LECTURE 6

The CSS Box Model and Display

Every element is a rectangular box — understanding its size and behavior is the core of CSS layout.

CSC336 Web Technologies

OVERVIEW

In This Lecture

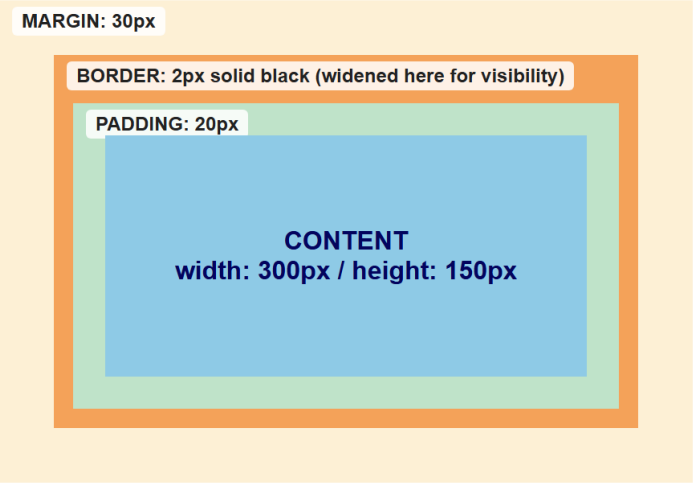
1. The four layers of the box model: content, padding, border, and margin
2. Margin collapsing, and why it surprises beginners
3. box-sizing: content-box vs. border-box
4. Controlling size with width, height, and min-/max- constraints
5. The display property: block, inline, inline-block, and none
6. A real technique: turning a <ul> into a horizontal menu with display

Four Nested Layers

1. Content — the actual text, image, or other content
2. Padding — transparent space inside the border, colored by the background
3. Border — a line (or nothing) drawn around the padding
4. Margin — transparent space outside the border, never filled with color

The Four Layers, Rendered

```
.content { width: 300px; height: 150px; background: steelblue; }
.padding { padding: 20px; background: seagreen; }
.border  { border: 20px solid darkorange; }
.margin  { margin: 20px; background: wheat; }
```

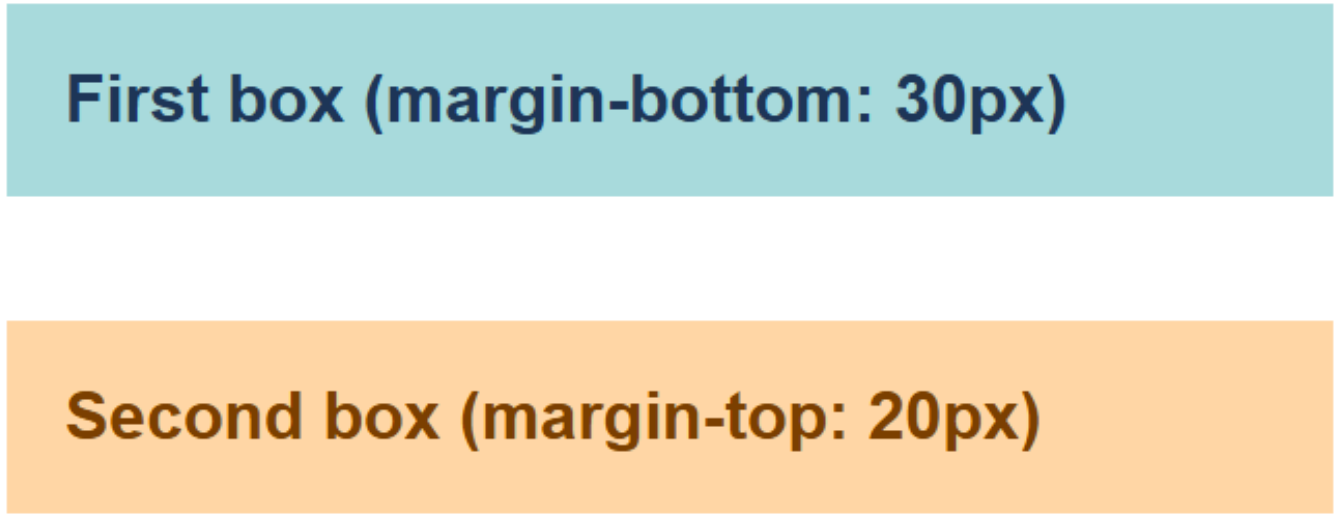


Two Adjacent Vertical Margins

```
.box-one { margin-bottom: 30px; }
.box-two { margin-top: 20px; }
```

You might expect a 50px gap — but margin collapsing means the actual gap is only 30px, the LARGER of the two.

