**📚 Detailed Web Development Notes**

**🎯 CSS 2D Transforms**

CSS 2D transforms allow you to manipulate elements in two-dimensional space without affecting the document flow. These transforms can move, rotate, resize, or skew elements to create dynamic and engaging layouts or animations.

\*\*`translate(x, y)`\*\* Moves (shifts) an element along the horizontal (X) and vertical (Y) axes by specified lengths. This visually repositions the element but does not change its actual document flow position. `transform: translate(50px, 20px);` moves the element 50 pixels right and 20 pixels down

\*\*`rotate(angle)`\*\* Rotates the element clockwise or counterclockwise around its center point by a specified angle in degrees. `transform: rotate(45deg);` spins the element 45 degrees clockwise

\*\*`scale(x, y)`\*\* Resizes the element by scaling it horizontally and vertically. Values >1 enlarge, values <1 shrink`transform: scale(1.5, 0.8);` — 1.5 times wider, 80% of original height \*\*`skew(x-angle, y-angle)`\*\* Tilts the element by skewing it along the X and/or Y axes, creating a slanted effect. Useful for stylized effects. `transform: skew(20deg, 10deg);` skews 20 degrees along X-axis, 10 degrees along Y-axis

\*\*Note:\*\* Multiple transforms can be combined by separating them with spaces:

```css

transform: translate(20px, 20px) rotate(10deg) scale(1.2);

```

Transforms are hardware accelerated and performant for animations.

---

**🎞️ CSS Animations Using `@keyframes`**

CSS animations let you change CSS property values over time, allowing for smooth and engaging effects without JavaScript.

\* \*\*`@keyframes`\*\* define the stages of the animation by specifying CSS property values at various points (usually `from`/`to` or percentages).

Example: Fading in an element gradually

```css

@keyframes fadeIn {

from { opacity: 0; }

to { opacity: 1; }

}

.element {

animation-name: fadeIn;

animation-duration: 2s;

animation-timing-function: ease-in-out;

animation-iteration-count: infinite; /\* loops forever \*/

}

```

**Key animation properties:**

\* `animation-name` — links to the `@keyframes` block.

\* `animation-duration` — length of time for one cycle.

\* `animation-delay` — time to wait before starting.

\* `animation-iteration-count` — how many times the animation runs (`infinite` for looping).

\* `animation-timing-function` — controls pacing (`linear`, `ease`, `ease-in-out`, etc.).

\*\*Animations can target various properties:\*\* colors, position (`left`, `top`), size (`width`, `height`), transforms, opacity, shadows, and more.

✅ Basic Syntax of @keyframes

@keyframes animationName {

0% { /\* styles at start \*/ }

100% { /\* styles at end \*/ }

}

Then apply it to an element:

.element {

animation-name: animationName;

animation-duration: 2s;

animation-timing-function: ease-in-out;

animation-iteration-count: infinite;

}

🎯 Example 1: Fade In

HTML

<div class="fade-in">Hello World</div>

CSS

.fade-in {

opacity: 0;

animation: fadeIn 2s ease-in forwards;

}

@keyframes fadeIn {

to {

opacity: 1;

}

}

**Bounce Effect**

.bounce {

animation: bounce 1s infinite;

}

@keyframes bounce {

0%, 100% {

transform: translateY(0);

}

50% {

transform: translateY(-30px);

}

}