General Good Frontend Coding Practices

HTML

- Always use <!DOCTYPE html> to prevent Quirks mode
- One <h1> per page; use semantic elements (<header>, <main>, <footer>, <nav>, etc.) instead of generic divs
- Always add alt attributes to images; prefer native controls like <button> for built-in keyboard/accessibility support
- Write HTML in lowercase; avoid inline styles and !important; minimize the number of imported stylesheets

CSS

- Use CSS custom properties for all reusable values, colors, spacing, font sizes. Define in :root, change once and it updates everywhere. They can also be manipulated with JS in real time, great for win/lose state theming
- Use CSS Grid for 2D layouts, Flexbox for 1D. Prefer relative units over fixed px for responsiveness
- Design mobile-first: base styles for small screens, add complexity with min-width media queries
- Remove unused CSS (PurgeCSS), minify before deployment, avoid inline styles

JavaScript

- camelCase for variables/functions, PascalCase for classes. Use const by default, let when needed, never var
- One function = one job. Keep game logic, DOM updates, and animations in separate functions
- Wrap code in an IIFE or ES Module to avoid polluting the global scope, important for keeping 50 runs isolated
- Toggle CSS classes to change visual state rather than setting .style directly in JS
- Cache DOM references outside of loops/animation frames; minify and lazy-load assets for performance
- Use ES6+: arrow functions, template literals, destructuring. Pick a style guide and lint with ESLint

Accessibility

- Four principles: Perceivable, Operable, Understandable, Robust
- Minimum contrast ratio: 4.5:1 for normal text, 3:1 for large text

- Never rely on color alone to convey meaning; all buttons must be keyboard-operable; never remove: focus outlines
- Minimum font size 16px; minimum touch target 44*44px; use prefers-reduced-motion for animations
- Test with Chrome Lighthouse and the Axe browser extension

Performance

- Minify HTML/CSS/JS; compress with Gzip/Brotli; use WebP images with loading="lazy"
- Inline critical (above-the-fold) CSS in <head> to eliminate render-blocking
- Use font-display: swap for web fonts; audit regularly with Chrome Lighthouse

Team & Version Control

- Feature branches + pull requests; meaningful commit messages; never commit directly to main
- Consistent folder structure (/css, /js, /assets, index.html); document decisions in README

Relevance to Slot Machine Project

- Use CSS variables for win/lose/token color states so they're easy to swap during refinement rounds
- IIFE or ES Module scope keeps each of the 50 runs clean and isolated
- Toggle CSS classes (not inline styles) for spin animations, win flashes, etc.
- Responsive Flexbox/Grid layout could be a differentiator in the evaluation rubric
- Keyboard-accessible spin button is a quick accessibility win