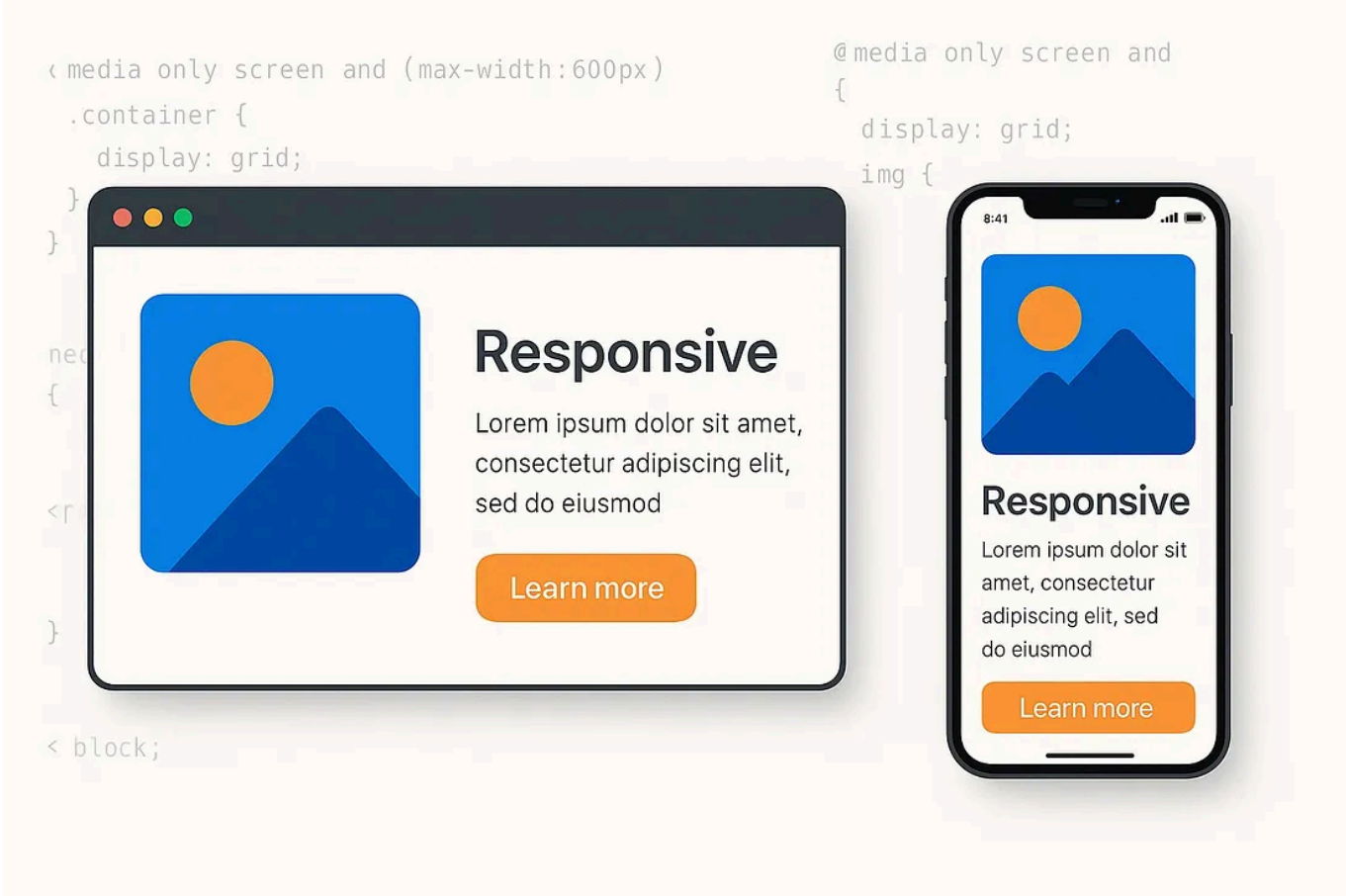
How to Create Responsive Websites Like a Pro

Prajapatiankur Following 4 min read · Jun 3, 2025

12 7

In this post, let's break down how to **actually** build fully responsive websites that don't break the internet (or your codebase). No fluff, just the real dev wisdom.

