**UX Instructions Exercise 2a**

Let's now add **Media Queries and Container Queries** concepts directly into your current project:

1. **Explanation of importance of meta viewport tag**
2. **Multiple media queries (with logical conditions)**
3. **A preview of Level 5 media queries (prefers-color-scheme)**
4. **Basic container query with fallback**

**In index.html (may already included):**

<meta name="viewport" content="width=device-width, initial-scale=1.0”>

**Why this is important:**  
It tells browsers to scale the page to the device’s width, enabling media queries to work correctly on phones/tablets.

**Add to styles.css (near your media queries):**

/\* Media query for very small screens \*/

@media (max-width: 480px) {

.site-header h1 {

font-size: 1.25rem;

}

.intro p {

font-size: 0.9rem;

}

}

/\* Media query for larger desktops \*/

@media (min-width: 1200px) {

.container {

padding-left: 4rem;

padding-right: 4rem;

}

}

/\* Media Query Level 5: Respect user's theme preference \*/

@media (prefers-color-scheme: dark) {

body {

background-color: #111;

color: #f9f9f9;

}

header, footer {

background-color: #444;

}

.main-nav a {

color: #f9f9f9;

}

.card {

background-color: #222;

color: #f9f9f9;

}

}

**Container Query Preview (add at bottom of styles.css):**

/\* Container query demo \*/

.card-container {

container-type: inline-size;

container-name: carddemo;

}

@container carddemo (min-width: 400px) {

.card {

font-size: 1.1rem;

background: #e7f3ff;

}

}

**Make sure to wrap .card elements in a container div:**

<section class="grid card-container">

<!-- your .card articles go here -->

</section>

**Quiz:**

1. What does the meta viewport tag do, and why is it essential for responsive design?
2. How do media queries use conditional logic?
3. What’s the difference between media queries and container queries?
4. What are interaction media features (like hover, pointer) used for?
5. Try changing your system theme—did your site respond to prefers-color-scheme?

**Explanations:**

**meta viewport tag**

This tag controls the visible area of the page on mobile devices.

<meta name="viewport" content="width=device-width, initial-scale=1.0">

**What it does:**

* width=device-width: Sets the layout width to the screen’s width.
* initial-scale=1.0: Sets the zoom level.

Without this, your responsive styles **won’t work correctly** on phones or tablets.

**Media Queries – Syntax Examples**

/\* Target screen widths below 600px \*/

@media (max-width: 600px) {

body {

font-size: 0.9rem;

}

}

/\* Target screens wider than 1000px \*/

@media (min-width: 1000px) {

.container {

padding-left: 4rem;

padding-right: 4rem;

}

}

/\* Combine logic \*/

@media (min-width: 600px) and (max-width: 900px) {

body {

background-color: #f5f5f5;

}

}

**Media Queries in Different Places**

In external CSS:

@media (max-width: 600px) {

body { background: pink; }

}

In <style> tag inside HTML:

<style>

@media (max-width: 600px) {

body { background: pink; }

}

</style>

In imported CSS:

@import url("small.css") screen and (max-width: 600px);

**Media Query Level 5 Features**

**What is pointer: coarse?**

**coarse means:**

The user is interacting with the screen using a **less precise pointing device**, such as a **finger on a touchscreen**.

**Example Devices with pointer: coarse:**

* Most smartphones and tablets
* Touchscreen kiosks

**Media Query Syntax:**

@media (pointer: coarse) {

/\* styles for touch devices \*/

.button {

padding: 1rem; /\* bigger tap target \*/

}

}

**Other Values for pointer:**

|  |  |  |
| --- | --- | --- |
| **Value** | **Meaning** | **Example** |
| none | No pointing device | Voice-only device |
| coarse | Low-accuracy (touch, stylus) | Smartphone, iPad |
| fine | High-accuracy (mouse, trackpad) | Desktop computer |

**Why It Matters**

Using pointer: coarse lets you:

* Enlarge touch targets
* Add more spacing
* Avoid hover-only interactions

@media (prefers-color-scheme: dark) {

body {

background-color: #111;

color: white;

}

}

@media (hover: hover) {

button:hover {

background-color: gold;

}

}

@media (pointer: coarse) {

/\* Devices like phones or tablets with touch \*/

button {

padding: 2rem;

}

}

**Container Queries**

Unlike media queries (which depend on screen size), container queries depend on **the size of the container**.

