Day 14 of training

Mini Project 2: Building a Portfolio Website (Part 2 - Styling & Animations)

1. **Advanced Styling with Pseudo-elements**
    - Today's work on the portfolio project focused on the finer details of styling and bringing the page to life with motion. A key technique used was **pseudo-elements**, specifically ::before and ::after, to create sophisticated hover effects without adding extra HTML.
    - **Button Hover Effect:** The "Hire me" button has a creative hover effect where a white background appears to slide in from the left. This is achieved by:
        1. Giving the .button a position: relative; and a transparent background.
        2. Creating a ::before pseudo-element with position: absolute;, a white background, and dimensions that cover the entire button.
        3. Initially, the ::before element is scaled to zero horizontally using transform: scaleX(0); and its transform-origin is set to left.
        4. On hover (.button:hover::before), its scale is changed to transform: scaleX(1);. A transition on the transform property makes this scaling effect smooth, creating the slide-in animation.
    - **Project Card Overlay:** A similar technique is used for the project cards. A ::before pseudo-element with a linear-gradient background is used to create a colored overlay. On hover, this overlay scales from scaleX(0) to scaleX(1), creating a stylish wipe effect that reveals more information.
2. **Creating Dynamic Motion with CSS Animations and Keyframes**
    - The hero section is full of life thanks to **CSS animations**. These are created using the @keyframes at-rule, which allows you to define the stages of an animation.
    - **Defining an Animation (@keyframes):** A set of keyframes is defined with a name (e.g., uimage). Inside, you specify the styles at different points of the animation, like 0% (the start), 50% (the middle), and 100% (the end).
    - **Applying an Animation:** The animation property is then used to apply these keyframes to an element. It's a shorthand that can include:
        - animation-name: The name of the @keyframes rule.

        - animation-duration: How long the animation takes (e.g., 4s).
        - animation-timing-function: The speed curve of the animation (e.g., linear).
        - animation-iteration-count: How many times it should repeat (e.g., infinite).
    - **Examples from the Project:**
        - **User Image Animation (uimage):** The user's image slowly scales up and gains a grayscale filter at the 50% mark, then returns to its original state. This creates a subtle, pulsing effect.
        - **Icon Animations:** Each decorative icon has its own unique animation. The dots (dota) move up and down using transform: translateY(). The cube (cuba) rotates on its Y-axis using transform: rotateY(). These small, continuous movements make the page feel dynamic and engaging.
3. **Smooth Interactions with CSS Transitions**
    - While animations are for continuous motion, **CSS Transitions** are used to create smooth changes between states, typically on hover.
    - **Hover Effects on Links and Logos:** The navigation links have a transition on their font-weight. When you hover, the weight changes smoothly from normal to bold, rather than snapping instantly.
    - **Skill Logos:** Each skill logo in the "Skills" section has a transition property. On hover, a transform: scale(1.2); is applied, making the icon grow slightly. The transition ensures this scaling is a smooth and satisfying animation.
4. **JavaScript Integration for a Dynamic Typing Effect**
    - The project includes a small but very effective piece of JavaScript to create a "typing" animation for the developer's role in the hero section (e.g., "Web Developer," "Programmer").
    - **External Library (Typed.js):** Instead of writing this complex logic from scratch, the project cleverly integrates a popular external JavaScript library called **Typed.js**. This is done by adding a <script> tag that links to the library's file from a CDN.
    - **Initializing the Script:** A second, small <script> block is added to the HTML file. This script creates a new Typed instance, tells it which element to target (.role), and provides an array of strings to type out, along with options like

loop, typeSpeed, and backSpeed. This is a great example of how a small amount of JavaScript can add a highly professional and dynamic feature to a website.