WEB ENGINEERING FUNDAMENTALS

CSS Media Queries

A Complete Beginner's Guide to Building Responsive Websites

This is one of the most important PDFs in your Web Development series because it's where readers move from understanding responsive design concepts to actually implementing them in CSS.

haas.dev • dev-roast-app.vercel.app

Table of Contents

1. Introduction
2. What Are Media Queries?
3. Why Media Queries Matter
4. How Media Queries Work
5. Understanding Breakpoints
6. Common Media Query Features
7. Mobile-First vs Desktop-First
8. Combining Multiple Conditions
9. Real-World Examples
10. Best Practices
11. Common Beginner Mistakes
12. Practical Action Plan
13. Mini Project
14. Key Takeaways
15. Summary Page
16. Media Query Cheat Sheet
17. Related Resources
18. Recommended Next Learning Path

1. Introduction

Imagine opening the same website on three different devices.

On a desktop:

- The navigation menu is fully visible.
- Four product cards appear in a row.
- Images are large.

On a tablet:

- The cards become two per row.
- Some spacing is reduced.

On a phone:

- The navigation becomes a hamburger menu.
- Cards stack vertically.
- Text becomes slightly smaller.
- Buttons become easier to tap.

The website hasn't changed.

The HTML hasn't changed.

The browser is simply applying different CSS rules depending on the screen size.

This is exactly what CSS Media Queries do.

Media Queries are one of the most important features of modern CSS because they allow a single website to adapt to thousands of different devices without maintaining separate versions.

2. What Are CSS Media Queries?

A CSS Media Query is a conditional rule that tells the browser:

> "Apply these styles only when specific conditions are true."

Those conditions might include:

- screen width
- screen height
- orientation (portrait or landscape)
- resolution
- print mode

Think of a Media Query like an if statement for CSS.

If the condition matches, the enclosed styles are applied.

If not, the browser ignores them.

3. Why Media Queries Are Important

Without Media Queries:

- Desktop layouts would simply shrink on phones.
- Text could become unreadable.
- Buttons might be too small to tap.
- Navigation bars could overflow the screen.
- Users would constantly zoom and scroll horizontally.

Media Queries solve these problems by allowing the layout to respond intelligently.

Benefits include:

- Better user experience
- Improved accessibility
- Mobile-friendly interfaces
- Cleaner responsive layouts
- Better SEO through mobile usability
- One codebase for multiple devices

4. How Media Queries Work

The browser continuously monitors the environment.

For example:

- viewport width
- viewport height
- device orientation

Whenever those values match a Media Query, the associated CSS rules become active.

If the conditions change—for example, when the user rotates a phone—the browser recalculates the layout automatically.

5. Understanding Breakpoints

A breakpoint is a screen width where the layout changes.

For example:

- A desktop navigation may become a mobile menu.
- Four cards may become two.
- Font sizes may decrease slightly.

Beginners often think breakpoints are based on specific devices.

Professional developers instead choose breakpoints based on when the design starts to break, not on specific phone or laptop models.

This makes layouts more future-proof.

6. Common Media Query Features

Media Queries can respond to many conditions.

The most commonly used are:

Width

Adjust layouts based on the viewport width.

Height

Useful for devices with limited vertical space.

Orientation

Detect whether the device is:

- Portrait
- Landscape

This is useful for galleries, dashboards, and media-heavy interfaces.

Print

Apply special styles when a page is printed.

For example:

- Hide navigation.
- Remove backgrounds.
- Optimize spacing for paper.

7. Mobile-First vs Desktop-First

There are two common approaches.

Mobile-First

Start with styles for the smallest screens.

Then progressively enhance the layout for larger screens.

Advantages:

- Simpler CSS
- Better performance
- Encourages prioritizing essential content

Desktop-First

Start with desktop styles.

Then override them for smaller devices.

This can work, but it often requires more overrides and additional maintenance.

Modern projects generally prefer the mobile-first approach.

8. Combining Multiple Conditions

Media Queries are not limited to a single condition.

Developers can combine rules to target more specific situations.

Examples include:

- Wide screens in landscape mode
- Small screens in portrait mode
- Large displays with high resolution

Combining conditions provides greater control over responsive behavior.

9. Real-World Examples

E-commerce Store

Desktop:

- Four product cards

Tablet:

- Two product cards

Phone:

- One product card

Blog Website

Desktop:

- Sidebar beside the article

Phone:

- Sidebar moves below the content

haas.dev Learning Platform

Desktop:

- Multi-column resource grid

Tablet:

- Two-column layout

Mobile:

- Single-column list for comfortable reading and tapping

> Learn More
> Read Responsive Web Design Fundamentals to understand the design principles that Media Queries help implement.

10. Best Practices

- Build mobile-first whenever possible.
- Choose breakpoints based on the design, not device names.
- Test layouts by resizing the browser.
- Keep Media Queries organized.
- Avoid writing duplicate styles.
- Use relative units alongside Media Queries for greater flexibility.

11. Common Beginner Mistakes

- Creating too many breakpoints.
- Choosing breakpoints for specific phone models.
- Writing duplicate CSS inside every Media Query.
- Forgetting to test landscape orientation.
- Assuming Media Queries alone make a website responsive.

12. Practical Action Plan

Take one of your existing webpages.

Adapt it for:

- desktop
- tablet
- mobile

Observe:

- typography
- spacing
- navigation
- images
- cards

Document every layout change before writing CSS.

This encourages planning before coding, following the haas.dev philosophy:

```
Understand → Break → Design → Build
```

13. Mini Project

Create a simple landing page with:

- Header
- Navigation
- Hero section
- Three feature cards
- Footer

Make sure the layout changes smoothly between:

- Mobile
- Tablet
- Desktop

Focus on usability rather than visual complexity.

14. Key Takeaways

- Media Queries allow CSS to respond to different environments.
- They are the foundation of responsive implementation.
- Breakpoints should follow the design, not device brands.
- Mobile-first development simplifies responsive CSS.
- Good responsive design improves usability on every screen.

15. Summary Page

Media Query Cheat Sheet

Media Queries apply CSS conditionally.

Common conditions: Width, Height, Orientation, Print

Prefer mobile-first development.

Choose breakpoints based on layout needs.

Test on multiple screen sizes.

16. Media Query Cheat Sheet

A quick-reference checklist you can keep beside your editor while you build.

- [ ] Write mobile styles first, then enhance for larger screens
- [ ] Set breakpoints where the design breaks, not by device name
- [ ] Test both portrait and landscape orientation
- [ ] Combine conditions only when a situation truly needs it
- [ ] Avoid duplicating styles across multiple Media Queries
- [ ] Pair Media Queries with relative units (%, rem, em, vw, vh)
- [ ] Resize the browser window to test, not just presets

17. Related Resources

Responsive Web Design Fundamentals

Why read it: Learn the principles behind responsive layouts before implementing them.

CSS Flexbox

Why read it: Build responsive navigation bars and one-dimensional layouts.

CSS Grid

Why read it: Create responsive two-dimensional page structures.

CSS Display & Positioning

Why read it: Understand how layout behavior changes before adapting it for different screens.

18. Recommended Next Learning Path

```
Step 1

Responsive Web Design Fundamentals
↓
Step 2

CSS Media Queries (Current PDF)
↓
Step 3

Responsive CSS Units (%, rem, em, vw, vh)
↓
Step 4

Responsive Images
↓
Step 5

Mobile-First Design
↓
Step 6

CSS Variables
↓
Step 7

CSS Transitions
↓
Step 8

CSS Animations
```

---

haas.dev • dev-roast-app.vercel.app