DOMS | Page No.
Date / /

# Css Properties

1. wha Css Property is

- A property is the thing you set on an element to control a visual or layout characteristics.
  color, margin, display
- A css rule has selector (s) + declaration block:

Selector { Property : value ; Property 2 : value 2; }

P { color : #333; font-size : 16 Px ; }

2. Categories of Css Properties

Grouping property by purpose helps remember them & reason about layout

- Box model & spacing

⇒ width, height, min-width, max-width, min-height, max-height

⇒ margin, padding, border, box-sizing

- Display & layout

⇒ display, position, top/right/bottom/left, float, clear

⇒ Modern layout flex & grid related: flex, flex direction, justify-content, align-items, grid-template-columns. etc.

DOMS | Page No.
Date / /

- Typography & Text

⇒ font-family, font-size, font-weight, line-height, letter-spacing, text-align, text-decoration, text-transfor

- Color & backgrounds

⇒ color, background, background-color, background-imag background size, background-position, background-repeat

- Visuals & decoration

⇒ border, border-radius, box-shadow, opacity, outline

- Transform & animation

⇒ transform, transition, animation

- Visibility & animation

⇒ visibility, overflow, z-index

- Other helpers

⇒ Cursor, pointer-events, user-select, resize

3. The Box Model

Every elements rectangular box = content + padding + border + margin

Content: where text image like (width/height) apply here unless box sizing changes it

Padding space inside the border; increases element visible area

border line around padding; border width counts in size. depending on box sizing

margin external spacing between elements (can collapse vertically in certain cases)

Important Property: box-sizing

- Content-box (default): width & height target Content area only
- border-box: width & height include padding + border This is often easier for layout

```
* { box-sizing : border-box ; }

.card {
    width : 300px ;
    padding : 20px ;
    border : 2px solid #ccc ;
    margin : 16 px ;
}
```

DOMS | Page No.
Date / /

4. Display & layout Properties

display

- Controls how element participates in layout

Main Values

block occupies full width, starts new line ( )

inline flows with text, width/height ignored ()

inline-block inline-level box but width/height

none removes element from rendering (not rendered, no space).

flex, grid makes element a flex/grid container for children

Positioning

Position: static | relative | absolute | fixed | sticky

Static normal flow (default)

relative normal flow but can be offset; offsets don't remove from flow.

fixed positioned relative to Viewport

absolute removed from flow : positioned relative to nearest positioned ancestor

Sticky behaves like relative until scrolling reaches a threshold, then acts fixed

```
.container { position: relative; }
.badge {
    position: absolute;
    top: 10px;
    right: 10px;
}
```

Flexbox basics (essential)

- Parent display: flex;
- Main properties on parent:
- flex direction (row | row reverse | column | column reverse)
- justify-content (controls main axis alignment)
- align-items (controls cross axis alignment)
- flex-wrap
- On children:
- flex shorthand for (flex-grow flex-shrink ; flex-basis)
- align-self;

```
.row {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
.items { flex: 1 1 200px; }
```

Grid basics (short)

- Parent: display: grid;
- Define tracks: grid-template-columns: 1fr 2fr;
- Place items with grid-column, grid-row or grid-area

5. Spacing: Margin vs Padding

- margin outside the border. Values: one to four values
  Shorthand: (margin: top right bottom left;)
- Padding inside the border. Same shorthand
- Margin collapse. vertical margins of adjacent blocks may collapse into a single margin (common confusion point)
- Use margin for spacing between elements, padding to give content breathing room inside the box

6. Border & background properties

- border : [width] [style] [color]; border : 2px solid #000;
- border-radius rounded corners (supports individual corners)
- background Shorthand :
  background : url('img.png') no-repeat center/cover;
  - background-size (cover / contain / explicit sizes)
  - background-repeat
  - background-position

```
.hero {
  background-image : url('hero.jpg');
  background-position : center;
  background-size : cover;
  background-repeat : no-repeat;
}
```

7. Typography & text properties

- font family fallback stack center: font-family : "Inter", system-ui, sans-serif;
- font-size can use px, rem, em, % - prefer rem for consistent scaling
- line-height vertical spacing between lines, can be unitless (recommended) 1.5

- font-weight : normal, bold numeric 400, 700 etc
- text-align, text-decoration, text-transform, letter-spacing, word spacing, white space

Accessibility tip

use font-size & line-height that scales well; avoid tiny px values ensures contrast see color + background contrast)

8. Color

- values hex #RRGGBB Shorthand #RGB ; rgb(), rgba() (alpha)

hsl(), hsla(), transparent

- opacity affects entire element & children rgba() affects only color value (no child impact).
- Css variables (--my-color) useful for theme Consistency

```
:root {
  --brand : #06b;
}
.button { background : var (--brand); }
```

DOMS | Page No.
Date / /

9. Visibility & overflow

- visibility : visible / hidden - hidden preserves layout, hidy but space remains.
- display : none - removes element completely
- overflow : visible / hidden / scroll / auto
  - overflow-x , overflow-y control axes individually
- z-index Controls stacking context (only applies to positioned elements or elements that creates stacking context).