.card-container {

container-type: inline-size;

container-name: carddemo;

}

@container carddemo (min-width: 400px) {

.card {

background-color: lightblue;

font-size: 1.2rem;

}

}

This allows a component to adapt based on its own container size, not the full page!

Additional hands-on:

**Container Query Tuning**

1. Wrap your .card section in:  
     
   <section class="grid card-container"> ... </section>
2. Add to styles.css:  
   .card-container {

container-type: inline-size;

container-name: carddemo;

}

@container carddemo (min-width: 400px) {

.card {

background-color: #dff0ff;

padding: 2rem;

}

}

1. Resize the container using Developer Tools and observe how .card adapts independently of the page width.

**Lab Activity: Dark Mode with prefers-color-scheme**

1. Add to styles.css:  
     
   @media (prefers-color-scheme: dark) {

body {

background: #111;

color: #eee;

}

header, footer {

background: #333;

}

.card {

background: #222;

}

}

1. Change your system theme (macOS, Windows, or Chrome) to dark and reload the page.

**Optional Challenge: Add hover and pointer media queries**

@media (hover: hover) {

.main-nav a:hover {

text-decoration: underline;

}

}

@media (pointer: coarse) {

.main-nav a {

font-size: 1.25rem;

padding: 1rem;

}

}

**How to Simulate prefers-color-scheme in Chrome:**

1. **Open your site in Chrome**
2. Right-click and select **"Inspect"** (or press Cmd + Option + I on Mac / Ctrl + Shift + I on Windows)
3. Click the **“…” (3-dot menu)** in the top right of DevTools
4. Go to **More tools** → **Rendering**
5. Scroll to **“Emulate CSS media feature prefers-color-scheme”**
6. Choose:
   * no-preference
   * light
   * dark

**What to Look For:**

Your styles like:

@media (prefers-color-scheme: dark) {

body {

background-color: #111;

color: #f9f9f9;

}

}

…will activate instantly when “dark” is selected.

**Advanced Enhancements**

**1. Interaction Media Features**

Let’s detect whether the user **can hover** or **has fine pointer precision**:

@media (hover: hover) and (pointer: fine) {

.card:hover {

transform: translateY(-5px);

box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);

transition: all 0.2s ease-in-out;

}

}

This gives enhanced hover effects only for users who actually have hover-capable input (like a mouse), skipping touch devices.

**2. User Preference Media Features**

You already use:

@media (prefers-color-scheme: dark) {

/\* Dark mode styles \*/

}

You can add:

@media (prefers-reduced-motion: reduce) {

\* {

animation: none !important;

transition: none !important;

}

}

Respects users who request minimal motion due to motion sickness or accessibility needs.

**3. Split Media Queries into Their Own Files (Optional)**

You could break your CSS into:

* base.css
* dark-mode.css
* reduced-motion.css
* layout-large.css

Then include conditionally, Good for modular design, but more HTTP requests unless bundled.

<link rel="stylesheet" href="dark-mode.css" media="(prefers-color-scheme: dark)">

**4. Custom Container Queries (Expand)**

Right now, you use:

.card-container {

container-type: inline-size;

container-name: carddemo;

}

@container carddemo (min-width: 400px) {

.card {

font-size: 1.1rem;

background: #e7f3ff;

}

}

You can expand this into different containers. This gives you **component-level control** instead of relying on global breakpoints:

@container carddemo (min-width: 600px) {

.card {

display: flex;

gap: 1rem;

align-items: center;

}

.card img {

width: 50%;

}

.card p {

flex: 1;

text-align: left;

}

}

**5. Add Visible Feedback for Viewport & Breakpoint**

In your script.js, add a little debug display:

const bp = document.createElement('div');

bp.textContent = window.innerWidth + "px";

bp.style.position = "fixed";

bp.style.bottom = "0";

bp.style.right = "0";

bp.style.background = "#000";

bp.style.color = "#fff";

bp.style.padding = "4px";

bp.style.zIndex = "9999";

document.body.appendChild(bp);

window.addEventListener('resize', () => {

bp.textContent = window.innerWidth + "px";

});

Below is your complete responsive landing page, enhanced with features:

* Viewport meta tag
* Media queries (basic, advanced, user preferences)
* Container queries
* Skip link
* Accessible navigation
* Responsive typography with clamp()
* Fluid layout using CSS Grid

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Responsive Web Lab</title>

<link rel="stylesheet" href="styles.css" />

</head>

<body>

<a href="#main" class="skip-link">Skip to main content</a>

<nav class="main-nav" role="navigation" aria-label="Main navigation">

<button aria-label="Toggle navigation menu" class="nav-toggle" onclick="document.body.classList.toggle('menu-open')">☰ Menu</button>

