Web development

lesson 9

Oct 27, 2024

CSS Units: Introduction

- **What Are CSS Units?**
  - CSS units define the measurements for properties like width, height, margin, padding, and font-size.
  - There are two main types: **Absolute Units** and **Relative Units**.
  - Choosing the right unit affects responsiveness, accessibility, and user experience.
- **Types of CSS Units:**
  - **Absolute Units**: Fixed measurements (e.g., px, in).
  - **Relative Units**: Measurements relative to other elements or the viewport (e.g., %, em, vw).

Absolute Units: Pixels (px)

- **What Are Pixels?**
  - Pixels (`px`) are the most common absolute unit, representing a single dot on the screen.
  - Not affected by parent elements, viewport, or user settings (e.g., zoom).
- **Use Cases:**
  - Precise control for small elements or fine details.
  - Fixed layouts where consistent size is essential.
- **Disadvantages:**
  - Less flexible and can impact responsiveness, especially on different devices or screen resolutions.

```
.fixed-box {
  width: 300px;
  height: 150px;
}
```

Absolute Units: Points (pt), Inches (in), and Centimeters (cm)

- **Points (pt):**
  - Primarily used for print media; 1pt = 1/72 of an inch.
  - Common in print style sheets but rare in digital content.
- **Inches (in) and Centimeters (cm):**
  - Absolute physical units rarely used in web design due to varying screen sizes.
  - Ideal for physical measurements in print layouts.
- **Examples:**

```
.print {
  font-size: 12pt;
  width: 8.5in;
  height: 11in;
}
```

Relative Units: Percentages (%)

- **How Percentages Work:**
  - Percentages (`%`) are relative to the parent element's size.
  - Often used for responsive layouts, allowing elements to scale based on their container.
- **Use Cases:**
  - Defining widths, heights, and margins in fluid layouts.
  - Responsive typography and containers.
- **Example:**

```
.responsive-box {
  width: 50%;
  height: 30%;
}
```

  - **Caution:**: Ensure parent elements have a defined size, as percentages depend on them.

Relative Units: em and rem

- **What Are em and rem?**
  - `em` : Relative to the font-size of the nearest parent.
  - `rem` : Relative to the font-size of the root element (`html`), unaffected by other parent elements.
- **Use Cases:**
  - `em` : Adaptive spacing and responsive typography.
  - `rem` : Consistent sizing across components while respecting global scaling.
- **Example:**

```
.text {
  font-size: 2em; /* 2x parent font-size */
}
.text-large {
  font-size: 1.5rem; /* 1.5x root font-size */
}
```

Viewport Units: vw, vh, vmin, vmax

- **Viewport Units Overview:**
  - `vw` : Viewport width (1vw = 1% of the viewport width).
  - `vh` : Viewport height (1vh = 1% of the viewport height).
  - `vmin` and `vmax` : Minimum and maximum of `vw` and `vh` .
- **Use Cases:**
  - Full-width or full-height elements, especially in responsive designs.
  - Dynamic scaling for elements based on viewport size.
- **Example:**

```
.full-screen {
  width: 100vw;
  height: 100vh;
}
```

Relative Units: ex, ch

- **Understanding ex and ch:**
  - `ex` : Relative to the x-height (height of lowercase "x") of the font. Useful for text-specific scaling.
  - `ch` : Width of the "0" character. Useful for creating character-based widths.
- **Use Cases:**
  - Designing with typography in mind.
  - Creating elements that scale based on text characteristics.
- **Example:**

```
.text-box {
  width: 30ch; /* Suitable for limiting characters */
}
```

Advanced Units: calc() Function

- **What is calc()?**
  - `calc()` allows for complex calculations involving different units (e.g., `px`, `em`, `%`).
  - Useful for creating adaptive layouts without hardcoding values.
- **Syntax and Use Cases:**
  - Syntax: `calc(expression)`
  - Combines relative and absolute units for custom layouts, margins, and padding.
- **Example:**

```
.dynamic-width {
  width: calc(100% - 50px);
}
```

  - Requires spaces around operators (+, -, *, /).
  - May not be supported by very old browsers, though widely accepted in modern ones.

