CSS 3 Quick Reference Cheatsheet

A comprehensive cheat sheet covering essential CSS3 selectors, properties, values, units, layout techniques, and best practices for modern web design.

Selectors & Basic Styling

Basic Selectors

`*`	Selects all elements.
`element`	Selects elements of a specific type (e.g., `p`).
`.class`	Selects elements with a specific class (e.g., `.button`).
`#id`	Selects the element with a specific ID (e.g., `#header`).
`[attribute]`	Selects elements with a specific attribute (e.g., `[type="text"]`).
`[attribute="value"]`	Selects elements with a specific attribute and value.
`[attribute~="value"]`	Selects elements whose attribute value contains a specified word.
`[attribute|="value"]`	Selects elements whose attribute value starts with the specified word followed by a hyphen.
`[attribute^="value"]`	Selects elements whose attribute value begins with a specified value.
`[attribute$="value"]`	Selects elements whose attribute value ends with a specified value.
`[attribute*="value"]`	Selects elements whose attribute value contains a specified value.

Combinators

`A B`	Descendant Selector: Selects B inside A.
`A > B`	Child Selector: Selects direct children B of A.
`A + B`	Adjacent Sibling Selector: Selects B immediately following A.
`A ~ B`	General Sibling Selector: Selects B anywhere after A within the same parent.

```
div p { color: blue; }
/* Selects all p
elements inside div */
```

```
div > p { color: red; }
/* Selects p elements that are
direct children of div */
```

```
h1 + p { margin-top:
20px; }
/* Selects the first p
element immediately
after an h1 */
```

```
h2 ~ p { font-style: italic; }
/* Selects all p elements that are
siblings of h2 and appear after it
*/
```

Pseudo-Classes & Pseudo-Elements

`:hover`	Selects an element when you mouse over it.
`:active`	Selects the active link/element.
`:focus`	Selects an element that has focus.
`:visited`	Selects links that have been visited.
`:link`	Selects unvisited links.
`:first-child`	Selects the first child of its parent.
`:last-child`	Selects the last child of its parent.
`:nth-child(n)`	Selects the n-th child of its parent (e.g., `odd`, `even`, `3`, `2n+1`).
`:not(selector)`	Selects elements that do NOT match the selector.
`:checked`	Selects checked input elements (radio or checkbox).
`::before`	Inserts content before an element.
`::after`	Inserts content after an element.
`::first-line`	Selects the first line of a block-level element.
`::first-letter`	Selects the first letter of a block-level element.

Box Model (Block)

`display`	Sets how an element is displayed (`block`, `inline`, `inline-block`, `none`, `flex`, `grid`).
`width`, `height`	Sets the width and height of an element.
`margin`	Shorthand for `margin-top`, `margin-right`, `margin-bottom`, `margin-left`. Creates space *outside* the border.
`padding`	Shorthand for `padding-top`, `padding-right`, `padding-bottom`, `padding-left`. Creates space *inside* the border.
`border`	Shorthand for `border-width`, `border-style`, `border-color`. Defines border around padding and content.
`box-sizing`	Defines how the total width and height of an element are calculated (`content-box` (default), `border-box`). Use `border-box` for easier layout.

```
.box {
  width: 100px;
  height: 100px;
  padding: 10px;
  border: 1px
solid black;
  margin: 20px;
  box-sizing:
border-box;
}
/* Total width:
100px
(content+padding
+border) */
```

```
.box-old {
  width: 100px;
  height: 100px;
  padding: 10px;
  border: 1px solid black;
  margin: 20px;
  box-sizing: content-box;
}
/* Total width: 100+10+10+1+1 = 122px */
```

Colors & Backgrounds

`color`	Sets the text color.
`background-color`	Sets the background color of an element.
`background-image`	Sets the background image (`url(...)`).
`background-repeat`	Sets how a background image repeats (`repeat` , `no-repeat` , `repeat-x` , `repeat-y`).
`background-position`	Sets the starting position of a background image (`center` , `top left` , `50% 50%` , `10px 20px`).
`background-size`	Sets the size of the background image (`auto` , `cover` , `contain` , `50%` , `200px`).
`background-attachment`	Sets whether a background image scrolls with the page or is fixed (`scroll` , `fixed` , `local`).
`background`	Shorthand for all background properties.
Color Formats:	`#RRGGBB` , `#RGB` , `rgb(R,G,B)` , `rgba(R,G,B,A)` , `hsl(H,S,L)` , `hsla(H,S,L,A)` , predefined names (e.g., `red`), `currentColor` , `transparent` .