<ul class="nav-list">

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Courses</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

<header class="site-header">

<h1>Responsive Web Lab</h1>

</header>

<main id="main" class="container">

<section class="intro">

<h2>Welcome</h2>

<p>This page demonstrates responsive techniques including flexible images, fluid grids, media queries, and user preference detection.</p>

</section>

<section class="grid card-container">

<article class="card">

<img src="images/sample.jpg" alt="Sample responsive" />

<p>This card scales on any screen.</p>

</article>

<article class="card">

<img src="images/sample.jpg" alt="Another example" />

<p>Try resizing the browser window.</p>

</article>

<article class="card">

<img src="images/sample.jpg" alt="Responsive image" />

<p>Images and layout adjust accordingly.</p>

</article>

</section>

</main>

<footer class="site-footer">

<p>&copy; 2025 Responsive Design Lab</p>

</footer>

<script src="script.js"></script>

</body>

</html>

**styles.css**

:root {

--primary-color: #0076ce;

--padding: 1rem;

--max-width: 1200px;

}

\* {

box-sizing: border-box;

}

body {

font-family: sans-serif;

margin: 0;

line-height: 1.6;

font-size: clamp(1rem, 1vw + 0.5rem, 1.2rem);

padding: var(--padding);

color: #111;

background-color: #f9f9f9;

}

header, footer {

background-color: var(--primary-color);

color: white;

text-align: center;

padding: 1rem 0;

}

.container {

max-width: var(--max-width);

margin: 0 auto;

}

.main-nav {

position: sticky;

top: 0;

background: #eee;

z-index: 1000;

}

.nav-toggle {

display: none;

background: none;

border: none;

font-size: 1.2rem;

padding: 1rem;

cursor: pointer;

}

.nav-list {

list-style: none;

margin: 0;

padding: 0;

display: flex;

gap: 1rem;

justify-content: center;

}

.main-nav a {

text-decoration: none;

color: var(--primary-color);

padding: 0.5rem 1rem;

display: inline-block;

}

.grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 1.5rem;

margin-top: 2rem;

}

.card {

background-color: white;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

padding: 1rem;

text-align: center;

}

.card img {

max-width: 100%;

height: auto;

}

.skip-link {

position: absolute;

top: -40px;

left: 0;

background: #000;

color: #fff;

padding: 8px;

z-index: 100;

transition: top 0.3s;

}

.skip-link:focus {

top: 0;

}

@media (max-width: 768px) {

.nav-toggle {

display: block;

}

.nav-list {

display: none;

flex-direction: column;

background: #eee;

}

body.menu-open .nav-list {

display: flex;

}

}

@media (min-width: 1000px) {

.card {

padding: 2rem;

font-size: 1.2rem;

}

}

/\* User Preference: Dark Mode \*/

@media (prefers-color-scheme: dark) {

body {

background-color: #111;

color: #f9f9f9;

}

header, footer {

background-color: #444;

}

.main-nav a {

color: #f9f9f9;

}

.card {

background-color: #222;

color: #f9f9f9;

}

}

/\* Reduced Motion Accessibility \*/

@media (prefers-reduced-motion: reduce) {

\* {

animation: none !important;

transition: none !important;

}

}

/\* Extra small device tweaks \*/

@media (max-width: 480px) {

.site-header h1 {

font-size: 1.25rem;

}

.intro p {

font-size: 0.9rem;

}

}

/\* Extra wide layout \*/

@media (min-width: 1200px) {

.container {

padding-left: 4rem;

padding-right: 4rem;

}

}

/\* Container Queries \*/

.card-container {

container-type: inline-size;

container-name: carddemo;

}

@container carddemo (min-width: 400px) {

.card {

font-size: 1.1rem;

background: #e7f3ff;

}

}

/\* Interaction Media Features \*/

@media (hover: hover) and (pointer: fine) {

.card:hover {

transform: translateY(-5px);

box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);

transition: all 0.2s ease-in-out;

}

}

**script.js**

// Optional: show current viewport width

const bp = document.createElement('div');

bp.textContent = window.innerWidth + "px";

bp.style.position = "fixed";

bp.style.bottom = "0";

bp.style.right = "0";

bp.style.background = "#000";

bp.style.color = "#fff";

bp.style.padding = "4px";

bp.style.zIndex = "9999";

document.body.appendChild(bp);

window.addEventListener('resize', () => {

bp.textContent = window.innerWidth + "px";

