CSS Layout — Box Model, Flexbox & Grid

Web Technologies — Lecture 5

Masoud Hamad

State University of Zanzibar (SUZA)

Semester II, 2025/2026

Outline

Block, Inline, Inline-Block

Every HTML element has a default `display` value.

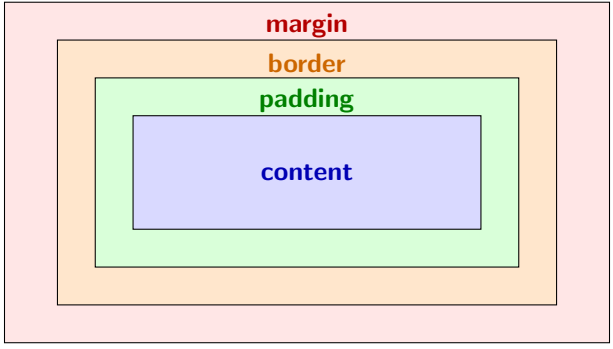
- **block** — takes a full line; respects `width`, `height`, all margins
  defaults: `<div>`, `<p>`, `<h1>`, `<section>`, `<header>`
- **inline** — flows in text; ignores `width`/`height`; only horizontal padding/margin
  defaults: `<span>`, `<a>`, `<strong>`, `<em>`
- **inline-block** — flows like inline, sizes like block
- **none** — removes from layout entirely (different from `visibility:hidden`)
- **flex**, **grid** — container layout systems (next sections)

Override the Default

```
/* make a span behave like a block */
span.button { display: inline-block; padding: 0.5rem 1rem; }

/* hide responsively */
@media (max-width: 600px) {
  .desktop-only { display: none; }
}
```

Every Element is a Box



box-sizing

By default, `width` sets only the content area — padding and border are added on top. This causes off-by-N pixel surprises.

```
/* recommended: width includes padding and border */
* { box-sizing: border-box; }
```

```
.card {
  width: 300px;
  padding: 20px;        /* still 300px wide, content shrinks to 260px */
  border: 1px solid;
}
```

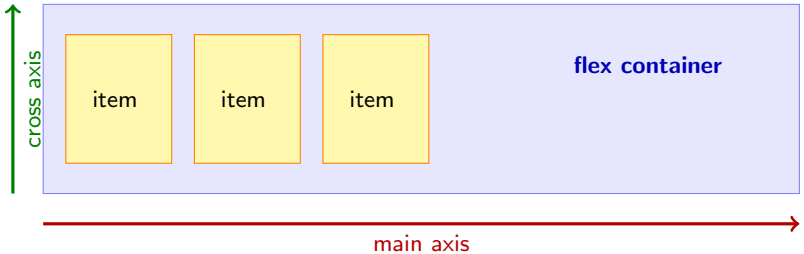
Margins Collapse

Vertical margins of adjacent block elements **merge** into the larger of the two.

```
h2 { margin-bottom: 24px; }
p  { margin-top:    16px; }
/* gap between an h2 and the next p = 24px (NOT 40px) */
```

Margin collapse does *not* happen with horizontal margins, inside flex/grid containers, or across non-empty padding/border.

Flexbox Vocabulary



- **Flex container** — the parent (`display: flex`)
- **Flex items** — the direct children
- **Main axis** — the direction items flow (default: row)
- **Cross axis** — perpendicular to main

Container Properties

```
.container {
  display: flex;
  flex-direction: row;          /* row | row-reverse | column | column-reverse */
  flex-wrap: wrap;              /* nowrap (default) | wrap | wrap-reverse */
  gap: 1rem;                    /* gap between items */

  /* alignment along main axis */
  justify-content: space-between;
  /* flex-start | center | flex-end |
     space-between | space-around | space-evenly */

  /* alignment along cross axis */
  align-items: center;
  /* flex-start | center | flex-end | stretch | baseline */
}
```

Item Properties

```
.item {
  flex: 1;                 /* shorthand: grow=1 shrink=1 basis=0 */
  /* equivalent: flex: 1 1 0; */

  flex-grow:   1;          /* how to share extra space */
  flex-shrink: 1;          /* how to give up space when too tight */
  flex-basis:  200px;      /* preferred size before grow/shrink */

  align-self: flex-end;    /* override align-items for this item */
  order:      2;           /* visual reorder without changing HTML */
}
```

