**Engineer**

# Button

a control a person presses to make something happen

## What it is

A button is the part of an interface that acts when it is activated: it sends a message, saves a draft, opens a dialog, mutes a call. On the web it is usually the HTML `<button>` element, and in accessibility terms it is anything that carries the `button` role. “Push button” is an older name for the same thing.

WAI-ARIA 1.2 defines the role as “an input that allows for user-triggered actions when clicked or pressed” and adds that buttons are mostly used for discrete actions. A plain command button keeps no state. A toggle button, marked with `aria-pressed`, stays on or off until it is pressed again, and a menu button, marked with `aria-haspopup`, opens a list of further choices.

The line that matters most runs between a button and a link. A link takes the reader somewhere; a button changes something where the reader already is. The ARIA Authoring Practices Guide calls the two functions distinctly different and, when a link is drawn as a button or the reverse, prefers changing the visual design to match the function over borrowing the other element’s role.

The HTML Living Standard describes the `<button>` element as a button labelled by its contents. Its `type` attribute has three keywords. `submit` sends the form the button belongs to, `reset` returns the form’s fields to their starting values, and `button` does nothing by itself, leaving the action to script. When `type` is missing or misspelt, the button is in the Auto state, and a button in that state inside a form, with no command attached, is a submit button. No interactive content may sit inside it: no link, no field and no second button.

The standard also lets a button act on another element without script: `commandfor` names the target by its ID, and `command` says what to do with it, such as showing a popover or opening a dialog as a modal. How much of that a given

browser ships is a question for that browser's own release notes.

### Use the concept in a decision

Put "What is a button?" to work in a real decision. We can help relate the concept to the system you are building or buying. Interfaces that load quickly, work on a phone and behave the same in every browser your customers happen to use.

**Explore our services** Get in touch →

### Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Directed full-stack GIS map development — full-stack mapping, from database to browser

**Directed full-stack GIS map development** Get in touch →

## Why it matters

A button is where reading turns into doing: buying, sending, saving and deleting all pass through one. A button that some people cannot find, reach or understand shuts them out of the part of the product that pays for it.

**The real element arrives with its behaviour attached.** A `<button>` takes keyboard focus, runs its action on Space and on Enter, tells assistive technology that it is a button and what it is called, and stops responding when it carries `disabled`. The Understanding document for WCAG 2.2 success criterion 4.1.2, Name, Role, Value, notes that standard HTML controls already meet the criterion when they are used according to the specification. A `<div>` with a click handler has none of this, and every piece left out shuts somebody out.

**The name is the promise.** A screen reader announces a button by its accessible name and its role, so the name has to say what the button does, not what it looks like. The APG adds a rule for toggles that is easy to break: the label stays the same while the state changes. A button named "Mute" is announced as "Mute toggle button pressed" when it is on. If a design prefers a label that switches between "Mute" and "Unmute", `aria-pressed` is not needed, because the label itself reports the state.

**Size is a requirement, not a taste.** WCAG 2.2 success criterion 2.5.8, Target Size (Minimum), at level AA, asks that a pointer target be "at least 24 by 24 CSS pixels". The exceptions cover small targets with enough space around them, a full-size control elsewhere on the page that does the same job, links inside a sentence, controls whose size the browser sets, and sizes that are essential. A tremor, a moving bus and a thumb on a phone all make small targets harder.

**Focus has to land somewhere sensible.** After a press, the APG expects focus to move inside a dialog that the button opened and, when that dialog closes, typically to return to the button that opened it. When the press leaves the reader in the same context, as an Apply button does, focus typically stays on the button. A keyboard user whose focus jumps to the top of the page after every save starts the page over each time.

## What it looks like in practice

The clearest case starts with one announcement. In a SwiftUI application built from system symbols, the screen reader read the send button aloud as “arrow up circle, button”, because an unlabelled symbol is announced by its internal name. The work of naming all seventeen icon-only controls for screen readers gave each one a label naming its action rather than its shape, and where a control’s meaning depends on its state, the label follows the state. The more lasting part is the check: a linter that finds the construction creating an icon-only control and requires an accessibility label within eight lines of it. Eight lines was chosen by measurement and is deliberately crude: the rule proves that a label exists nearby, not that it is right, and someone still has to listen.

A second lesson is about what a button cannot do. In the payments and entitlements layer built on Stripe and on Apple and Google in-app purchase, a separate access schema of 11 tables decides whether an account may use the company directory, search and data export. The server asks that question again for each request it serves, because a hidden button is a courtesy and not a control.

Buttons matter to readers that are not people, too. The site’s page on being ready for AI agents notes that many agents that work a browser read a page through its accessibility tree, as a screen reader does. Real buttons and labelled forms are part of what lets an agent find a form and fill it in.

This site’s own templates contain no `<button>` element at all. The filled “Get in touch” call is an `<a>` styled as a button, because it leads to the contact page, and its stylesheet rule sets a `min-height` of 24 pixels with a comment citing success criterion 2.5.8. The main menu is a `<details>` element whose `<summary>` opens and closes it, and the template’s comment says that the browser owns the open state and the button semantics, with no script.

## Where it goes wrong

- **A clickable box stands in for the element.** A `<div>` or `<span>` with a click handler answers a mouse and little else: it takes no focus, ignores Space and Enter, and is announced as plain text. Adding `role="button"` corrects the announcement only; focus and keys still have to be written by hand.
- **A link does a button's job, or the reverse.** A link that deletes a record can be opened in a new tab or bookmarked, which repeats the deletion or loses track of it. A button that only navigates cannot be opened in a new tab and has no address to copy.
- **A bare button inside a form submits it.** Because a missing `type` puts a button in the Auto state, an "Add another row" or "Show password" button placed inside a form sends the form the moment it is pressed. Writing `type="button"` on every button that is not meant to submit removes the surprise.
- **The icon is the whole label.** A bin, a pencil or an arrow means something to the person who drew it. With no accessible name, the screen reader reads a file name, a symbol name or just "button". A visible word is best; where there is no room for one, an accessible name that states the action fills the gap.
- **A hidden or disabled button is treated as security.** Taking the button off the page stops honest users only; the request behind it can still be sent, and the server has to decide on its own whether this account may do this.

## Further reading

- WAI-ARIA 1.2: button role — the role's definition, its toggle and menu variants, and the states and properties it supports.
- HTML Living Standard: the button element — the content model, the submit, reset and button types, the Auto state, and the `command` and `commandfor` attributes.
- ARIA Authoring Practices Guide: Button Pattern — buttons versus links, keyboard behaviour, where focus goes after a press and the rule for toggle labels.
- Understanding SC 2.5.8: Target Size (Minimum) — the 24 by 24 pixel minimum, the spacing test and the other exceptions.
- Understanding SC 4.1.2: Name, Role, Value — why every control needs a name and a role that software can read, and when standard HTML controls already meet the criterion.



Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

**See what it costs** Get in touch →

# Make the terminology practical

Ask how “What is a button?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

**Explore our services**

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/button/