
  transition: rotate 0.25s;
}
select:open::picker-icon {
  rotate: 180deg;
}
```

More about customizable selects

- `<option>` elements within a customizable select have a display of `flex` by default, so can be styled using flexbox
- `::checkmark` and `::picker-icon` not included in accessibility tree
- CSS animations can be added for opening and closing the `<select>` element
- Can use our old friend anchor positioning to position the open select container in different ways!

In review...

With all the advances in HTML and CSS in the past few years, you don't need to reach for JavaScript for things such as:

- Dropdowns using the Popover API, the Invoker Commands API, and anchor positioning
- Animations using cross document view transitions and interpolate-size
- Form improvements such as field sizing and customizable select elements

Further reading

- https://matuzo.at/blog/2026/better-defaults-for-popovers
- https://www.htmhell.dev/adventcalendar/2025/7/
- https://webkit.org/blog/17240/a-gentle-introduction-to-anchor-positioning
- https://solid-web.com/css-interpolate-size-tutorial
- https://blog.gitbutler.com/the-great-css-expansion
- https://frontendmasters.com/blog/what-you-need-to-know-about-modern-css-2025-edition

About me

Aubrey Sambor

- Lead Front-end Developer
- Over 25 years of development experience
- Loves CSS, knitting, beer, and coffee

Find me on...

- Mastodon labyrinth.social/@starshaped
- Website aubreysambor.com
- Drupal.org starshaped

Thank you!