CSS3 Cheatsheet

Selectors

- Basic Selectors:
  - `*`: Universal selector matches all elements.
  - `element`: Tag selector (e.g., `p`, `div`).
  - `.class`: Class selector.
  - `#id`: ID selector.
- Combinators:
  - `A B`: Descendant selector (all B under A).
  - `A > B`: Child selector (direct B under A).
  - `A + B`: Adjacent sibling selector (B immediately follows A).
  - `A ~ B`: General sibling selector (all B following A).
- Pseudo-classes:
  - `:hover`, `:focus`, `:active`: User interaction.
  - `:nth-child(n)`, `:first-child`, `:last-child`: Position-based.
- Pseudo-elements:
  - `::before`, `::after`: Add content before/after element's content.
  - `::placeholder`: Placeholder text of an input field.

Box Model

- `width`, `height`: Width/height of the content area.
- `padding`: Space between content and border.
- `border`: Border around padding. (e.g., `border: 1px solid black;`)
- `margin`: Space outside the border.
- `box-sizing`: How box sizes are calculated.
  - `content-box` (default): `width` and `height` include only content area.
  - `border-box`: `width` and `height` include `padding` and `border`.

Layout

Display

- `display: block;`: Block-level element. Takes 100% width.
- `display: inline;`: Inline element. Takes only as much width as content.
- `display: inline-block;`: Placed like `inline` but allows `width`, `height` like `block`.
- `display: none;`: Completely hides element from the screen.

Position

- `position: static;` (default)
- `position: relative;`: Moved via `top`, `right`, `bottom`, `left` relative to its original position.
- `position: absolute;`: Positioned relative to the nearest non-static ancestor.
- `position: fixed;`: Positioned relative to the viewport. Fixed even when scrolling.
- `position: sticky;`: Toggles between `relative` and `fixed` based on scroll position.

Flexbox

Powerful model for 1D layout.

- Container Properties (`display: flex;`):
  - `flex-direction`: Set main axis direction (`row`, `column`).
  - `justify-content`: Main axis alignment.
  - `align-items`: Cross axis alignment.
  - `flex-wrap`: Line wrapping.
  - `gap`: Space between items.
- Item Properties:
  - `flex-grow`: Growth ratio.
  - `flex-shrink`: Shrink ratio.
  - `flex-basis`: Default size.
  - `order`: Placement order.

Grid

Model for 2D layout.

- Container Properties (`display: grid;`):
  - `grid-template-columns`, `grid-template-rows`: Define grid track sizes.
  - `grid-gap` or `gap`: Space between grid cells.
  - `justify-items`, `align-items`: Item alignment inside cells.
- Item Properties:
  - `grid-column`, `grid-row`: Place items using grid lines.

Typography

- `font-family`: Specify font.
- `font-size`: Font size.
- `font-weight`: Font weight (`normal`, `bold`, `100-900`).
- `color`: Font color.
- `text-align`: Text alignment (`left`, `center`, `right`).
- `line-height`: Line height.
- `letter-spacing`: Spacing between letters.
- `text-decoration`: `none`, `underline`, `line-through`.

Transitions and Animations

Transition

Smoothly express changes in property values. `transition: property duration timing-function delay; transition: background-color 0.5s ease-in-out;`

Animation

Define animation sequences with `@keyframes`.

```
@keyframes slide-in {
  from { transform: translateX(-100%); }
  to { transform: translateX(0); }
}

.element {
  animation: slide-in 1s ease-out forwards;
}
```

Media Queries

Key to responsive design. Applies different styles based on conditions (e.g., screen size).

```
/* When screen width is 600px or less */
@media (max-width: 600px) {
  .container {
    flex-direction: column;
  }
}
```

Units

- Absolute Units: `px`, `pt`
- Relative Units:
  - `%`: Percentage relative to the parent element.
  - `em`: Multiple of parent's `font-size`.
  - `rem`: Multiple of root element's (`html`) `font-size`.
  - `vw`, `vh`: Percentage relative to the viewport width/height.

Google Style Guide

Common Principles

- Protocol: Use `HTTPS` for all resources.
- Indentation: 2 spaces (no tabs).
- Capitalization: Use lowercase only for all code.
- Encoding: Use `UTF-8` (no BOM). Specify `<meta charset="utf-8">` in HTML.

HTML Guide

- Doctype: Always use `<!doctype html>`.
- Validity: Write valid HTML.
- Semantics: Use tags according to purpose (e.g., `<a>`, `<p>`, `<header>` instead of `<div>`).
- Multimedia: Alternative text via `alt` attribute is mandatory.
- Optional Tags: Optional tags (html, body, li, etc.) can be omitted for optimization.
- Quotation: Use double quotes (") for attribute values.

CSS Guide

- Class Naming: Use semantic or functional names. Separate with hyphens (`-`).
- ID Selectors: Avoid ID selectors; use Class selectors instead.
- Shorthand: Use shorthand properties (`font`, `margin`, etc.) whenever possible.
- 0 & Units: Omit units if value is 0.
- Hex Notation: Use 3-character hex codes where possible.
- Important: Refrain from using `!important`.

Formatting and Conventions

- Order: Alphabetical sorting for CSS declarations is recommended.
- Spacing: 1 space before opening brace, 1 space after property colon (`:`).
- Comments: Partition complex code with comments.
- Separation: Strictly separate structure (HTML), presentation (CSS), and behavior (JS).