Properties of CSS

Name : Aaditya Bhosale

Cohort : Demis Hassabis

Roll No : 150096725029

1. field-sizing

- **Year of Standardization:** 2025 (Baseline Status)
- **Syntax:** field-sizing: fixed | content;
- **Purpose:** Allows form inputs (text areas, inputs) to automatically resize based on their content height/width, eliminating the need for JavaScript auto-grow scripts.

Working Example:
CSS
```
textarea {
  field-sizing: content;
  min-height: 40px;

}
```

- **Browser Support:** Supported in all major engines (Chrome, Edge, Firefox) by late 2025.
- **Reference:** MDN Web Docs - field-sizing

2. interpolate-size

- **Year of Introduction:** 2025
- **Syntax:** interpolate-size: allow-keywords | numeric-only;
- **Purpose:** Enables animations from 0 to auto (or min-content/fit-content). This solves the decade-old problem of animating height: auto for accordions.

Working Example:
CSS
```
:root {
  interpolate-size: allow-keywords;
```

```
}
.accordion-content {
  height: 0;
  overflow: hidden;
  transition: height 0.5s ease;
}
.accordion:hover .accordion-content {
  height: auto;
}
```

-
- **Browser Support:** Rapidly adopted in Chrome/Edge 130+, Firefox implementation in progress (2025).
- **Reference:** W3C CSS Values and Units Module Level 5

3. **text-box-trim (formerly leading-trim)**

- **Year of Standardization:** 2025
- **Syntax:** text-box-trim: none | trim-start | trim-end | trim-both;
- **Purpose:** Removes the annoying extra whitespace (leading) above and below text characters, allowing for perfect optical alignment of buttons and icons.

Working Example:

```
CSS
h1 {
  font-size: 3rem;
  line-height: 1.5;
  text-box-trim: trim-both;
   Removes the "invisible" vertical gap above the capital letters
  border: 1px solid red;
Border now hugs the text cap-height exactly
}
```

- **Browser Support:** Chrome 133+, Safari & Firefox Experimental (2025-2026).
- **Reference:** CSS Inline Layout Module Level 3

4. view-transition-name

- **Year of Widest Support:** 2024-2025
- **Syntax:** view-transition-name: <custom-ident> | none;
- **Purpose:** Assigns a unique identity to an element so the browser can morph it into a different element on a new page (Shared Element Transitions).

Working Example:
CSS
On the product listing page
```
.product-thumb {
  view-transition-name: product-image-123;
}
```

On the product detail page
```
.hero-image {
  view-transition-name: product-image-123;
}
```
The image will morph position and size automatically between pages

-
- **Browser Support:** Full Baseline Support (Chrome, Safari, Firefox) in 2025.
- **Reference:** MDN - View Transitions

5. anchor-name

- **Year of Introduction:** 2024 (Stable in 2025)
- **Syntax:** anchor-name: --<dashed-ident>;
- **Purpose:** Marks an element as a "reference anchor" so other absolute-positioned elements (like tooltips) can tether themselves to it without JavaScript.

Working Example:
CSS
```
.button {
  anchor-name: --my-tooltip-trigger;
}
```

```
.tooltip {
  position: absolute;
  position-anchor: --my-tooltip-trigger;
  top: anchor(bottom);  Sits right below the button
  left: anchor(center);
}
```

-
- **Browser Support:** Chrome/Edge 125+, Firefox Geckoview (2025).
- **Reference:** W3C CSS Anchor Positioning

6. position-area

- **Year of Introduction:** 2025
- **Syntax:** position-area: top | bottom | center | block-start ...
- **Purpose:** A shorthand for anchor positioning that uses a 3x3 grid concept to place popovers relative to their anchor. Simpler than using top: anchor(...).

Working Example:
CSS
```
.tooltip {
  position: absolute;
  position-anchor: --menu-btn;
  position-area: bottom center;
   Instantly centers the tooltip below the button
}
```

-
- **Browser Support:** Chrome 130+, Safari TP.
- **Reference:** MDN Web Docs - position-area

7. scroll-start

- **Year of Introduction:** 2025
- **Syntax:** scroll-start: <length> | <percentage> | auto;

- **Purpose:** Sets the initial scroll position of a container. Useful for starting a carousel in the middle or scrolling to the bottom of a chat window on load.

