CSS

Style, spacing, and everyday layout.

01 Rules & declarations

```
.card {
  color: #18302b;
  background: white;
  padding: 1rem;
}
```

A selector chooses elements. Declarations pair properties with values. Link a .css file from the HTML head using rel="stylesheet".

02 Everyday selectors

`p / .card`	Element / class
`#intro`	Element with this id
`.card p`	Descendant paragraphs
`.card > p`	Direct child paragraphs
`h2 + p`	Next sibling paragraph
`[type="email"]`	Matching attribute

Separate selectors with commas to share a rule. Prefer reusable classes for component styling.

03 States & pseudo-elements

```
a:hover { text-decoration: underline; }
button:focus-visible {
  outline: 2px solid currentColor;
  outline-offset: 3px;
}
li:first-child { font-weight: 700; }
p::first-letter { font-size: 1.5em; }
```

: selects a state or condition; :: targets a pseudo-element. Keep keyboard focus visible.

04 Cascade & inheritance

```
body { color: #222; }
p { color: navy; }
.note { color: teal; }
.note { color: purple; }
```

Within the same origin, importance, and layer, specificity compares IDs, classes/attributes/pseudo-classes, then types. Later rules break ties here. Text color inherits; margins do not.

05 Box model & sizing

```
*, *::before, *::after {
  box-sizing: border-box;
}
.card {
  width: 20rem;
  max-width: 100%;
  padding: 1rem;
  border: 1px solid #ccc;
}
```

border-box includes padding and border in the declared width and height. Margin sits outside.

06 Spacing & centering

```
.page {
  width: min(100% - 2rem, 70rem);
  margin-inline: auto;
  padding-block: 2rem;
}
.card { margin: 1rem 2rem; }
```

Two shorthand values mean vertical, horizontal; four mean top, right, bottom, left. Some vertical block margins collapse.

07 Units & useful functions

`px`	CSS pixel
`rem`	Root font size
`em`	Contextual font size
`%`	Relative to the property
`vw / vh`	1% of viewport dimensions
`clamp(a,b,c)`	Min, preferred, max

calc() combines units; space + and - operators. On font-size, em uses the parent font size.

08 Typography

```
body {
  font-family: system-ui, sans-serif;
  font-size: 1rem; line-height: 1.5;
}
h1 { font-size: clamp(2rem, 5vw, 3rem); }
p { max-width: 65ch; }
```

Unitless line-height scales with font size. font-weight sets weight; text-align aligns inline content.

09 Color, borders & backgrounds

```
.card {
  color: #18302b;
  background: rgb(240 248 244);
  border: 1px solid #cbd5d1;
  border-radius: 0.75rem;
  box-shadow: 0 2px 8px #0002;
}
```

Use named, hex, rgb(), or hsl() colors. currentColor uses text color. Check readable contrast.

CSS

Build flexible interfaces.

10 Display & overflow

```
.row { display: flex; }
.badge { display: inline-block; }
.hidden { display: none; }
.panel {
  max-height: 20rem;
  overflow: auto;
}
```

display: none removes layout space; visibility: hidden keeps it. overflow: auto adds scrolling when needed.

11 Flexbox: one-dimensional layout

```
.toolbar {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
  justify-content: space-between;
  align-items: center;
}
```

justify-content aligns on the main axis; align-items on the cross axis. flex-direction sets the main axis (row by default).

12 Flexible children

```
.sidebar { flex: 0 0 14rem; }
.content {
  flex: 1 1 0;
  min-width: 0;
}
.actions { margin-left: auto; }
```

flex is grow, shrink, basis. min-width: 0 lets a flex child shrink below its content's automatic minimum. Use gap on the parent for spacing.

13 Grid: rows & columns

```
.grid {
  display: grid;
  grid-template-columns:
    repeat(3, minmax(0, 1fr));
  gap: 1rem;
}
.wide { grid-column: 1 / -1; }
```

fr shares leftover space. minmax(0, 1fr) lets tracks shrink. grid-column and grid-row place direct children on grid lines.

14 Responsive layouts

```
.grid { grid-template-columns: 1fr; }
@media (min-width: 48rem) {
  .grid {
    grid-template-columns: 1fr 1fr;
  }
}
```

With display: grid, this starts with one column and adds a second on wider viewports. Choose breakpoints where the content needs them.

15 Positioning & stacking

```
.card { position: relative; }
.badge {
  position: absolute;
  top: 0.5rem; right: 0.5rem;
}
.nav { position: sticky; top: 0; }
```

absolute leaves normal flow and uses a containing block (here .card). sticky needs an inset and room to move. z-index applies within stacking contexts.

16 Images & aspect ratios

```
.cover {
  width: 100%;
  aspect-ratio: 16 / 9;
  object-fit: cover;
  display: block;
}
```

cover fills the box by cropping; contain fits the whole image. For uncropped responsive images, use max-width: 100% and height: auto.

17 Custom properties

```
:root { --accent: #087b55; --space: 1rem; }
.card {
  padding: var(--space);
  color: var(--accent, teal);
}
```

Custom properties follow the cascade and usually inherit. var() reads a value with an optional fallback; override variables on a component.

18 Transitions & reduced motion

```
.button {
  transition: transform 150ms ease;
}
.button:hover { transform: scale(1.03); }
@media (prefers-reduced-motion: reduce) {
  .button { transition: none; }
}
```

A transition animates a property change. Respect reduced-motion preferences and keep essential feedback available without animation.