ScopeTreasure Courses

CSS COURSES · LESSON 2.9 · CODE SCOPE SERIES

Responsive Design & Media Queries

One page, every screen: the viewport meta tag, media queries, breakpoints and the mobile-first way of thinking.

Introduction

Your visitors arrive on phones, tablets, laptops and TVs — and most of them, honestly, on phones. A page designed only for a wide laptop screen arrives on a phone tiny, sideways-scrolling and unusable. **Responsive design** is the discipline of one page that reshapes itself for every screen: columns stack, menus simplify, text stays readable. Its two tools are tiny — the **viewport meta tag** and the **media query** — and this lesson puts both to work on a real page. *(Roman Urdu: Responsive matlab aik hi page, har screen par theek — phone ho ya laptop.)*

Key points

- **The viewport meta tag — the entry ticket:** `<meta name="viewport" content="width=device-width, initial-scale=1.0">` in the `<head>`. It tells the phone: "the page is as wide as *this device*, shown at scale one-dot-zero". Without it, phones pretend to be 980px-wide desktops and shrink your page to ant size. No responsive page exists without this line.
- **A media query is an if-statement for screens:** `@media (max-width: 768px) { ... }` means "apply these rules **only** when the screen is 768px wide or less". Inside it you *override* the base rules — smaller fonts, one column, a simpler menu.
- **Breakpoints** are the widths where the design changes its shape. Common working points: **≤768px** phones/tablets, **≥1024px** desktops — but the honest rule is: put a breakpoint where *your content starts to break*, not where a device chart says.
  (Roman Urdu: Breakpoint woh jagah hai jahan design apni shakal badalta hai.)
- **Fluid first, breakpoints second:** percentages, `fr`, flex-wrap and `max-width: 100%` on images make a layout bend on its own; media queries then handle the few places where bending is not enough (columns → stack, row → column).
- **Mobile-first thinking:** write the **base CSS for the small phone screen**, then add `@media (min-width: 768px)` blocks that *upgrade* the layout as screens grow. Phones are the majority audience and the weakest devices — design for them first, enhance upward. The opposite (desktop-first with `max-width` patches) works, but fights you more.
- **Test like a visitor:** drag the browser window narrow, and use DevTools' device toolbar (phone sizes, throttles). If it survives 360px wide and 1440px wide, it survives your audience.

Code examples

Example 1 — The viewport tag and a fluid image

```
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    img { max-width: 100%; height: auto; }
    .wrap { max-width: 1000px; margin: 0 auto; padding: 0 16px; }
  </style>
</head>
```

What it does: The meta tag switches the phone into honest mode — layout width = device width. `img { max-width: 100% }` promises no image ever overflows its container (it shrinks instead, `height: auto` keeping its shape). The `.wrap` column is capped at 1000px on big screens yet fluid below that.

What you will see: open the page on a phone (or device toolbar): text at a readable size, image inside the screen — no pinch-zoom, no sideways scroll. Comment out the meta tag and refresh: the "desktop illusion" returns, tiny and zoomed-out. One line made that difference.

Example 2 — Columns that stack: the 768px media query

```
.cards { display: flex; gap: 16px; }
.cards .card { flex: 1; }

@media (max-width: 768px) {
  .cards { flex-direction: column; }
  h1 { font-size: 28px; }
}
```

What it does: On wide screens the three cards share a row (the flex pattern from Lesson 2.6). At 768px and below, the media query flips the container to a **column** — cards stack full-width, one thumb-scroll after another — and the heading steps down a size so it stops wrapping awkwardly.

What you will see: drag the window across the 768px line and watch the row fold into a stack *live*, then unfold again. Nothing in the HTML changed — the same page, two shapes, chosen by width.

Example 3 — The same page, written mobile-first

```
/* base = the phone design */
.menu { display: flex; flex-direction: column; gap: 8px; }
.hero h1 { font-size: 30px; }

/* tablet and up */
@media (min-width: 768px) {
  .menu { flex-direction: row; gap: 20px; }
  .hero h1 { font-size: 44px; }
}

/* desktop and up */
@media (min-width: 1024px) {
  .hero h1 { font-size: 56px; }
  .grid { display: grid; grid-template-columns: repeat(3, 1fr); gap: 16px; }
}
```

What it does: The base rules describe the phone (stacked menu, modest heading). Each `min-width` block *adds* — a row menu at 768, a full 3-column grid and the biggest heading at 1024. This is the exact structure of Example 2 turned inside-out: same result, but the CSS grows from the smallest screen up — the mobile-first order.

What you will see: one page with three honest shapes (phone / tablet / desktop) — and a stylesheet you can read top-to-bottom as "small, then bigger, then biggest". Most professional CSS files on earth are organised exactly like this.

Quick recap

- No **viewport meta** = no responsive page, whatever else you write.
- A **media query** is an if for screens: `@media (max-width: 768px) { … }` overrides rules under that width.
- **Breakpoints** where the content needs them; **mobile-first** = base for phones, `min-width` upgrades upward.
- Fluid tools (%, fr, wrap, `max-width: 100%`) do the bending; media queries do the reshaping.

Common mistakes

1. **Forgetting the viewport meta** — the page "ignores all my media queries" and renders zoomed-out on phones. The queries may be perfect; the phone is still pretending to be a desktop. Check the head first.
2. **Designing only at full-screen width** — the layout is built and admired at 1440px, then collapses at 390px where most visitors live. Keep one narrow window (or the device toolbar) open *while* you design.
3. **Fixed pixel widths on layout boxes** — `width: 1200px` on a container guarantees sideways scrolling on everything smaller. Use `max-width` (a ceiling) + fluid widths instead, so boxes can shrink but never overflow.

Practice

1. Take the card gallery from Lesson 2.7 and make it responsive: 3 columns on desktop, 2 under 900px, 1 under 560px — media queries only, no HTML changes.
2. Rewrite Example 2's page mobile-first (base = stacked, upgrades at 768 and 1024) and verify all three shapes in the device toolbar at 390px, 768px and 1280px.

Practice quiz

1. What does @media (max-width: 600px) do?

- Different styles for small screens
- A new animation
- A font import
- A colour variable

Answer: Different styles for small screens

Watch the video lesson: https://youtu.be/IeK4LrwqJJQ

Read online: https://scopetreasure.com/notes/css-responsive-media-queries · More free tools and notes at https://scopetreasure.com