Working Example:
CSS
.chat-window {
  overflow-y: scroll;
  scroll-start: block-end;
  Automatically starts scrolled to the very bottom
}

-
- **Browser Support:** Chrome 130+, Firefox Nightly.
- **Reference:** W3C CSS Scroll Snap Module Level 2

8. animation-timeline

- **Year of Widest Support:** 2025
- **Syntax:** animation-timeline: scroll() | view() | <timeline-name>;
- **Purpose:** Links a CSS animation to the scroll position of a container rather than time.

Working Example:
CSS
.progress-bar {
  animation: fill-bar linear;
  animation-timeline: scroll(root);
  (Bar fills from 0% to 100% as you scroll down the page)

}

-
- **Browser Support:** Baseline 2025 (Chrome, Edge, Firefox, Safari).
- **Reference:** MDN - Scroll-driven Animations

9. timeline-scope

- **Year of Introduction:** 2024 (Widely supported 2025)
- **Syntax:** timeline-scope: --<dashed-ident>;
- **Purpose:** Allows a scroll timeline defined in one component to drive an animation in a completely different part of the DOM tree (e.g., sidebar scrolling controls header opacity).

Working Example:

```
CSS
body {
  timeline-scope: --sidebar-scroll;
}
.sidebar {
  scroll-timeline: --sidebar-scroll y;
}
.header-logo {
  animation: rotate-logo linear;
  animation-timeline: --sidebar-scroll;
}
```

-
- **Browser Support:** Chrome 116+, Safari 18+.
- **Reference:** W3C Scroll-driven Animations

10. reading-flow

- **Year of Introduction:** 2025 (Experimental)
- **Syntax:** reading-flow: flex-visual | flex-flow | grid-rows | grid-columns;
- **Purpose:** Fixes accessibility issues in Flexbox/Grid where visual order differs from DOM order. It tells screen readers to follow the *visual* layout.

Working Example:

```
CSS
.grid-layout {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  reading-flow: grid-rows;
```

(Screen reader now reads row by row, regardless of DOM order manipulation)
}

-
- **Browser Support:** Chrome Canary (2025), WebKit Draft.
- **Reference:** CSS Display Module Level 4

11. transition-behavior

- **Year of Standardization:** 2024/2025
- **Syntax:** transition-behavior: allow-discrete | normal;
- **Purpose:** Allows discrete properties like display or overlay to be part of a transition. Essential for fading out elements before setting them to display: none.

Working Example:

```
CSS
.modal {
  display: none;
  opacity: 0;
  transition: opacity 0.5s, display 0.5s;
  transition-behavior: allow-discrete;
}
.modal.open {
  display: block;
  opacity: 1;
}
```

-
- **Browser Support:** Baseline 2025.
- **Reference:** MDN - transition-behavior

12. overlay

- **Year of Introduction:** 2024 (Standardized 2025)
- **Syntax:** overlay: auto | none;
- **Purpose:** Used in transitions to keep an element in the "top layer" (like a popover) during its exit animation,

ensuring it doesn't get clipped by z-index during fade-out.

Working Example:

```
CSS
.popover {
  transition: overlay 0.5s allow-discrete, opacity 0.5s;
  opacity: 0;
  overlay: none;
}
.popover:popover-open {
  opacity: 1;
  overlay: auto;
}
```

-
- **Browser Support:** Chrome 117+, Firefox Nightly.
- **Reference:** W3C CSS Positioned Layout Level 4

13. margin-trim

- **Year of Introduction:** 2024/2025
- **Syntax:** margin-trim: none | block | block-start | block-end | inline...
- **Purpose:** Automatically removes margins from the first or last children in a container, replacing the need for :first-child { margin-top: 0 }.

