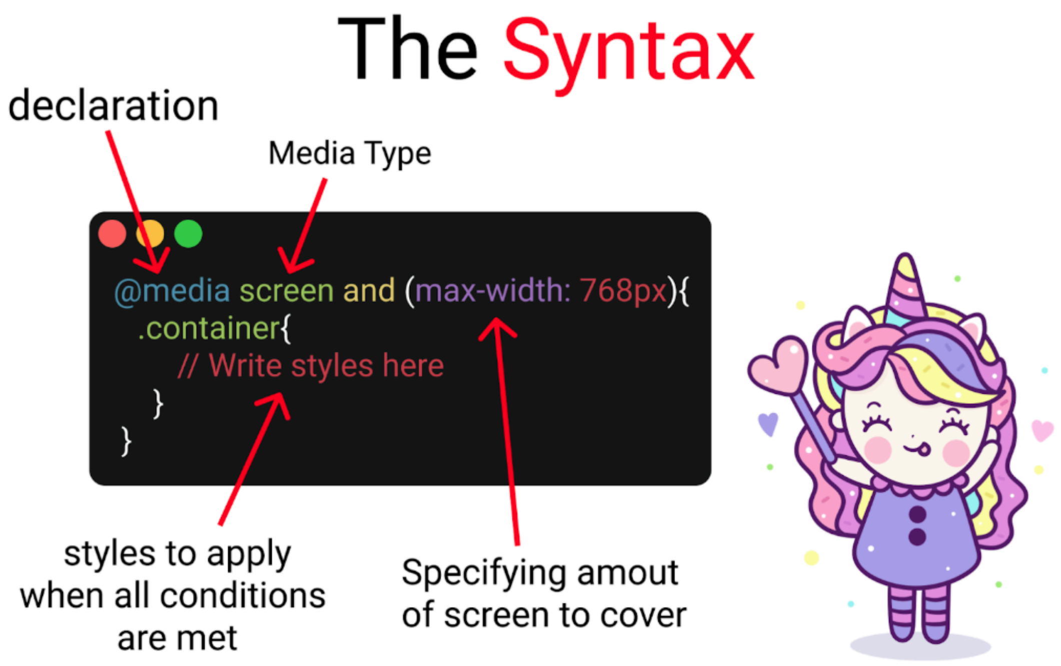
🧠 Activity: Know Your Viewport: Responsive Design with Media Queries

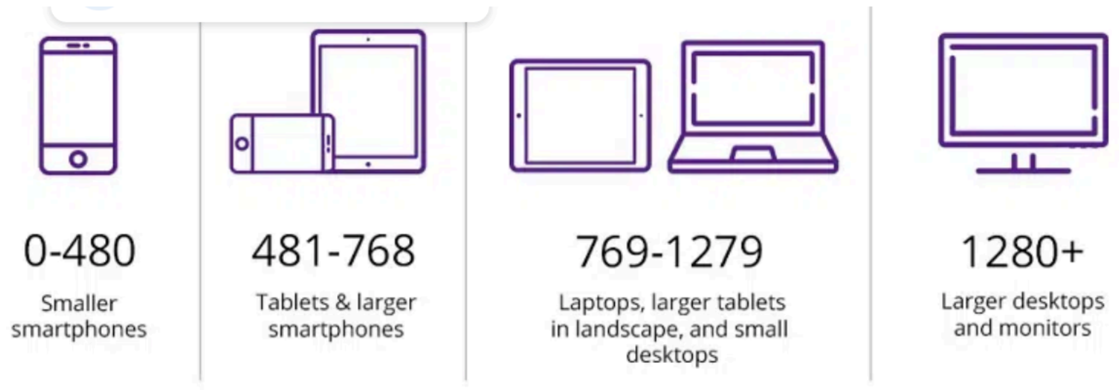
Duration: 60 minutes
Objective: Understand and apply media queries, pseudo-elements, and responsive design patterns in CSS.

