ScopeTreasure Courses

CSS COURSES · LESSON 2.5 (SHORT) · CODE SCOPE SERIES

Display: block vs inline vs inline-block

The display property decides how a box sits on the page: block takes the full row, inline flows with text, inline-block does both.

Introduction

Why does a `<div>` always start on a new line while a `<span>` sits quietly inside a sentence? Because every element is born with a **display** type. **Block** elements own their whole row; **inline** elements flow with the text like words; and **inline-block** is the hybrid — it sits in the line *and* accepts a width and height. One property, three personalities — and `display` is the same property that powers Flexbox (2.6) and Grid (2.7), so this little Short is quietly the foundation of the whole layout half of this series. *(Roman Urdu: Display tay karta hai ke box poori line lega, ya jumlay ke andar hi baithe ga.)*

Key points

- **Block** — starts on a **new line** and stretches to the **full width** available. `width` and `height` work. Born-block elements: `div`, `p`, `h1`–`h6`, `section`, `li`.
- **Inline** — stays **inside the line of text**, only as wide as its content. `width` and `height` are **ignored**, and top/bottom margins do nothing. Born-inline elements: `span`, `a`, `strong`, `em`, `img` (roughly).
  (Roman Urdu: Inline element sirf apni jagah leta hai — us ko width dena bekaar hai.)
- **Inline-block** — the hybrid: it **flows in the line** like inline (several can sit side by side), but **accepts width, height, padding and margins** like a block. Buttons and menu items live here.
- **The size table** (the recap that matters): block → width ✓ height ✓ new-line ✓ · inline → width ✗ height ✗ new-line ✗ · inline-block → width ✓ height ✓ new-line ✗.
- **`display: none`** — the fourth value to know: the element vanishes completely, taking **no space at all** (different from hiding with colour tricks — the layout closes up as if it never existed).
- **Centring follow-up from 2.4:** `margin: 0 auto` centres a **block** box. For inline/inline-block children, centre the *parent's text* instead: `text-align: center` on the parent.

Code examples

Example 1 — Block vs inline, side by side

```
.box { background: #ffd9cc; border: 1px solid #e8490f; }
span.tag { background: #d8f3dc; border: 1px solid #0b6e4f; width: 200px; height: 60px; }
```

```
<div class="box">I am a div — block.</div>
<div class="box">Me too — so I start below.</div>
<p>Text with <span class="tag">a span</span> inside the line.</p>
```

What it does: The two divs stack vertically, each stretching across the whole row — block behaviour. The span carries `width: 200px; height: 60px` in its rule… and completely ignores both: inline elements refuse size.

What you will see: two full-width orange boxes stacked, then a sentence with a small green tag exactly the size of its words. The ignored width/height on the span is not a bug — it is the definition of inline.

Example 2 — inline-block buttons in a row

```
.btn {
  display: inline-block;
  width: 140px;
  padding: 10px 0;
  text-align: center;
  background: #0b6e4f; color: white;
  border-radius: 8px;
}
```

```
<a class="btn" href="#">Home</a>
<a class="btn" href="#">Notes</a>
<a class="btn" href="#">Contact</a>
```

What it does: Links are born inline — no width, odd padding behaviour. One line, `display: inline-block`, promotes them: now each takes its 140px width and full padding, yet the three still line up in one row like words in a sentence.

What you will see: three equal green buttons in a neat row. This is how most simple navbars and button groups were built for a decade, and it still works perfectly for small rows.

Example 3 — Hide it, centre it

```
.secret { display: none; }
.wrap { text-align: center; }
.chip {
  display: inline-block;
  padding: 8px 18px;
  background: #0b0f19; color: #faf9f6;
  border-radius: 999px;
}
```

What it does: `.secret` removes its element from the page *and* from the layout. `.wrap` centres its inline-level children with `text-align: center` — the correct centring tool for inline-block, since `margin: 0 auto` only

works on blocks.

What you will see: the secret element nowhere — no gap, no ghost space — and a dark pill-shaped chip sitting in the exact centre of its container.

Quick recap

- **Block**: new line, full width, size works. **Inline**: in the line, content-sized, size ignored. **Inline-block**: in the line, size works.
- `display: none` removes the element and its space.
- Centre blocks with `margin: 0 auto`; centre inline-level children with `text-align: center` on the parent.

Common mistakes

1. **Setting a width on a span and refreshing angrily** — inline ignores width/height by definition. Add `display: inline-block` (or block) first, then size it.
2. **Trying to centre inline elements with margin: auto** — auto margins need a block box with a set width. For inline-level content, `text-align: center` on the parent is the tool.
3. **Using `display: none` to "hide for later" in markup you ship** — it removes the space completely; content meant to appear on hover/tap usually wants a class toggle, not a missing layout. (And screen readers skip `display: none` content entirely.)

Frequently asked questions

What is the difference between inline and inline-block? Inline elements flow inside the line and ignore width and height. Inline-block elements also flow inside the line, but they accept width, height, padding and margins like a block.

Why is my width ignored on a span? A span is inline by definition, and inline elements refuse width and height. Add `display: inline-block` (or `display: block`) first, then the size works.

When should I use inline-block instead of flex? For small rows — menu buttons, chips, badges, button groups — inline-block is the simplest tool. For bigger, responsive layouts, flexbox or grid are the better choice.

Do top and bottom margins work on inline-block? Yes. Inline-block accepts margins on all four sides, unlike inline, where top and bottom margins do nothing.

What is the difference between display: none and visibility: hidden? `display: none` removes the element and its space completely, as if it never existed. `visibility: hidden` hides the element but keeps its space in the layout.

Practice

1. Make three boxes: one block, one inline, one inline-block — give all three `width: 150px; height: 50px` and write down which ones obeyed.

2. Build a row of four menu buttons with `display: inline-block`, then centre the row using the parent's `text-align`. Finally hide the second button with `display: none` and watch the row close up.

Practice quiz

1. How does a block element sit on the page?

- Inside the line of text, only as wide as its content
- On a new line, stretching to the full width available
- Always centred, whatever its width
- It never takes any space at all

Answer: On a new line, stretching to the full width available

2. What happens to width and height set on an inline element like a span?

- They work exactly as written
- They are ignored, the element stays content-sized
- They make it a block element
- They hide the element

Answer: They are ignored, the element stays content-sized

3. What does display: none do to an element?

- It hides it but keeps its space
- It removes it completely, taking no space at all
- It only hides it on phones
- It makes it transparent but clickable

Answer: It removes it completely, taking no space at all

Watch the video lesson: https://youtube.com/shorts/SR2wDmf0Ysk

Read online: https://scopetreasure.com/notes/css-display-block-inline · More free tools and notes at https://scopetreasure.com