Working Example:

```
CSS
.card {
  padding: 20px;
  gap: 10px;
  margin-trim: block;
  /* Removes margin-top from first child and margin-bottom from last child */
}
```

-

- **Browser Support:** Safari 16.4+, Chrome (Experimental 2025).
- **Reference:** W3C CSS Box Model Module Level 4

14. scrollbar-gutter

- **Year of Standardization:** 2024 (Widely adopted)
- **Syntax:** scrollbar-gutter: auto | stable | stable both-edges;
- **Purpose:** Reserves space for the scrollbar prevents "layout shift" when content grows and a scrollbar suddenly appears.

Working Example:

```
CSS
body {
  scrollbar-gutter: stable;
  /* The layout width is calculated as if the scrollbar is always there */
}
```

-
- **Browser Support:** Baseline 2024/2025.
- **Reference:** MDN - scrollbar-gutter

15. font-palette

- **Year of Widest Support:** 2024/2025
- **Syntax:** font-palette: normal | --<palette-name>;
- **Purpose:** Allows you to select and animate color palettes inside Color Fonts (like Emoji fonts or custom headers).

Working Example:

```
CSS
@font-palette-values --my-colors {
  font-family: "MyColorFont";
  override-colors: 0 red, 1 blue;
}
h1 {
  font-family: "MyColorFont";
```

```
  font-palette: --my-colors;
}
```

-
- **Browser Support:** Chrome 101+, Firefox 107+, Safari 15.4+ (Baseline).
- **Reference:** MDN - font-palette

16. text-wrap

- **Year of Standardization:** 2024 (New values)
- **Syntax:** text-wrap: wrap | nowrap | balance | pretty;
- **Purpose:** balance makes headlines look symmetric (no lone words). pretty prevents orphans at the end of paragraphs with better performance.

Working Example:

CSS

```
h2 {
  text-wrap: balance; /* Perfect specific for headlines */
}
p {
  text-wrap: pretty; /* Optimized for body text reading */
}
```

-
- **Browser Support:** Baseline 2024/2025.
- **Reference:** W3C CSS Text Module Level 4

17. initial-letter

- **Year of Standardization:** 2024/2025
- **Syntax:** initial-letter: <number> <integer>?;
- **Purpose:** Creates "Drop Caps" natively in CSS. The first number is the size (in lines), the second is the sink (how deep it sits).

Working Example:

CSS

```
p::first-letter {
```

```
  initial-letter: 3 2;
  /* 3 lines tall, sinks 2 lines down */
  color: gold;
  font-weight: bold;
}
```

-
- **Browser Support:** Chrome 110+, Safari 9+, Firefox (In development 2025).
- **Reference:** MDN - initial-letter

18. masonry-template-tracks (Grid Masonry)

- **Year of Introduction:** 2025 (Revised Spec)
- **Syntax:** grid-template-rows: masonry; (Syntax under debate, this is the leading candidate).
- **Purpose:** Enables Pinterest-style layouts natively in CSS Grid without JavaScript. Items pack vertically based on available space.

Working Example:

```
CSS
.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  grid-template-rows: masonry;
  /* Items automatically stack tightly */
}
```

-
- **Browser Support:** Firefox Nightly, Chrome Experimental (2025).
- **Reference:** W3C CSS Grid Layout Module Level 3

19. position-visibility

- **Year of Introduction:** 2025
- **Syntax:** position-visibility: always | anchors-visible | no-overflow;

- **Purpose:** Determines if an anchored element (like a tooltip) should remain visible if its anchor moves off-screen.

Working Example:
CSS
```
.tooltip {
  position-anchor: --btn;
  position-visibility: anchors-visible;
  /* Tooltip automatically hides if the button scrolls out of view
*/
}
```

-
- **Browser Support:** Chrome 129+, Experimental.
- **Reference:** W3C CSS Anchor Positioning

20. view-transition-class

- **Year of Introduction:** 2025
- **Syntax:** view-transition-class: <custom-ident>;
- **Purpose:** Allows you to group multiple view transition elements under one class to animate them together, rather than writing keyframes for every single unique view-transition-name.

Working Example:
CSS
```
.card {
  view-transition-class: product-card;
}
::view-transition-group(.product-card) {
  animation-duration: 0.5s;
}
```

-
- **Browser Support:** Chrome 125+, Baseline 2025.
- **Reference:** W3C CSS View Transitions Level 2