});

**Troubleshooting and more explanations:**

**Emulate prefers-color-scheme (Light/Dark Mode)**

1. Open your webpage in Chrome.
2. Right-click → **Inspect** (or press Cmd+Option+I on Mac / Ctrl+Shift+I on Windows).
3. Click the **three-dot menu** (︙) in the top-right corner of DevTools.
4. Go to **More tools** → **Rendering**.
5. Scroll to **“Emulate CSS media feature prefers-color-scheme”**.
6. Choose:
   * light
   * dark
   * no-preference

This triggers any @media (prefers-color-scheme: dark) or light styles in your CSS.

**2. Emulate prefers-reduced-motion**

1. In the **same Rendering panel** of DevTools.
2. Look for **“Emulate CSS media feature prefers-reduced-motion”**.
3. Choose:
   * reduce – disables transitions/animations.
   * no-preference – allows animations as normal.

Use this to test how your page responds to:

@media (prefers-reduced-motion: reduce) {

\* {

animation: none !important;

transition: none !important;

}

}

**3. Emulate hover and pointer media features**

Still in the **Rendering tab**, under **Emulate CSS media features**:

* For hover, test with devices that:
  + **Have hover** (mouse)
  + **Don’t have hover** (touchscreen)
* For pointer, test:
  + **Fine pointer** (mouse)
  + **Coarse pointer** (touch)

These affect rules like:

@media (hover: hover) and (pointer: fine) {

.button:hover {

background-color: blue;

}

}

**How to Enable the “Rendering” Panel in Chrome DevTools if Hidden**

1. **Open DevTools**
   * Right-click on any page → **Inspect**
   * Or press Ctrl+Shift+I (Windows/Linux) or Cmd+Option+I (Mac)
2. **Show the Command Menu**
   * Press Cmd+Shift+P (Mac) or Ctrl+Shift+P (Windows/Linux)
   * This opens the **command palette**
3. **Type: rendering**
   * Select **“Show Rendering”** from the dropdown
   * It will open a new “Rendering” tab at the bottom of DevTools
4. **Scroll down in the Rendering tab**
   * Find these options:
     + **Emulate CSS media feature prefers-color-scheme**
     + **Emulate CSS media feature prefers-reduced-motion**
     + (Also: hover, pointer, and other media features)

**Scroll in the “Rendering” Tab:**

1. **Open the Rendering panel** using:
   * Cmd+Shift+P → "Show Rendering"
2. **Hover your mouse over the panel** — it appears at the bottom of DevTools.
3. Use:
   * **Mouse scroll wheel**
   * **Trackpad gesture**
   * Or grab the **scrollbar** on the right side (if visible)
4. Keep scrolling until you see:
   * **“Emulate CSS media feature prefers-color-scheme”**
   * **“Emulate CSS media feature prefers-reduced-motion”**

**Tip:**

The rendering panel is often **not very tall by default**, so if you **don’t see a scrollbar**, you may need to:

* Drag the top edge of the panel upward to **make it taller**, and then scroll.
* Or click inside the panel and press **down arrow** on your keyboard to scroll line-by-line.

**Hamburger Menu Breakdown**

**1. HTML Structure**

In your <nav>:

<nav class="main-nav" role="navigation" aria-label="Main navigation">

<button

aria-label="Toggle navigation menu"

class="nav-toggle"

onclick="document.body.classList.toggle('menu-open')">

☰ Menu

</button>

<ul class="nav-list">

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Courses</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

* The button.nav-toggle is the hamburger icon (☰).
* It toggles a menu-open class on the <body> when clicked.

**2. CSS: Show/Hide Based on Screen Size**

**Default (desktop view):**

.nav-toggle {

display: none;

}

.nav-list {

display: flex;

}

**In media query for small screens (e.g., max-width: 768px)**

@media (max-width: 768px) {

.nav-toggle {

display: block;

}

.nav-list {

display: none;

flex-direction: column;

background: #eee;

padding: 1rem;

}

body.menu-open .nav-list {

display: flex;

}

}

**3. JavaScript**

Just this in the onclick of the <button>:

document.body.classList.toggle('menu-open')

This adds/removes the class menu-open on <body>, which triggers the mobile menu to show/hide.

**Accessibility Notes**

* aria-label="Toggle navigation menu" on the button provides screen readers with a description.
* The nav uses role="navigation" and aria-label="Main navigation".