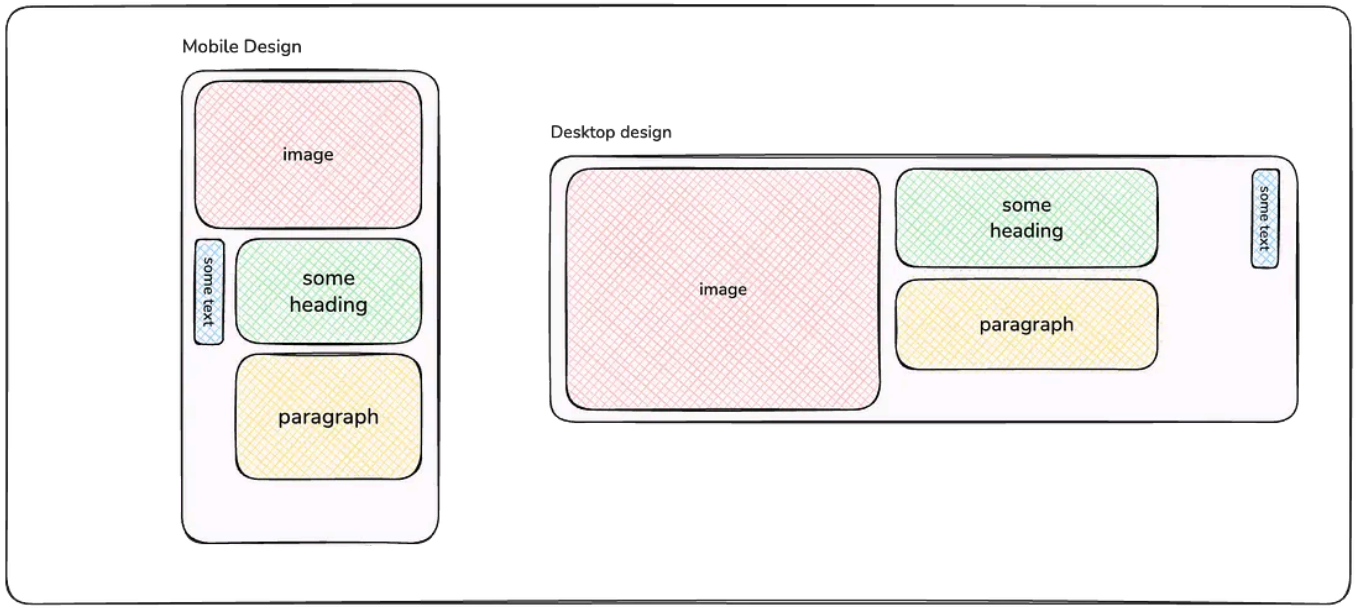


1. Start with Breakpoints in Mind

Before you even touch your CSS, take a step back and study your design across **all screen sizes** — mobile, tablet, desktop, and large desktops.

Use a smart layout with nested containers so that adapting to screen sizes is as simple as changing `flex-direction` or `grid-template-columns`.

> 50% of responsive design is just **solid HTML structure.**




```
<!-- mobile first approach -->
<section class="some-section">
    <div class="top">
        <img
            src="hero.webp"
            alt="hero image"
        >
    </div>
    <div class="bottom">
        <div class="left">
            <p>some text</p>
        </div>
        <div class="right">
```

```
            <h1>some heading</h1>
            <p>paragraph</p>
        </div>
    </div>
</section>
```

2. Always Use Mobile-First CSS

Write your CSS for **mobile first**, then scale up using media queries.

It keeps the code clean and the UX solid for the majority of users who visit from phones.

```
/* Mobile styles */
.example {
    font-size: 1rem;
}

/* Tablet styles */
@media (min-width: 768px) {
    .example {
        font-size: 1.25rem;
    }
}

/* Desktop styles */
@media (min-width: 1024px) {
    .example {
        font-size: 1.5rem;
```

```
    }
}
```

3. Use Responsive CSS Units (`rem`, `em`)

Using `px` locks values to a fixed size, which breaks flexibility across devices and ignores user accessibility settings.

Instead, use `rem`, `em`, or `%` to make your design **fluid**, **scalable**, and **responsive to browser settings** like Zoom (`Ctrl +`) or custom font size.

```
html {
  font-size: clamp(0.875rem, 1vw + 0.5rem, 1.125rem);
}

h1 {
  font-size: 2rem; /* Scales based on the root size */
}
```

4. Flexbox & Grid

Flexbox and CSS Grid are your best friends for building **responsive layouts.**

You can create stacked layouts for mobile and horizontal ones for bigger screens — without adding a ton of media queries.

```
.responsive-section {
    display: flex;
    flex-direction: column;
}

@media (min-width: 768px) {
    .responsive-section {
        flex-direction: row;
    }
}
```

5. Use `clamp()` for Font Sizes

This ensures that text stays comfortably readable on smaller screens, grows proportionally on tablets and desktops, and avoids becoming overwhelmingly large on 4K or ultra-wide displays. It's the perfect balance between accessibility and aesthetics, making sure your typography adapts naturally while still respecting boundaries.

```
html {
    font-size: clamp(0.875rem, 1vw + 0.5rem, 1.125rem);
```

```
}
```

6. Define CSS Variables (custom properties) for Reusability

This approach centralizes your design tokens, making it easy to update a value in one place and have that change reflected across your entire site.