Margin Collapsing, Rendered



The visible gap between the two boxes above is 30px — the larger of the two margins — not 50px (30px + 20px), because of margin collapsing.

Only Happens Vertically

WARNING

Margin collapsing applies only to top/bottom margins of block elements in normal flow. It never happens with left/right margins, or with flex, grid, floats, or absolute positioning.

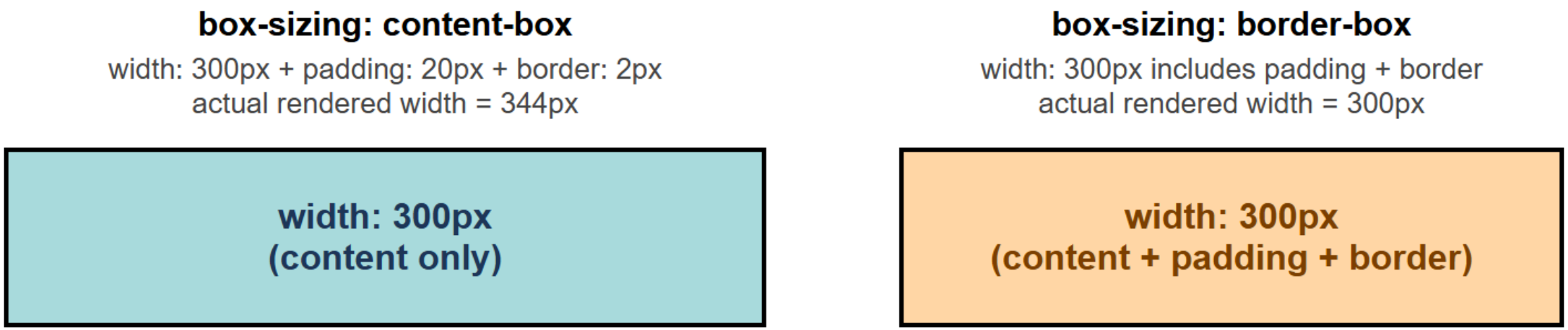
content-box (Default) vs. border-box

```
/* content-box: padding + border ADD to width */
.a { box-sizing: content-box; width: 300px; padding: 20px; border: 2px solid; }
/* actual rendered width = 344px */

/* border-box: padding + border are INCLUDED in width */
.b { box-sizing: border-box; width: 300px; padding: 20px; border: 2px solid; }
/* actual rendered width = exactly 300px */
```

Same Width, Different Sizing

Same width, padding, and border on both — content-box renders visibly wider than border-box.



border-box: The Practical Default

TIP

Because border-box makes sizing far more predictable, most real style sheets start with *, *::before, *::after { box-sizing: border-box; } as a site-wide reset.

min-/max- Constraints

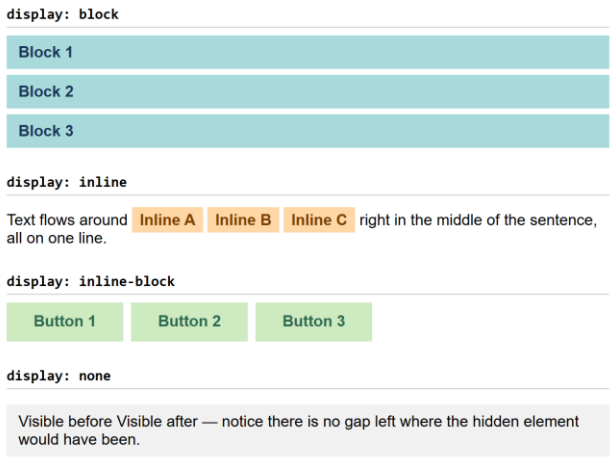
- max-width caps how wide a box can grow — never wider than 1000px, even on a huge screen
- min-width / min-height set a floor the box will never shrink below
- Together they let a box scale down on narrow screens but stop growing past a sensible limit on wide monitors

