HOW SOFTWARE WORKS · BEGINNER · 14 MIN READ

Forms that work: labels, checks and safe sending

Sign-up, contact and waitlist forms are the first thing most people build. Make them easy to fill in, check them in the browser and on the server, send them once, and write errors people can fix.

Pass the check (4 of 5) online to earn the “Form Fixer” badge:
https://digitalbuilders757.com/university/study/forms-that-work

Build it from real form parts

- Wrap the fields in a <form>, and give every field a <label> linked with for and id. Clicking the label focuses the field, and screen readers read it out.
- A placeholder isn't a label: it disappears as soon as someone types, and it's often too faint to read.
- Pick the right type: email, tel, number, date or password. Phones then show the right keyboard, and the browser can check the format.
- Give every field a name. It's the key the value is sent under, so a field without one is quietly left out.
- Add autocomplete so browsers and password managers can fill things in: autocomplete="email", "name", "new-password" on sign-up and "current-password" on sign-in.
- End with a real <button type="submit">, so pressing Enter sends the form too.

Check it twice

Browser checks such as required, minlength, maxlength and type="email" give people instant feedback before anything is sent. They're for convenience, not safety: anyone can switch them off in DevTools or send a request straight to your server.

So the server checks every request again: the right fields, the right types, sensible lengths, and whether this person is allowed to do this. Libraries such as Zod let you describe the shape once and check it in both places.

Send it once, and say what happened

- When JavaScript handles the submit, call event.preventDefault() first, or the browser reloads the page before your code finishes.

- Disable the button and show 'Sending…' while you wait, so an impatient double click doesn't send it twice.
- Afterwards, say what happened: a clear success message, or an error that says what to do next.
- If it fails, keep everything the person typed. Nobody wants to fill a form in twice.
- In Next.js, a Server Action can take the form directly on the server; the same rules apply.

Errors people can fix

- Say what's wrong and how to fix it, next to the field: 'Enter an email like name@example.com', not 'Invalid input'.
- Link the message to its field with aria-describedby, and set aria-invalid="true" on the field, so screen readers announce it.
- Don't rely on red alone. Add words, and an icon if you like.
- Move keyboard focus to the first field with a problem, or to a short summary at the top of a long form.

Keep it safe and kind

- Ask only for what you need. Every extra field is more to protect, and more reasons to give up.
- Against spam bots, start with a hidden 'honeypot' field that people never fill in, plus a rate limit on the server, before reaching for puzzles.
- Never email someone their password, and never store one yourself: use a sign-in service.
- For newsletters, record that the person agreed, and confirm the address with an email they click.

Form bugs to spot in AI-written code

- Placeholders used instead of labels.
- Inputs without a name, so the server receives nothing for them.
- Checks only in the browser, with nothing on the server.
- No preventDefault, so the page reloads and the message never shows.
- A button that stays clickable while sending, so orders and sign-ups arrive twice.

- Errors that only appear in the console, or that clear the whole form.

Key terms

Label (forms)
The visible name of a form field, linked to it with for and id. Clicking it focuses the field; screen readers read it.

autocomplete (HTML)
Tells browsers and password managers what a field is, like email or new-password, so they can fill it in.

Event listener
A function that runs when something happens on the page, like a click or a form submit.

Rate limit
The most requests or tokens a key may use in a period, such as per minute.

Form validation
Checking what someone typed before using it: in the browser for quick feedback, and always again on the server.

Honeypot field
A form field hidden from people. Spam bots fill it in, so anything that arrives with it filled is ignored.

Accessibility (a11y)
Making apps usable by everyone, including people using screen readers or keyboards.

Try it: Fix a real form

1. Open the 'Error messages people can fix' starter in the code playground. Submit it empty, then with just 'sam'. What does the error say?
2. Make the error say how to fix it, add autocomplete="email" to the input, and try again.
3. Open a form you've built with DevTools' Elements panel. Does every field have a label and a name?
4. Delete required from a field in DevTools, submit it empty, and see whether your server still says no.
5. Ask your AI helper to review the form for labels, names, server checks and double submits, then check each thing it finds.

KNOWLEDGE CHECK

Forms that work: labels, checks and safe sending

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. Your form uses required and type="email". Does the server still need to check the email?

A. No, because the browser won't send the form until the email is valid
B. Only for sign-up forms, because contact forms are never attacked
C. Yes: browser checks are easy to skip, so the server checks every request
D. No, as long as the form also has a placeholder that shows people the right format

2. Every field has a placeholder but no label. What's the problem?

A. Placeholders make the form send twice
B. Browsers won't submit a form when any of its fields uses a placeholder
C. Placeholders are only allowed on password fields, so the others are invalid
D. Placeholders vanish when typing starts and aren't reliable labels

3. The server gets the name but never the email, though people fill both in. The likely bug?

A. The email input has no name attribute, so it isn't sent
B. The email field is too short to hold most addresses
C. Email addresses can't be sent in forms without encrypting them first
D. The server only reads the first field of every form

4. People keep signing up twice by double-clicking Join. The best fix in the form?

A. Tell people in the page's footer to click only once
B. Disable the button and show 'Sending…' until the reply comes back
C. Make the Join button smaller so it's harder to click twice
D. Remove the button and send the form when they leave the last field

5. Which error message helps someone fix their email address?

A. Invalid input
B. Error 400
C. Enter an email like name@example.com
D. Please check the information you entered and try again

Official sites

MDN: Client-side form validation: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Form_validation
W3C WAI: Forms tutorial, user notification: https://www.w3.org/WAI/tutorials/forms/notifications/
web.dev: Learn Forms, autofill: https://web.dev/learn/forms/autofill

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

Forms that work: labels, checks and safe sending

1. C: Yes: browser checks are easy to skip, so the server checks every request
Anyone can switch browser checks off or send a request straight to your server. Browser checks are for convenience; server checks are for safety.

2. D: Placeholders vanish when typing starts and aren't reliable labels
Once someone types, the hint is gone, and placeholders are often too faint. A <label> stays visible, focuses the field and is read by screen readers.

3. A: The email input has no name attribute, so it isn't sent
A form sends each field under its name. Without one, the value is left out of the request.

4. B: Disable the button and show 'Sending…' until the reply comes back
A disabled button can't be clicked again, and 'Sending…' shows something is happening. The server can also ignore duplicates.

5. C: Enter an email like name@example.com
Good errors say what's wrong and how to fix it, right next to the field.