Media Query Codepen: https://codepen.io/Mido-Sayed/pen/wBKpXdR



```css
@media screen and (max-width: 768px){
  .container{
     // Write styles here
   }
}
```

The Syntax
declaration
Media Type
styles to apply when all conditions are met
Specifying amout of screen to cover


0-480
Smaller smartphones
481-768
Tablets & larger smartphones
769-1279
Laptops, larger tablets in landscape, and small desktops
1280+
Larger desktops and monitors

🔍 Part 1: Theory & Conceptual Exploration (20 minutes)

Answer the following questions to demonstrate your understanding of the CSS and HTML used in the demo. Use your own words. You may research online if needed.

📘 Section A: CSS Pseudo-elements

1. What does the `::after` pseudo-element do in general? It provides a condition to the CSS that takes effect after those conditions are met.

2. What's the difference between `::after` and `::before`? One happens before the specific conditions are met while the other happens after all conditions are met.

3. Why does `.container::after` still work even though the container has no inner content? Because the content will display after the conditions are met.

4. Can we style a pseudo-element like a real element (e.g., padding, font-size)? Are there limitations?
   Yes, but theory can only accept most CSS properties but they are limited when using classes, they can't be directly changed using javascript, "content" has to be added if you want it to display even when there is nothing to go in it. Can't chain them together. Accessibility tech in browsers may not interpret them properly.

🌐 Section B: Media Queries and Responsive Design

5. What is the purpose of a media query in CSS? To use up all of the available space on the screen when the screen sizes change across devices.

6. What does the following mean?

```
@media only screen and (min-width: 600px) and (max-width: 899px)
```

It means that this screen adjustment or change will only apply when the screen has a min-width of 600px and a max-width of 899px.

7. Why do developers typically use three main breakpoints (mobile, tablet, desktop)?
   It keeps all of the content readable and visibly pleasing for the user on the screen. It also provides the necessary information as the screen shrinks to eliminate clutter on the page.Provides faster loading and better accessibility.

8. What is the **mobile-first approach** and how would you modify this code to follow it?

Starting with specs for a mobile screen size and then layering additional styles that take effect when the screen size changes.

9. What would happen if two media queries overlap? Which rule will the browser use?

The standard CSS rule applies and determines which style takes precedence.

🎯 Section C: Visual Understanding

Look at the original CSS and explain:

10. What message or behaviour is communicated to users on:

- Mobile view - The screen changes to blue and the content displays mobile view.
- Tablet view - The screen changes to green content displays tablet view.
- Desktop view - The screen changes to red content displays desktop view.

11. How is the text in `.container::after` changing across breakpoints?

It changes when the conditions are met for each screen size transition.

12. What other properties (besides `background-color` and `content`) could you change across breakpoints to improve UX?

The padding, margins, display and font color can all be changed to improve UX.

💻 Part 2: Implementation Challenge (35 minutes)

You will build on the demo and create your own **responsive web section** with a visual indicator of the current device category. The goal is to *demonstrate how layout and style adapt based on viewport width*.

✅ Step 1: Add This HTML to `index.html`

```
<div class="device-box"></div>
```

✅ **Step 2: Extend `styles.css` with the following base**

```
.device-box::after {
  content: "Default";
  display: block;
  font-size: 2em;
  text-align: center;
  padding: 60px;
  background-color: lightgray;
  border: 3px solid black;
}
```

✅ **Step 3: Add the following responsive behaviour**

- **Mobile (max-width: 599px):**
  - Change background color to light blue
  - Text: `"Mobile Device"`
  - Font-size: `1.5em`
- **Tablet (600px–899px):**
  - Background: light green
  - Text: `"Tablet Device"`
  - Add a dotted border
- **Desktop (min-width: 900px):**
  - Background: light coral
  - Text: `"Desktop Device"`

    - Add padding and center the box in the viewport

🔄 Part 3: Reflection & Share (5 minutes)

In pairs or breakout rooms:

1. Show each other your screen at different breakpoints.
2. Discuss:
    - What worked well in your implementation?
    - What would you do differently if this were a production site?
    - How can this technique help users on slow connections or small screens?

🧩 Optional Bonus (Advanced)

- Add a **fourth view**: large desktops (`min-width: 1200px`) and show `"Ultra-Wide Device"`.
- Use Flexbox or Grid to center the `.device-box` on all screens.
- Add a subtle animation when changing views (e.g., fade-in).