Common Flex Patterns

```
/* Centre anything */
.parent { display: flex; justify-content: center; align-items: center; }

/* Header: logo left, nav right */
.header { display: flex; justify-content: space-between; align-items: center; }

/* Equal-width buttons */
.toolbar { display: flex; gap: 0.5rem; }
.toolbar button { flex: 1; }

/* Sidebar 250px + content fills the rest */
.layout { display: flex; gap: 1rem; }
.sidebar { flex: 0 0 250px; }
.content { flex: 1; }
```

Grid Basics

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;      /* three equal columns */
  gap: 1rem;
}

/* Or use repeat() */
.grid { grid-template-columns: repeat(3, 1fr); }

/* Mixed sizing */
.layout {
  grid-template-columns: 250px 1fr;        /* fixed sidebar + flex content */
}
```

`fr` = “fractional unit” — shares the leftover space.

Responsive Grids Without Media Queries

```
.cards {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 1rem;
}
/* As the viewport shrinks, columns drop automatically. */
```

`auto-fit`: collapse empty tracks. `auto-fill`: keep them as empty space.

Grid Item Placement

```
.feature {
  grid-column: 1 / 3;         /* span columns 1 to 3 */
  grid-row:    1 / 2;
}

/* span shorthand */
.wide { grid-column: span 2; }

/* Named template areas (very readable) */
.layout {
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-areas:
    "head head"
    "side main"
    "foot foot";
}
.layout > header { grid-area: head; }
.layout > nav    { grid-area: side; }
.layout > main   { grid-area: main; }
.layout > footer { grid-area: foot; }
```

When to Use Which

Flexbox	**Grid**
1-D (row OR column)	2-D (rows AND columns)
Content-driven sizing	Layout-driven sizing
Navigation bars, toolbars	Page layouts, dashboards
Centring, distributing	Card grids, photo galleries
Components	Whole pages

They compose: a Grid item can have `display: flex`, and vice versa.

Mobile-First Media Queries

```css
/* base styles -- mobile */
.nav { display: block; }

/* tablet and up */
@media (min-width: 768px) {
  .nav { display: flex; gap: 2rem; }
}

/* desktop and up */
@media (min-width: 1024px) {
  .container { max-width: 1100px; margin: 0 auto; }
}
```

Why mobile-first? Smaller styles load first on the slowest devices; larger screens layer on top.

Common Breakpoints

- Phones: 320–480 px
- Large phones: 481–767 px
- Tablets: 768–1023 px
- Laptops: 1024–1439 px
- Desktops: 1440 px+

Tip: pick breakpoints based on *your content*, not specific devices. Resize the browser slowly until something breaks — that's a breakpoint.

Custom Properties

```
:root {
  --primary: #4f46e5;
  --bg:      #f8fafc;
  --space:   1rem;
  --radius:  8px;
}

.button {
  background: var(--primary);
  border-radius: var(--radius);
  padding: var(--space);
  transition: transform 0.15s ease;
}

@media (prefers-color-scheme: dark) {
  :root { --bg: #0f172a; }
}
```

Modern Sizing Functions

```
/* clamp(min, preferred, max) -- fluid type */
h1 { font-size: clamp(1.5rem, 4vw, 3rem); }

/* min() and max() -- pick the smaller / larger */
.container { width: min(90%, 1100px); }

/* aspect-ratio -- no more padding hacks */
.video { aspect-ratio: 16 / 9; width: 100%; }

/* logical properties (RTL-safe) */
.card { padding-inline: 1rem; padding-block: 0.5rem; }
```

Layout Debugging

- Toggle a temporary `outline: 1px solid red` to see boxes
- Browser devtools → Inspector: visualises box, flex, grid
- Firefox has the best Grid + Flex inspectors
- Watch for **overflow**: `overflow: auto` reveals hidden content

Summary

- `display` sets layout mode: block, inline, flex, grid
- Box model: content + padding + border + margin (use `border-box`)
- Flexbox = 1-D layouts (rows or columns); Grid = 2-D layouts
- Mobile-first media queries layer larger-screen rules on top
- CSS variables, `clamp()`, `aspect-ratio` are the modern toolkit

Next: JavaScript and the DOM.

Questions?