It also pairs beautifully with media queries — you can redefine variables at different breakpoints for fully responsive design without rewriting the entire CSS block. Plus, when working in a team or maintaining a large project, variables instantly improve readability and reduce bugs caused by inconsistent values.

```
:root {
  --font-size-sm: 0.875rem;
  --font-size-md: 1rem;
  --border-radius-lg: 0.5rem;
}
```

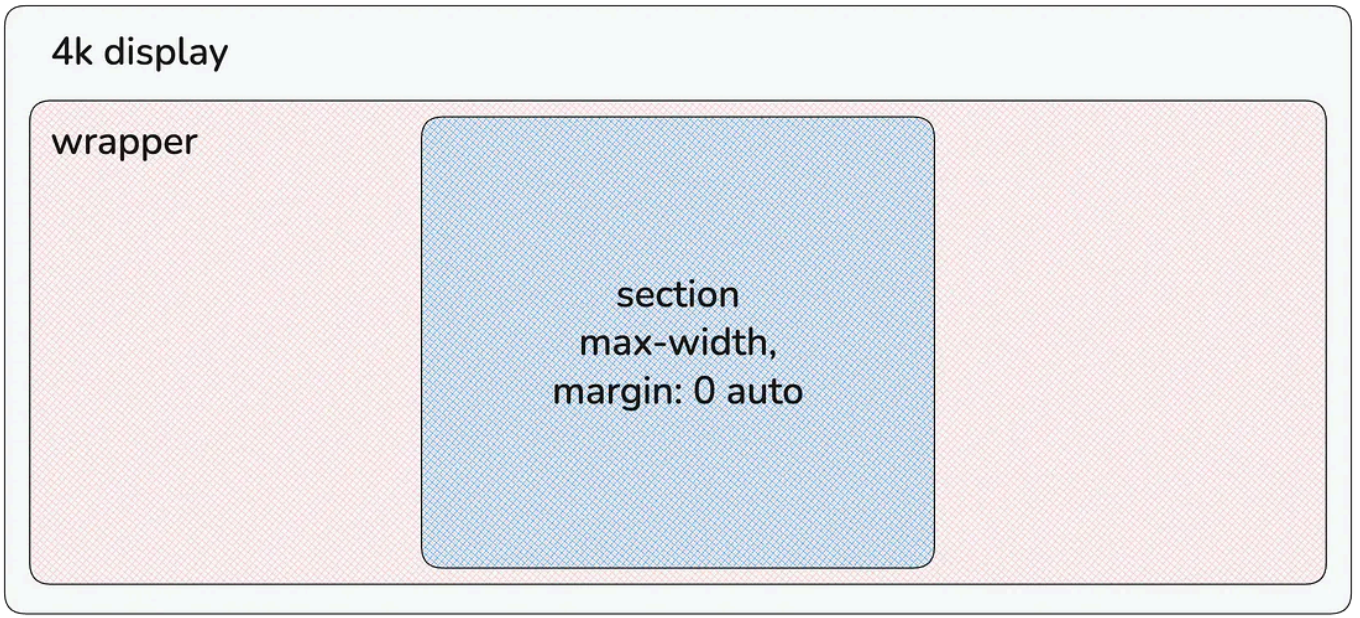
7. Use Wrapper Sections for Centered Layouts

When designing for large screens, it's important to prevent your content from stretching edge-to-edge, which can make it harder to read and visually

unbalanced.

To fix this, wrap your section content inside a container div (commonly called a “wrapper”) and apply a `max-width` along with `margin: auto`. This centers the content and restricts its width, ensuring it stays clean, readable, and aligned — especially on desktop and ultra-wide monitors.

```
.wrapper {
  max-width: 1200px;
  margin: 0 auto;
}
```



It’s a small trick that makes a huge difference in your overall UI polish.

