HOW SOFTWARE WORKS · BEGINNER · 14 MIN READ

CSS layout: flexbox, grid and fitting every screen

Put things in a row with flexbox, in rows and columns with grid, and make pages that fit a phone as well as a laptop. Plus the layout bugs AI-written CSS makes most.

Pass the check (4 of 5) online to earn the “Layout Pro” badge:
https://digitalbuilders757.com/university/study/css-layout-flexbox-grid-responsive

Start with the normal flow

Without any layout CSS, block elements such as headings, paragraphs and divs stack from top to bottom, each as wide as its container, while text and links flow along a line. Most layout trouble comes from fighting that flow with fixed sizes and positions instead of using the tools built for the job.

Rule of thumb: flexbox for things in a line (a menu, a row of buttons, a card's icon and text), grid for rows and columns at once (a gallery, a dashboard, a page layout). Both go on the parent, and they arrange its children.

Flexbox: things in a line

- display: flex on the parent puts its children side by side in a row. flex-direction: column stacks them instead.
- justify-content places them along the line (start, center, or space-between to spread them out). align-items lines them up across it (center, start, stretch).
- gap: 12px puts space between the items, so you don't need margins on each one.
- flex-wrap: wrap lets items move onto a new line when they run out of room, instead of squashing.
- flex: 1 on one item makes it take up all the space that's left, such as a search box between a logo and a button.
- Centre anything inside a box: display: flex; justify-content: center; align-items: center on the parent.

Grid: rows and columns

display: grid; grid-template-columns: repeat(3, 1fr); gap: 16px makes three equal columns. fr means a share of the free space, so 2fr 1fr gives the first column twice as much as the second.

The most useful line in CSS for galleries and card lists: grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)). It makes as many columns as fit, each at least 200px wide, so a laptop shows four and a phone shows one, with no media queries at all.

An item can take more room with grid-column: span 2.

Fit every screen

- Pages need the viewport tag, <meta name="viewport" content="width=device-width, initial-scale=1">, or phones show a shrunken desktop page. Next.js and most app builders add it for you.
- Work mobile-first: write the phone styles first, then add @media (min-width: 768px) { ... } for bigger screens. Tailwind's md: and lg: prefixes work the same way.
- Avoid fixed widths. max-width: 700px lets a box shrink on a phone and stop growing on a big screen; widths in %, fr or flex do the same.
- Give images max-width: 100% and height: auto so they never burst out of their box.
- Test at phone size: DevTools' device toolbar shows your page at 375px wide in one click.

Layout bugs to spot in AI-written CSS

- The page scrolls sideways on phones: something is wider than the screen, such as a fixed width, a long URL or a wide table. Find it in DevTools, then use max-width, overflow-wrap: anywhere, or a scrolling wrapper for the table.
- height: 100vh on phones is taller than what you can see, because it ignores the browser's toolbars. 100dvh uses the visible height.
- position: absolute used for whole layouts: it looks fine on one screen and overlaps on every other. Use flexbox or grid.
- z-index that does nothing: it only works on positioned elements and on flex or grid items.
- Text cut off in a box with a fixed height. Let the height grow with min-height instead.

Key terms

CSS
The language for how a page looks: colours, fonts, spacing, layout.

Flexbox
A CSS layout tool for arranging items in a row or column and spacing them out.

CSS grid
A CSS layout tool for rows and columns at once, like galleries, dashboards and whole pages.

Box model
Every element is a box: content, then padding, then border, then margin.

Responsive design
Layouts that adapt to any screen size, from phone to desktop.

Media query
CSS that applies only on some screens, such as @media (min-width: 768px) for tablets and wider.

Viewport
The visible area of the page in the browser window. On a phone it's about 375 to 430 pixels wide.

Mobile-first
Writing the phone layout first, then adding rules for bigger screens with min-width media queries.

Try it: Lay out a real page

1. Open the 'Lay out boxes with flexbox' starter in the code playground: centre the boxes, then stack them.
2. Open 'A responsive card grid'. Switch to repeat(auto-fit, minmax(120px, 1fr)), narrow your browser window and watch the columns wrap.
3. Open 'Stop the sideways scroll' and fix the banner with max-width.
4. Open any site you've built in DevTools' device toolbar at 375px wide. Does anything stick out, overlap or get cut off?
5. Ask your AI helper to explain one layout rule in your project line by line, then change one value and see if it was right.

KNOWLEDGE CHECK

CSS layout: flexbox, grid and fitting every screen

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

1. Three buttons should sit side by side, with space between them. What do you add to their parent?

A. display: flex and gap: 12px
B. display: block on each button, so they line up
C. position: absolute, with a different left value for each button
D. float: right on the parent element

2. What does grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)) give you?

A. Exactly 200 columns that share the space equally
B. As many columns as fit, each at least 200px wide
C. One column 200px wide, then one that fills the rest
D. Columns that only appear on screens wider than 200px

3. What does mobile-first CSS mean?

A. Build the mobile app first and the website once the app is finished
B. Use max-width media queries so desktop styles load before phone ones
C. Phone styles first, then min-width media queries for bigger screens
D. Only test on a phone, because most visitors use one

4. Your page scrolls sideways on phones. What's the most likely cause?

A. The phone's browser can't read CSS written for desktop websites
B. The font is too small for the screen
C. The page has too many images, so the browser widens it to fit them
D. Something is wider than the screen, like a fixed width or a long URL

5. How do you centre a box both ways inside its parent?

A. On the parent: display: flex; justify-content: center; align-items: center
B. Give the box margin: 50% on every side
C. Use text-align: center on the box, which centres it both ways
D. Set position: center on the box itself

Official sites

MDN: Flexbox: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Flexbox
web.dev: Learn CSS, Grid: https://web.dev/learn/css/grid
web.dev: The large, small and dynamic viewport units: https://web.dev/blog/viewport-units

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

ANSWER KEY

CSS layout: flexbox, grid and fitting every screen

1. A: display: flex and gap: 12px
Flexbox lines the parent's children up in a row, and gap spaces them evenly without margins on each one.

2. B: As many columns as fit, each at least 200px wide
auto-fit makes as many columns as fit and minmax keeps each one at least 200px, so the grid reflows from many columns to one by itself.

3. C: Phone styles first, then min-width media queries for bigger screens
Start with the simplest layout, for the smallest screen, then add rules as screens get wider. Tailwind's md: and lg: prefixes work this way.

4. D: Something is wider than the screen, like a fixed width or a long URL
One element wider than the screen widens the whole page. Find it in DevTools and swap fixed widths for max-width.

5. A: On the parent: display: flex; justify-content: center; align-items: center
Flexbox centres along the line with justify-content and across it with align-items. text-align only centres text and inline content.