10. Transforms , transitions , & animations

- transform : translateX (), rotate(), scale() GPU accelerated tranform
- transition : Property duration timing-function delay

  a { transition : color 0.2s ease ; }

@ Keyframes & animation for multi step animations

DOMS | Page No.
Date / /

11. Inheritence & specificity notes

- Inherited properties : typography-related properties (color, font-family, line-height) typically inherit.
- Non-herited layout properties (margin, padding, displa do not inherit
- use inherit, initial, unset keywords to control inheritence explicitly
- Specificity determines which rule applies : Inline style > IDs > Classes / attributes / pseudo-classes > elements / pseudo-elements
- !important overrides specificity but avoid using it except as last resort.

12 Units : when to use which

- Absolute : px fixed pixels (precise but not scalable)
- Relative
  - em relative to current element's font size
  - rem relative to root <html> font size (great for consisten
  - % often relative to parent dimension
  - Viewport vw, vh percentage of viewport width / height

- use rem for layout & typography baseline em inside components where relative scaling is desired

13 Common pitfalls & debugging tips

- forgetting box-sizing : border box leads to width confusion when using padding/border
- margin collapsing creates unexpected spacing - inspect using devtools.
- Position : absolute elements disappear if no positioned ancestor - ensure position : relative on container if needed
- overflow : hidden can clip child elements unintentionally (esp with shadows)
- low contrast text - fails accessibility use contrast checker

use devtools inspector to toggle rules & watch computed styles - often fastest fix

DOMS | Page No.
Date / /

14 Best practices & performance

- Prefer transform and opacity for animations (GPU-accelerated) instead of animating top/left (layout thrash)
- Keep CSS modular: component classes rather than over-specific selectors.
- Use shorthand properties to reduce files size, but be explicit when you need to override only one side use margin-top
- use CSS variable for themeability & fewer repeated values
- Minify CSS for production; avoid heavy selectors like body div ul li.class they make slower & are fragile

15 Handy Shorthand examples

Margin / Padding margin: 12px 8px 12px 8px;

Border Shorthand border: 1px solid #ddd;

Background Shorthand background: #fff url('pat.png') repeat-x right top;

font Shorthand font: italic small-caps 16px/1.6 right top, "Georgia", serif;

Be careful - font: shorthand resets properties you don't include

16. Quick refrence : Common properties & example usage

- box { width : 320 px ; height : 200px ; box-sizing : border-box; }
- section { padding : 24px 16px ; margin-bottom : 32px ; }
- card { border : 1 px solid #e0e0e0 ; border-radius : 8px ; box-shadow : 0 2px 8px rgba (0, 0, 0, 0.08); }
- h1 { font-size : 2 rem ; line-height : 1.2 ; font-weight : 700 ; margin : 0 0 1rem 0; }
- header { display : flex ; justify-content : space-between ; align-items : center; }
- grid { display : grid ; grid-template-columns : repeat (3, 1fr) ; gap : 16px ; }
- hero { background : linear-gradient (120 deg, #f6f8ff, #e7f3ff); }
- btn { transition : transform .15s ease, box-shadow .15s ease; }
- btn :hover { transform : translate Y (-2px) ; box-shadow : 0 6px 18px rgba (0, 0, 0, 0.12); }

## 17 Exercises (practice - do these)

1. Create a 3-column responsive card grid using CSS grid that becomes 1 column below 600 px.
2. Build a header with "logo" on left & "nav" on right using flexbox; vertically center items.
3. Make a button that scales up slightly on hover using transform & transition (hint use transform: translateZ(0) will-change if you see jitter)
4. Exp. change box-sizing to content-box and add 20 px padding + 2px border to a 300 px width element - what happens? Now switch to border-box Compare

## 18 Mini Cheat Sheet

1. box-sizing : border-box Easier sizing
2. display : flex one dimensional layout row/column
3. display: grid 2D layout.
4. position : relative anchor for absolutely positioned children
5. Overflow: auto adds scrollbars only when needed
6. visibility: hidden Vs display: none hidden keeps space, none removes it.

7. opacity affect children ; rgba() does not

8. rem for root relative sizing; em for component - relative

19 Extra tips for writing maintainable Css

- use meaningful class names (BEM or utility classes)
- Group related properties (layout → box → typography → visual) in your Css for readability
- Comment complex rules & keep a style guide (font scale, spacing scale)
- Prefer Semantic HTML + minimal classes over complex Css targeting (easier to reason about)

20 final Recap

Understanding Css properties & how they interact (box model, display, positioning) is the foundation of building reliable, maintainable UIs. Once you master these categories more, advanced techniques (responsive design, animations, component libraries) become simple combinations of these primitives.