Layout & Positioning

Display & Positioning

`display: block;`	Starts on a new line, takes full width.
`display: inline;`	Does not start on new line, takes width based on content, cannot set `width` / `height` .
`display: inline-block;`	Does not start on new line, takes width based on content, CAN set `width` / `height` .
`display: none;`	Element is completely removed from the layout.
`position: static;`	Default positioning; element is positioned according to normal flow. `top` , `left` , etc. have no effect.
`position: relative;`	Positioned relative to its normal position. `top` , `left` , etc. offset it from its normal position.
`position: absolute;`	Positioned relative to its nearest positioned ancestor (instead of the viewport). Removed from normal flow.
`position: fixed;`	Positioned relative to the viewport. Stays in place even when scrolling. Removed from normal flow.
`position: sticky;`	Positioned based on the user's scroll position. Behaves like `relative` until a threshold is met, then like `fixed` .
`top` , `bottom` , `left` , `right`	Used with `position: relative` , `absolute` , `fixed` , or `sticky` to specify offsets.
`z-index`	Specifies the stack order of a positioned element (higher number is higher in stack).

Flexbox (Container)

`display: flex;`	Makes the element a flex container, its direct children become flex items.
`flex-direction`	Defines the direction of flex items (`row` (default), `row-reverse` , `column` , `column-reverse`).
`flex-wrap`	Specifies if flex items should wrap (`nowrap` (default), `wrap` , `wrap-reverse`).
`justify-content`	Aligns items along the **main axis** (`flex-start` , `flex-end` , `center` , `space-between` , `space-around` , `space-evenly`).
`align-items`	Aligns items along the **cross axis** (`stretch` (default), `flex-start` , `flex-end` , `center` , `baseline`).
`align-content`	Aligns flex lines when there's extra space in cross axis (only if `flex-wrap: wrap`). Values similar to `justify-content` .
`gap` , `row-gap` , `column-gap`	Sets the gaps (gutters) between rows and columns in flex and grid layouts.
Tips:	Flexbox is great for 1D layouts (row or column). Use for navbars, centering items, distributing space.

Flexbox (Items)

`order`	Specifies the order of a flex item (`integer` , default is 0).
`flex-grow`	Specifies how much a flex item will grow relative to the rest (`number` , default 0).
`flex-shrink`	Specifies how much a flex item will shrink relative to the rest (`number` , default 1).
`flex-basis`	Specifies the initial size of a flex item before flexing (`length` or `auto` , default `auto`).
`flex`	Shorthand for `flex-grow` , `flex-shrink` , `flex-basis` (e.g., `flex: 1 1 auto;`). Common values: `flex: 1;` (grow/shrink with basis auto), `flex: auto;` (grow/shrink with basis auto), `flex: none;` (0 0 auto), `flex: 0 0 auto;` (no grow/shrink, basis auto).
`align-self`	Aligns the individual flex item along the cross axis (overrides `align-items` on the container). Values similar to `align-items` .

CSS Grid (Container)

`display: grid;`	Makes the element a grid container, its direct children become grid items.
`grid-template-columns`	Defines the columns in the grid (`px` , `%` , `em` , `rem` , `fr` , `auto` , `repeat()` , `minmax()`). Example: `grid-template-columns: 1fr 2fr 1fr;`
`grid-template-rows`	Defines the rows in the grid. Example: `grid-template-rows: auto 100px;`
`grid-template-areas`	Defines grid layout using named areas. Example: `grid-template-areas: "header header" "sidebar main";`
`gap` , `row-gap` , `column-gap`	Sets the gaps between grid rows and columns.
`justify-items`	Aligns items inside grid cells along the **inline** (row) axis (`start` , `end` , `center` , `stretch`).
`align-items`	Aligns items inside grid cells along the **block** (column) axis (`start` , `end` , `center` , `stretch`).
`justify-content`	Aligns the grid itself along the **inline** axis inside the container (`start` , `end` , `center` , `space-between` , `space-around` , `space-evenly`).
`align-content`	Aligns the grid itself along the **block** axis inside the container (`start` , `end` , `center` , `space-between` , `space-around` , `space-evenly`).
`grid-auto-columns`	Specifies the size of implicitly created columns.
`grid-auto-rows`	Specifies the size of implicitly created rows.
Tips:	Grid is powerful for 2D layouts (rows and columns simultaneously). Use for page layouts, complex components.

CSS Grid (Items)

`grid-area`	Assigns an item to a named grid area, or specifies its position using row/column start/end lines.
`grid-column-start` , `grid-column-end`	Specifies which column lines the item starts and ends at (`line-number` , `span number` , `line-name`).
`grid-row-start` , `grid-row-end`	Specifies which row lines the item starts and ends at (`line-number` , `span number` , `line-name`).
`grid-column`	Shorthand for `grid-column-start` and `grid-column-end` (e.g., `grid-column: 1 / 3;` or `grid-column: span 2;`).
`grid-row`	Shorthand for `grid-row-start` and `grid-row-end` .
`justify-self`	Aligns the individual grid item inside its cell along the **inline** axis (overrides `justify-items` on the container). Values similar to `justify-items` .
`align-self`	Aligns the individual grid item inside its cell along the **block** axis (overrides `align-items` on the container). Values similar to `align-items` .

