Best ways to remember these practices: Use checklists while coding, practice with purpose,

use Mnemonics or Acronyms**: C.R.O.W.N.F.L.U.V.V.**

C	COMMENTS
R	RESPONSIVE DESIGN
O	ORGAINIZED STYLES
W	WIREFRAME TAGS
N	NAMING CONVENTIONS
F	FLEXBOX/GRID
L	LIQUID LAYOUT
U	UNUSED STYLES REMOVED
V	VALIDATION
V	VARIABLES

<continue to next page>

Adhering vs Abandoning: The Best CSS Practices

CSS Practice	Advantages of Applying the Practice	Consequences of Skipping the Practice
External Stylesheets	Reusability, Cleaner Code by reducing clutter in HTML files, and Faster Load times	Messy HTML, Redundant code, Harder updates
Consistent Naming Conventions	Improved Readability, scalability, reduced errors, and enhanced collaboration (team members can understand and work with the code easily)	Confusing code, style conflicts, harder debugging, and reduced reusability.
Use CSS Variables	Consistency across styles, easy global updates, improved maintainability, and dynamic styling with JavaScript.	Repetitive code, difficult maintenance, limited flexibility, and performance concerns.
Keep your CSS D.R.Y. (Don't Repeat Yourself)	Maintainability: you only need to update styles in one place, Consistency: shared styles ensure a uniform look, avoiding accidental design mismatches, cleaner code, Performance: smaller, optimized stylesheets load faster and are easier for browsers to process.	Code Duplication: repeating styles increases file size and makes CSS harder to manage. Inconsistent Design: small changes may be missed in some areas, Harder debugging, and reduced flexibility.
Optimize for Performance	Faster load times, improved user experience, and reduced server load.	Slow page load, CSS conflicts: style overrides and bugs, mobile responsiveness issues, and poor accessibility.
Use of Flexbox & Grid for Layouts	Layout control, responsive design, alignment options, and cleaner code.	Poor responsiveness, layout challenges, time drain: needs more effort for debugging and inconsistent design.
Implementing Responsive Design	Device flexibility, improved user experience, SEO boost, saves time and resource.	Poor mobile experience, higher bounce rates, and accessibility barriers
Avoid Overusing: !important	Cleaner cascading logic: lets CSS do its job naturally, Easier maintenance, better collaboration among team members, and improved debugging.	Broken inheritance and cascade: overrides natural CSS behavior, Styling conflicts: difficult to make small adjustments, unscalable codebase, accessibility issues.
Organize Styles Logically	Improved readability with clear structure, efficient maintenance making changes easily, and better tool support.	Hard to debug with styles overwriting each other, and time drain: you end up re-writing instead of refining.
Validating CSS Code	Error Detection, cross-browser compatibility, and improved performance.	Hidden bugs: invalid declarations may be silently ignored leading to broken designs. Browser inconsistencies, and harder debugging.