8. Use Section Padding + Media Queries

Consistent vertical spacing makes layouts clean and breathable. Use `padding-top` and `padding-bottom` for each section, and adjust them with media queries. Define values as CSS variables for easy updates across breakpoints

Just make sure your media queries are spaced logically — intervals of 150px to 200px work well in most cases

```
@media (max-width: 800px) {
    section{
        --padding-top: 60px;
        --padding-bottom: 60px;
    }
}

@media (min-width: 800px) and (max-width: 1000px) {
    section{
        --padding-top: 100px;
        --padding-bottom: 100px;
    }
}

@media (min-width: 1000px) and (max-width: 1200px) {
    section{
        --padding-top: 150px;
        --padding-bottom: 100px;
    }
}

@media (min-width: 1200px){
    section{
        --padding-top: 200px;
        --padding-bottom: 150px;
    }
}
```

9. Don’t Be Scared of Media Queries

Media queries are essential for building truly responsive designs. While too many can clutter your code, avoiding them limits flexibility. The key is to

strike a balance — use them where they add real value, organize them cleanly, and combine them with techniques like `flex`, `grid`, and `clamp()` to minimize overuse. When used wisely, media queries help your UI feel tailored across all devices without becoming a maintenance nightmare.

Final Thoughts

Start mobile-first. Use fluid units. Embrace CSS grid/flex. Structure your HTML with intention.

Responsive Web HTML Html Structure Front End Development

rishabh-ml

What are your thoughts?

sarthak
Jun 3

Insightful

5 Reply

Vedrajput
Aug 13

This blog give ideas how to make our website responsive ,Very Helpful for beginners.

Reply

u

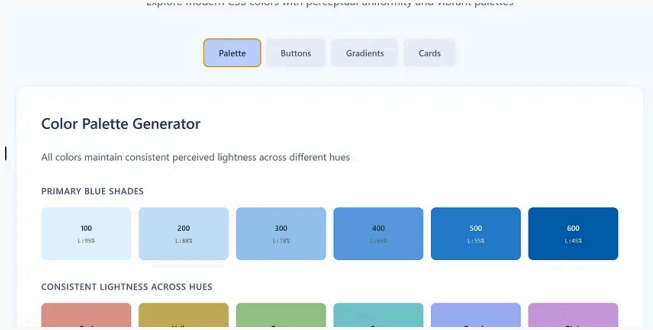
Tiwariudit
Aug 13

insightful

Reply

See all responses

Recommended from Medium



Alexander Burgos

OKLCH: The Modern CSS Color Space You Should Be Using in 2025

Stop Fighting With Colors — Meet OKLCH

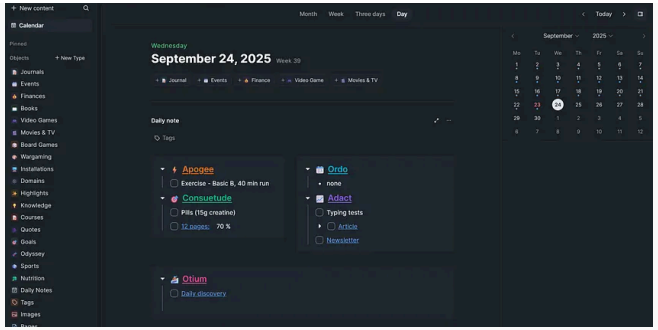
Oct 12 60

In UX Planet by Jason Teunissen

Common UX mistakes everyone still makes 2.0

Back in 2018 I wrote about the most common UX mistakes low-code developers make.

Oct 14 149 3



Tosny

7 Websites I Visit Every Day in 2025

If there is one thing I am addicted to, besides coffee, it is the internet.

Sep 23 4.8K 176

In Bootcamp by Rehan Pinjari

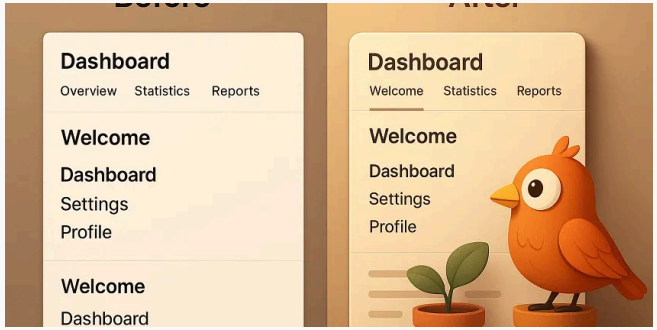
7 UI/UX mistakes that SCREAM you're a beginner (and exactly ho...

Even the best gradients can't hide rookie design habits. Here are 7 common UI/UX...

Oct 9 962 8

Usman Writes

60 Best JavaScript Libraries for Building Interactive UI...



DevLogics

9 CSS Tricks That Instantly Make Any UI Look Professional—Even...

Building a great user interface is more than writing clean code. It's about choosing the...

6d ago 45 5

Stop using browser defaults; three small CSS rules will make your interface look deliberat...

Oct 9 57

See more recommendations