Floats & Clearing

`float: left;`	Element floats to the left within its container. Text and inline elements wrap around it.
`float: right;`	Element floats to the right within its container.
`float: none;`	Element does not float (default).
`clear: left;`	Element moves below any floated elements on its left side.
`clear: right;`	Element moves below any floated elements on its right side.
`clear: both;`	Element moves below any floated elements on both sides.
The “clearfix” hack:	`.clearfix::after { content: ""; display: table; /* Use table to contain floats */ clear: both; }`

Typography & Advanced Styling

Fonts

`font-family`	Specifies the font(s) to use (comma-separated, with generic fallback).
`font-size`	Sets the size of the text (`px` , `em` , `rem` , `%` , `vw` , `vh`). `rem` is recommended for accessibility.
`font-weight`	Sets how thick or thin characters are (`normal` , `bold` , `lighter` , `bolder` , `100` to `900`).
`font-style`	Sets the style of the font (`normal` , `italic` , `oblique`).
`font-variant`	Sets whether or not text should be displayed in a small-caps font (`normal` , `small-caps`).
`line-height`	Sets the spacing between lines (`number` (multiplier), `length` , `%`). Unitless number is best practice.
`font`	Shorthand for `font-style` , `font-variant` , `font-weight` , `font-size` / `line-height` , and `font-family` .
`@font-face`	Rule for defining custom fonts to load.

Text Styling & Effects

`text-align`	Aligns the text inside its container (`left` , `right` , `center` , `justify`).
`text-decoration`	Adds decoration lines (`none` , `underline` , `overline` , `line-through`). Can also set color and style.
`text-transform`	Controls the capitalization of text (`none` , `uppercase` , `lowercase` , `capitalize`).
`text-indent`	Specifies the indentation of the first line of text.
`letter-spacing`	Increases or decreases the space between characters.
`word-spacing`	Increases or decreases the space between words.
`white-space`	Controls how whitespace is handled (`normal` , `nowrap` , `pre` , `pre-wrap` , `pre-line`).
`text-shadow`	Adds shadow to text (`h-shadow v-shadow blur-radius color`).

```
h1 { text-shadow: 2px 2px 5px blue; }
```

```
.truncate {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
```

Lists & Tables

`list-style-type`	Sets the type of list item marker (`disc` , `circle` , `square` , `decimal` , `none` , etc.).
`list-style-image`	Sets an image as the list item marker (`url(...)`).
`list-style-position`	Specifies the position of the list item marker (`inside` , `outside`).
`list-style`	Shorthand for all list style properties.
`border-collapse`	Specifies if table borders should be collapsed into a single border (`separate` , `collapse`).
`border-spacing`	Specifies the distance between the borders of adjacent cells (when `border-collapse: separate`).
`caption-side`	Specifies the position of the table caption (`top` , `bottom`).
`empty-cells`	Shows or hides borders and background on empty cells in a table (`show` , `hide`).

`ul { list-style: none; padding: 0; }`
`table { border-collapse: collapse; }`
`td, th { border: 1px solid #ddd; padding: 8px; }`
	Common reset for lists and basic table styling.

Units

Absolute Units (Physical Size)	`px` (pixels - relative to viewing device), `pt` (points - 1/72 of an inch), `pc` (picas - 12 points), `in` (inches), `cm` (centimeters), `mm` (millimeters).
Relative Units (Relative to Parent/Viewport/Root)	`%` (relative to parent element), `em` (relative to parent's font-size), `rem` (relative to root (`<html>`) font-size).
Viewport Units (Relative to Viewport Size)	`vw` (1% of viewport width), `vh` (1% of viewport height), `vmin` (1% of viewport's smaller dimension), `vmax` (1% of viewport's larger dimension).
Flex/Grid Units	`fr` (fraction of the available space in a grid container). `auto` (takes up remaining space or size of content).
Function Units	`calc()` (perform calculations), `min()` , `max()` , `clamp()` (constrain a value between two others).

```
.element {
  width: 50%; /* 50% of parent width */
  font-size: 1.2em; /* 1.2 times parent font-size */
  padding: 1rem; /* 1 time root font-size */
  height: 50vh; /* 50% of viewport height */
}

.grid-container {
  grid-template-columns: 1fr auto 1fr;
}
```

```
.responsive-text {
  font-size: clamp(1rem, 2.5vw, 2rem);
}
/* Font size will be between 1rem and 2rem, scaling with viewport width (2.5vw) */
```

Best Practice:	Use `rem` for font sizes (relative to root) and `em` for components (relative to parent component) for better scaling. Use `%` , `vw` / `vh` , `fr` for responsive layouts.

