Style focus

- Use :focus to always show a focus indicator
- Use :focus-visible to selectively show a focus indicator
- Use :focus-within to style the parent of a focused element
- When to display a focus indicator
- Avoid outline: none

The focus indicator (often signified by a "focus ring") identifies the currently focused element on your page. For users who are unable to use a mouse, this indicator is *extremely important* because it acts as a stand-in for their mouse-pointer.

If the browser's default focus indicator clashes with your design, you can use CSS to restyle it. Just remember to keep your keyboard users in mind!

Use `:focus` to always show a focus indicator

The `:focus` pseudo-class is applied any time an element is focused, regardless of the input device (mouse, keyboard, stylus, etc.) or method used to focus it. For example, the `<div>` below has a `tabindex` which makes it focusable. It also has a custom style for its `:focus` state:

```
div[tabindex="0"]:focus {
  outline: 4px dashed orange;
}
```

Regardless of whether you use a mouse to click on it or a keyboard to tab to it, the `<div>` will *always* look the same.

https://glitch.com/edit/#!/focus-style

Unfortunately browsers can be inconsistent with how they apply focus. Whether or not an element receives focus may depend on the browser and the operating system.

For example, the `<button>` below also has a custom style for its `:focus` state.

```
button:focus {
  outline: 4px dashed orange;
}
```

If you click on the `<button>` with a mouse in Chrome on macOS you should see its custom focus style. However, you will not see the custom focus style if you click on the `<button>` in Safari on macOS. This is because in Safari the element does not receive focus when you click on it.

https://glitch.com/edit/#!/focus-style2

See the MDN reference for `<button>` focus behavior for a summary of which browsers and operating systems will apply focus to `<button>` elements.

Because the behavior of focus is inconsistent, it may require a bit of testing on different devices to ensure your focus styles are acceptable to your users.

Use `:focus-visible` to selectively show a focus indicator

The new `:focus-visible` pseudo-class is applied any time that an element receives focus and the browser determines via heuristics that displaying a focus indicator would be beneficial to the user. In particular, if the most recent user interaction was via the keyboard and the key press did not include a meta, `ALT` / `OPTION`, or `CONTROL` key, then `:focus-visible` will match.

The button in the example below will *selectively* show a focus indicator. If you use a mouse to click on it, the results are different than if you first use a keyboard to tab to it.

```
button:focus-visible {
  outline: 4px dashed orange;
}
```

https://glitch.com/edit/#!/focus-visible-style

Use `:focus-within` to style the parent of a focused element

The `:focus-within` pseudo-class is applied to an element either when the element itself receives focus or when another element inside that element receives focus.

It can be used to highlight a region of the page to draw the user's attention to that area. For example, the form below receives focus both when the form itself is selected and also when any of its radio buttons are selected.

```
form:focus-within {
  background: #ffecb3;
}
```

https://glitch.com/edit/#!/focus-within-style

When to display a focus indicator

A good rule of thumb is to ask yourself, "If you clicked on this control while using a mobile device, would you expect it to display a keyboard?"

If the answer is "yes," then the control should probably *always* show a focus indicator, regardless of the input device used to focus it. A good example is the `<input type="text">` element. The user will need to send input to the element via the keyboard regardless of how the input element originally received focus, so it's helpful to always display a focus indicator.

If the answer is "no," then the control may choose to selectively show a focus indicator. A good example is the `<button>` element. If a user clicks on it with a mouse or touch screen, the action is complete, and a focus indicator may not be necessary. However, if the user is *navigating* with a keyboard, it's useful to show a focus indicator so the user can decide whether or not they want to click the control using the `ENTER` or `SPACE` keys.

Avoid `outline: none`

The way browsers decide when to draw a focus indicator is, frankly, very confusing. Changing the appearance of a `<button>` element with CSS or giving an element a `tabindex` will cause the browser's default focus ring behavior to kick-in.

A very common anti-pattern is to remove the focus indicator using CSS such as:

```
/* Don't do this!!! */
:focus {
  outline: none;
}
```

A better way to work around this issue is to use a combination of `:focus` and the `:focus-visible` polyfill. The first block of code below demonstrates how the polyfill works, and the sample app beneath it provides an example of using the polyfill to change the focus indicator on a button.

```
/*
  This will hide the focus indicator if the element receives focus via the
  mouse, but it will still show up on keyboard focus.
*/
.js-focus-visible :focus:not(.focus-visible) {
  outline: none;
}

/*
  Optionally: Define a strong focus indicator for keyboard focus.
  If you choose to skip this step, then the browser's default focus
  indicator will be displayed instead.
*/
.js-focus-visible .focus-visible {
  …
}
```

https://glitch.com/edit/#!/focus-visible

□□□ #3
□ □□□ □□□ 2026-10-31 21:33:32 UTC
□ cocobook-sync □□□ 2026-11-04 17:36:23 UTC