

Frontend · concept board

CSS

Box model, specificity, Flexbox, Grid, responsive design

1. Cascade and specificity

2. Box model

3. Positioning and stacking

4. Flexbox (1-D layout)

5. Grid (2-D layout)

6. Responsive design and units

7. Variables, animation and performance

Draw on any slide, then Present (arrow keys) or export the board as a PDF.

1. Cascade and specificity

Definition

CSS styles HTML. When several rules target the same element, the browser picks one by origin & !important
→ specificity → source order.

Selector	Specificity (a, b, c)
p	(0, 0, 1)
.card p	(0, 1, 1)
#main .card p	(1, 1, 1)
style="..." inline	beats all selectors
!important	beats everything above — avoid

- a = IDs, b = classes / attributes / pseudo-classes, c = elements / pseudo-elements. Compare left to right.
- Some properties inherit (color, font); most don't (margin, border).
- :where() has zero specificity; :is() takes its most specific argument.

Common mistake

Fighting specificity with !important or IDs. Keep selectors flat (one class), e.g. BEM:
.card__title--active.

2. Box model



From the inside out

Always use border-box

```
*, *::before, *::after {  
  box-sizing: border-box; /* width includes padding + border */  
}  
.card {  
  width: 300px;  
  padding: 16px;  
  border: 1px solid #ddd; /* still exactly 300px wide */  
}
```

- content-box (default): width = content only, so padding and border add to it.
- Vertical margins between blocks collapse (the larger one wins); flex and grid items don't collapse.
- display: none removes the element; visibility: hidden keeps its space; opacity: 0 keeps space and stays clickable.

3. Positioning and stacking

position	Relative to	In normal flow?
static	—	yes (default)
relative	its own normal position	yes (space kept)
absolute	nearest positioned ancestor	no
fixed	the viewport	no
sticky	scroll container; sticks at a threshold	yes until it sticks

Tip

z-index only works on positioned (or flex/grid) elements, and only competes inside the same stacking context (created by position + z-index, opacity < 1, transform, etc.).

4. Flexbox (1-D layout)

Navbar with Flexbox

```
<style>
  nav { display: flex; align-items: center; gap: 12px; padding: 12px; background: #0f172a; }
  nav a { color: white; text-decoration: none; }
  .spacer { margin-left: auto; } /* push the rest to the right */
  .grow { flex: 1; }           /* flex-grow: 1 - take the free space */
</style>
<nav>
  <a href="#">Logo</a>
  <a href="#">Docs</a>
  <a class="spacer" href="#">Login</a>
  <a href="#">Sign up</a>
</nav>
```

Property	Axis / meaning
flex-direction	row (default) or column — sets the main axis
justify-content	alignment along the main axis
align-items	alignment along the cross axis
flex-wrap	allow items to wrap to new lines
flex: 1	shorthand for grow 1, shrink 1, basis 0%
gap	space between items

5. Grid (2-D layout)

Responsive card grid — no media queries

```
<style>
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(160px, 1fr));
  gap: 12px;
}
.grid div { background: #e0f2fe; padding: 24px; border-radius: 8px; text-align: center; }
</style>
<div class="grid">
  <div>1</div><div>2</div><div>3</div><div>4</div><div>5</div><div>6</div>
</div>
```

- Use Flexbox for a row or column of items; use Grid for rows *and* columns (page layouts, galleries).
- fr = a fraction of the free space; minmax(min, max) sets size limits.
- grid-template-areas names regions ("header header" / "sidebar main").

6. Responsive design and units

- Mobile-first: base styles for small screens, then @media (min-width: 768px) { ... } for bigger ones.
- rem = relative to the root font size (good for spacing and type); em = relative to the parent font size.
- % relative to the parent; vw/vh relative to the viewport; dvh handles mobile browser bars.
- clamp(1rem, 2.5vw, 2rem) = fluid size with a min and max.
- Container queries (@container) respond to the parent's size, not the screen.

Common mistake

Fixed pixel widths on containers cause horizontal scrolling on phones. Use max-width + percentages, and test at 360px wide.

7. Variables, animation and performance

Custom properties for theming

```
:root { --brand: #0891b2; --radius: 12px; }  
[data-theme="dark"] { --brand: #22d3ee; }  
.button {  
  background: var(--brand);  
  border-radius: var(--radius);  
  transition: transform 150ms ease;  
}  
.button:hover { transform: translateY(-2px); } /* animate transform, not top */
```

- Animate transform and opacity — they skip layout and paint (GPU-friendly).
- Animating width, top or margin triggers reflow on every frame.
- Respect @media (prefers-reduced-motion: reduce).
- Pseudo-classes target state (:hover, :focus-visible, :nth-child); pseudo-elements add parts (::before, ::after).

Revise

Specificity (IDs > classes > elements), border-box, Flexbox for 1-D and Grid for 2-D, mobile-first with rem and clamp, animate transform/opacity.

Rapid-fire Q&A (1/2)

How is CSS specificity calculated?

Count IDs, then classes/attributes/pseudo-classes, then elements/pseudo-elements, and compare left to right. Inline styles beat selectors; !important beats everything.

content-box vs border-box?

content-box: width is content only, so padding and border add to it.
border-box: width includes padding and border.

Flexbox vs Grid?

Flexbox lays out in one dimension (a row or a column). Grid lays out in two dimensions (rows and columns).

How do you center a div?

Parent: display: flex;
justify-content: center; align-items: center — or display: grid;
place-items: center.

relative vs absolute vs fixed vs sticky?

relative offsets from its own spot;
absolute positions against the nearest positioned ancestor; fixed against the viewport; sticky scrolls until it hits a threshold, then sticks.

rem vs em?

rem is relative to the root font size; em is relative to the parent's font size, so it compounds when nested.

Rapid-fire Q&A (2/2)

What creates a stacking context?

position with z-index, opacity below 1, transform, filter, isolation: isolate and more. z-index only competes inside the same context.

display: none vs visibility: hidden?

display: none removes the element from layout; visibility: hidden hides it but keeps its space.

Which properties are cheap to animate?

transform and opacity — they avoid layout and paint, so the GPU compositor can handle them.

What is margin collapse?

Adjacent vertical margins of block elements merge into the larger one. It doesn't happen in flex or grid containers.