Custom Properties (Variables)

`--*name*: value;`

Define a custom property (variable) in a selector.

`var(--*name*);` or `var(--*name*, fallback-value);`

Use a custom property.

```
:root { /* Define globally or in a specific scope */
  --main-color: #06c;
  --padding: 10px 15px;
}

h1 {
  color: var(--main-color);
}

.button {
  padding: var(--padding);
  background-color: var(--main-color);
  border: 1px solid var(--main-color);
}

.element {
  color: var(--secondary-color, red); /* Use fallback if --
secondary-color is not defined */
}
```

Tips:	Makes CSS more maintainable. Useful for themes, consistent spacing, colors, etc. Can be updated dynamically with JavaScript.

Responsive Design & Interaction

Media Queries

`@media only screen and (max-width: 600px) { /* CSS rules */ }`

Apply styles only when the screen width is 600px or less.

`@media print { /* CSS rules */ }`

Apply styles only when printing the page.

Common Media Features:

- `width`, `min-width`, `max-width`
- `height`, `min-height`, `max-height`
- `orientation` (`portrait`, `landscape`)
- `resolution`

Logical Operators:

- `and`: Combines media features (`only screen and (min-width: 768px) and (max-width: 1024px)`)
- `,` (comma): Acts as OR (`screen and (max-width: 600px), print`)
- `not`: Negates a media query (not print)

```css
/* Mobile First Approach */
.container { width: 100%; }

@media (min-width: 768px) {
  /* Tablet styles */
  .container { width: 750px; margin: 0 auto; }
}

@media (min-width: 1200px) {
  /* Desktop styles */
  .container { width: 1170px; }
}
```

Tips:

- Use a mobile-first approach (design for small screens first, then use `min-width` media queries).
- Use relative units (`%`, `em`, `rem`, `vw` / `vh`) alongside media queries.

Transitions

`transition-property`	Specifies the CSS property to transition (`none`, `all`, or specific property name like `opacity`, `transform`, `color`).
`transition-duration`	Specifies how long the transition will take (`time` in seconds `s` or milliseconds `ms`).
`transition-timing-function`	Specifies the speed curve of the transition (`ease` (default), `linear`, `ease-in`, `ease-out`, `ease-in-out`, `cubic-bezier(n,n,n,n)`).
`transition-delay`	Specifies a delay before the transition starts (`time`).
`transition`	Shorthand for all transition properties (e.g., `transition: property duration timing-function delay;`).

```css
.button {
  background-color: blue;
  transition: background-color 0.3s ease;
}

.button:hover {
  background-color: darkblue;
}
```

```css
.box {
  width: 100px;
  transition: width 0.5s ease-in-out 0.1s; /* transition width over 0.5s, 0.1s delay */
}

.box:hover {
  width: 200px;
}
```

Tips: Transition properties that browsers can animate efficiently (`opacity`, `transform`) for better performance.

Transformations (2D)

`transform: translate(x, y);`	Moves an element from its current position.
`transform: translateX(x);`	Moves an element horizontally.
`transform: translateY(y);`	Moves an element vertically.
`transform: rotate(angle);`	Rotates an element (`angle` in `deg` , `grad` , `rad` , `turn`).
`transform: scale(x, y);`	Changes the size of an element (1 is original size).
`transform: scaleX(x);`	Changes the width of an element.
`transform: scaleY(y);`	Changes the height of an element.
`transform: skew(x-angle, y-angle);`	Skews an element along the X and Y axes.
`transform-origin`	Specifies the point around which a transform is performed (default is `center center`).

Other Useful Properties

`opacity`	Sets the transparency level of an element (0 is fully transparent, 1 is fully opaque).
`box-shadow`	Adds shadow to an element's box (`h-shadow v-shadow blur-radius spread-radius color inset`).
`border-radius`	Adds rounded corners to an element.
`outline`	Draws an outline around an element (outside the border). Use `outline: none;` with caution for accessibility.
`cursor`	Specifies the mouse cursor to be displayed (`pointer` , `grab` , `text` , `not-allowed` , `default` , etc.).
`visibility`	Hides or shows an element without affecting layout (`visible` , `hidden` , `collapse`).
`overflow`	Specifies how to handle content that overflows an element's box (`visible` , `hidden` , `scroll` , `auto`).
`pointer-events`	Controls whether an element can be the target of mouse events (`auto` , `none`).