CSS Borders: Basics

- **What Are CSS Borders?**
  - Borders are lines that surround an HTML element, providing a visual boundary.
  - You can control the **width**, **style**, and **color** of borders.
  - Borders can be applied to all sides of an element or to specific sides (top, right, bottom, left).
- **Basic Border Properties:**
  - `border` : A shorthand property to define width, style, and color.
  - `border-width` : Defines the thickness of the border.
  - `border-style` : Defines the style (e.g., solid, dotted).
  - `border-color` : Defines the color of the border.

```
.box {
  border: 2px solid black;
}
```

Border Width

- The `border-width` property sets the thickness of the border.
- You can specify the width in pixels (`px`), ems (`em`), or other units.
- You can define a uniform width for all sides or specify different widths for each side.
- Common units: `px` , `em` , `%` .
- Keywords: `thin` , `medium` , `thick` (default values).

```
.box {
  border-width: 5px; /* Uniform width on all sides */
}
.box-sides {
  border-width: 5px 10px 5px 0; /* Different widths: top, right, bottom, left */
}
```

- **Tip:** - Adjusting `border-width` can help with layout and design precision, especially when dealing with responsive layouts.

Border Style

- The `border-style` property defines the appearance or pattern of the border.
- Multiple styles are available: `solid`, `dashed`, `dotted`, `double`, `groove`, `ridge`, `inset`, `outset`, and `none`.
- **Solid**: A single, continuous line.
- **Dashed**: A line made of dashes.
- **Dotted**: A line made of dots.
- **Double**: Two parallel solid lines.
- **None**: No border (used to override inherited borders).

```
.solid-border {
  border-style: solid;
}
.dashed-border {
  border-style: dashed;
}
```

- **Tip:** - Use `border-style` to differentiate elements visually, especially for dividing sections of a page or creating emphasis.

Border Color

- The `border-color` property sets the color of the border.
- You can specify a single color for all sides, or different colors for each side.
- Named colors (`red` , `blue` , `black`).
- Hex values (`#ff0000` , `#3498db`).
- RGB, RGBA, HSL, HSLA values for transparency or specific shades.

```
.colored-border {
  border-color: red;
}
.multi-colored-border {
  border-color: red green blue yellow; /* Different colors for each side */
}
```

- **Tip:** - `rgba()` allows for transparent borders, creating effects like semi-transparent overlays.

Shorthand: border Property

- `border` is a shorthand property that lets you define `border-width`, `border-style`, and `border-color` in a single declaration.
- Syntax: `border: width style color;`

```
.box {
  border: 3px solid red; /* Width, style, color in one line */
}
```

- Use `border-top`, `border-right`, `border-bottom`, and `border-left` to define borders for specific sides.
- Example:

```
.top-border {
  border-top: 4px dotted blue;
}
```

- **Tip:** - Shorthand makes your code more concise, but for greater control over specific sides, use individual properties.

Border Radius

- `border-radius` creates rounded corners on elements.
- You can apply a uniform radius to all corners, or define specific values for each corner.
- Pixels (`px`), percentages (`%`), and ems (`em`).
- A percentage creates an ellipse, commonly used for circular elements (e.g., `50%` for circles).

```
.rounded-box {
  border-radius: 10px;
}
.circle {
  width: 100px;
  height: 100px;
  border-radius: 50%;
}
```

- **Tip:** - Combining `border-radius` with `box-shadow` can create button-like effects or emphasize elements.

Border Image

- The `border-image` property allows you to use an image as the border of an element.
- This property slices an image into sections and places those sections around the element.
- `border-image-source` : The image URL.
- `border-image-slice` : Specifies how the image is sliced for placement around the border.

```
.image-border {
  border: 10px solid;
  border-image: url('border-image.png') 30 round;
}
```

- **Tip:** - `border-image` is great for adding creative and decorative borders to enhance visual design.

Advanced: Box Shadows and Outlines

- `box-shadow` allows you to create a shadow around an element, often combined with borders for depth.
- Syntax: `box-shadow: h-offset v-offset blur color;`

```
.box-shadow {
  border: 1px solid black;
  box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.5);
}
```

- **Outlines vs Borders:**
  - `outline` is similar to `border`, but doesn't affect the box model (it doesn't occupy space).
  - Useful for accessibility as it highlights elements during keyboard navigation.

```
.outlined-box {
  outline: 2px dashed orange;
}
```

- **Tip:** - `box-shadow` is perfect for adding depth and highlighting interactive elements. Use `outline` for focus states in forms and navigation.