block, inline, inline-block, none

1. block — starts a new line, fills the full available width (div, p, h1-h6, ul)
2. inline — flows within text, width/height are ignored (span, a, strong)
3. inline-block — flows inline, but respects width/height/vertical margin (nav links, buttons)
4. none — removed from the page entirely, taking up no space at all

All Four Values, Compared

```
.section { display: block; }        /* new line, full width  */
.tag     { display: inline; }       /* flows in text, no w/h */
.button  { display: inline-block;   /* inline + respects w/h */
           width: 120px; height: 40px; }
.hidden-panel { display: none; }    /* removed entirely      */
```



A COMMON MIX-UP

display: none vs. visibility: hidden

```
.gone    { display: none; }      /* removed entirely — no space reserved   */
.invisible { visibility: hidden; } /* invisible, but still takes up its space */
```

Toggling display: none with JavaScript reflows the page (neighbors shift to fill the gap); visibility: hidden leaves a blank gap where the element used to be.

A Row of Equal-Sized Buttons

```
.btn {
  display: inline-block;
  width: 120px;
  text-align: center;
  padding: 10px 0;
  margin-right: 8px;
}
```

inline-block is what lets several fixed-width buttons sit on one line, side by side, without giving up width/height control the way plain inline would.

A PRACTICAL EXAMPLE

Turning a <ul> Into a Horizontal Menu

```
.menu { list-style: none; margin: 0; padding: 0; }
.menu li { display: inline-block; margin-right: 20px; }
```

li is block by default — this one change turns a vertical list into a horizontal menu, with no HTML change at all.

Before and After, Rendered

Default: li is block

- Home
- About
- Services
- Contact

```
.menu li { display: inline-block; }
```

Home About Services Contact

Why Build a Menu This Way?

- A nav menu IS, conceptually, a list of links — that's exactly what it is to a screen reader and a search engine
- Screen readers announce it as a list, telling users how many links there are
- The HTML never has to change later — redesign to a dropdown or hamburger menu by changing only CSS

Borders, Radius, and Shadow

```
.card {
  border: 2px solid #333;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}

.avatar { border-radius: 50%; }  /* a circle */
```

Box Shadow: Raised, Soft, and Inset

```
.raised {
  box-shadow: 0 4px 8px rgba(0,0,0,.2);
}

.soft {
  box-shadow: 0 8px 20px rgba(0,0,0,.15);
}

.pressed {
  box-shadow:
    inset 0 2px 4px rgba(0,0,0,.35);
}
```

Layered Shadows and Text Shadow

```
.card {
  box-shadow:
    0 1px 2px rgba(0,0,0,.08),
    0 4px 8px rgba(0,0,0,.08),
    0 12px 24px rgba(0,0,0,.08);
}

h1 {
  text-shadow: 2px 2px 4px rgba(0,0,0,.3);
}
```

Consistent Spacing Techniques

- A spacing scale — pick a small set of values (4px, 8px, 16px, 24px) and only ever use those
- CSS custom properties — define --space-md: 16px once in :root, reuse everywhere with var(--space-md)
- Change the whole site's spacing rhythm by editing one value in :root

KEY TAKEAWAYS

Lecture 6 in Six Points

1. Every box has four layers: content, padding, border, margin — vertically adjacent margins can collapse to the larger value.
2. box-sizing: border-box makes padding and border count inside your width, and is the practical default.
3. min-/max-width/height add flexible constraints on top of a base size.
4. display controls layout behavior: block fills width, inline flows in text, inline-block combines both, none removes the element entirely.
5. A <ul> styled with li { display: inline-block; } is a real way to build a menu without losing list semantics.
6. border-radius, box-shadow, and CSS custom properties keep a design finished and consistent.