Styling form controls

- Help users select an option
  - The <select> element
  - Checkboxes and Radio buttons
- How to use your site's colors for form controls
- Resources

In this module you learn how to style form controls, and how to match your other site styles.

Styling HTML form controls can be a challenge, but you should still use built-in elements wherever possible. Elements such as `<input>` and `<button>` are widely supported across browsers and platforms, and have built-in features that enhance usability and accessibility, which you don't need to implement yourself. Using a `<div>` instead doesn't provide these benefits.

Help users select an option

The `<select>` element

The default styles of a `<select>` element don't look great, and the appearance is inconsistent between browsers.

https://codepen.io/web-dot-dev/pen/eea33ada7ec976aa5791a5cd14756c34

You can also use an `<input>` in combination with the `<datalist>` element. This gives you a combination of a text field and a list of `<option>` elements. You can see examples of `<datalist>` here.

First, let's change the arrows.

```
select {
    -moz-appearance: none;
    -webkit-appearance: none;
    appearance: none;
```

```
    background-color: #fff;
    background-image: url(arrow.png);
    background-repeat: no-repeat;
    background-position: right .7em top 50%, 0 0;
    background-size: .65em auto;
}
```

To remove the default arrows of a `<select>` element, use the CSS `appearance` property. To show the arrow of your choice, define the arrow as a `background`.

To ensure the best cross-browser compatibility, you must also include the prefixed versions of `appearance`: `-moz-appearance` and `-webkit-appearance`.

Learn more about vendor prefixes.

You should also change the `font-size` to at least `1rem` (which for most browsers has a default value of 16px) for your `<select>` element. Doing so will prevent a page zoom on iOS Safari when the form control is focused.

You can, of course, also change the element colors to match your brand colors. After adding some more styles for spacing, `:hover`, and `:focus`, the appearance of the `<select>` element is now consistent between browsers.

See this in the following demo from Styling a Select Like It's 2019

https://codepen.io/web-dot-dev/pen/2f6101907a292463bccd8d7a72d09554

What about the `<option>` element? The `<option>` element is a so-called replaced element whose representation is outside the scope of CSS. As of this writing, you can't style the `<option>` element.

There is an ongoing proposal to allow web developers to style and extend built-in web UI controls, including `<select>` and `<option>` elements. This would make it easier to style form controls in the future.

Learn more about Open UI.

Checkboxes and Radio buttons

The appearance of `<input type="checkbox">` and `<input type="radio">` varies across browsers.

https://codepen.io/web-dot-dev/pen/74d28931d0c0e9aacc89f62380f365e4

Open the demo on various browsers to see the difference. Let's see how to make sure that checkboxes and radio buttons match your brand and look the same cross-browser.

In the past, developers could not style `<input type="checkbox">` and `<input type="radio">` controls directly. Checkboxes and radio buttons can be styled via their pseudo elements, now. Or the following replacement technique can be used to create custom styles for these elements.

First, hide the default checkbox and radio button visually.

```
input[type="radio"],
input[type="checkbox"] {
   position: absolute;
   opacity: 0;
}
```

It's important to use `position: absolute` in combination with `opacity: 0` instead of `display: none` or `visibility: hidden` so that the controls are only visually hidden. This will ensure they are still exposed by the browser's accessibility tree. Note that further styling may be needed to ensure that the visually hidden form controls remain positioned "on top" of their replacement elements. Doing so will help ensure that hovering over one of these elements, when a screen reader is on, or when using touch devices with screen readers enabled, the visually hidden controls will be discoverable if exploring by touch, and the screen reader's visible focus indicator will generally appear in the location the controls are rendered on screen.

To show your custom checkboxes and radio buttons, you have different options. You use the `::before` CSS pseudo-element and the CSS `background` property, or use inline SVG images.

```
input[type="radio"] + label::before {
  content: "";
  width: 1em;
  height: 1em;
  border: 1px solid black;
  display: inline-block;
  border-radius: 50%;
  margin-inline-end: 0.5em;
}

input[type="radio"]:checked + label::before {
  background: black;
}
```

Styled form controls must be easy to understand and use. People are used to a checkbox that looks like a checkbox, so make sure when you style a form control that users still understand how to use it.

There are a lot of possibilities with CSS to ensure checkboxes and radio buttons match your brand styles.

https://codepen.io/web-dot-dev/pen/433b6d50b9d7d706e31ee5a638790c3d

Learn more about styling `<input type="checkbox">`, and `<input type="radio">` and custom checkbox styles.

How to use your site's colors for form controls

Do you want to bring your site styles to form controls with one line of code? You can use the `accent-color` CSS property to achieve this.

```
checkbox {
  accent-color: orange;
}
```

As of this writing, only Chrome, Firefox and Edge support `accent-color`. To ensure cross-browser compatibility, you might want to use workarounds until `accent-color` is supported in all platforms.

Resources

- Accent color
- Styling the `<select>` element
- The Accessibility of Styled Form Controls
- Open UI

□□□ #3
□ □□□ □□□ 2026-10-31 22:50:30 UTC
□ cocobook-sync □□□ 2026-11-04 17:39:20 UTC