Day 9 of training

Advanced CSS Units & Positioning

1. **Relative Units in CSS**
    - Today's lesson built upon the concept of CSS units, moving from absolute units (like pixels) to **relative units**. Relative units define sizes based on a relation to another measurement, making designs more flexible and responsive.
    - **% (Percentage):** This unit defines a size as a percentage of a parent element's property. For example, setting width: 50%; on a child element means it will take up 50% of its parent's width. This is crucial for creating fluid layouts that adapt to different screen sizes.
    - **em:** This unit is primarily used for text-related properties and is relative to the font-size of the **parent element**. If the parent has a font-size of 16px, then 1em for a child would also be 16px. If the child's font-size is 0.5em, it would be 8px. For other properties like width, 1em refers to the element's *own* font-size.
    - **rem (Root EM):** This is another key relative unit, and it's relative to the font-size of the **root <html> element**. This makes rem more predictable than em because its base size is consistent across the entire document, regardless of nested parent elements. This is often preferred for overall typographic sizing.
    - **Viewport Units (vw and vh):**
        - **vw (Viewport Width):** Represents 1% of the viewport's width. Setting width: 50vw; means the element will take up 50% of the browser window's width.
        - **vh (Viewport Height):** Represents 1% of the viewport's height. Setting height: 100vh; means the element will take up 100% of the browser window's height. These units are invaluable for creating elements that scale directly with the browser window.
2. **CSS Position Property**

- The position property is fundamental for precisely placing elements on a webpage. It defines how an element is positioned within a document and affects how other layout properties (like top, bottom, left, right) work.
- **static (Default):** This is the default positioning for all HTML elements. Elements are positioned according to the normal flow of the document. The top, right, bottom, and left properties have no effect on static elements.
- **relative:** An element with position: relative; is positioned relative to its normal position. Using top, right, bottom, and left properties will shift the element from where it *would normally be* in the document flow. Crucially, the space it normally occupies remains, so other elements won't fill that gap.
- **absolute:** An element with position: absolute; is removed from the normal document flow. It's then positioned relative to its *closest positioned ancestor* (any ancestor element that has a position value other than static). If no such ancestor exists, it's positioned relative to the initial containing block (the <html> element). Absolutely positioned elements can overlap other elements.
- **fixed:** An element with position: fixed; is removed from the normal document flow and is positioned relative to the **viewport** (the browser window). It stays in the same place even when the page is scrolled. This is commonly used for persistent headers, footers, or navigation bars.
- **sticky:** This is a hybrid. An element with position: sticky; behaves like a relative element until it hits a specified scroll position, at which point it becomes "sticky" and behaves like a fixed element. This is useful for elements that need to stay in view as you scroll, like a table header.