Day 6 — Event Loop II (Rendering & Pitfalls)

1. Rendering Frames

Modern browsers target 60fps, giving each frame about 16.67ms to run JavaScript, recalculate styles, layout the page, paint, and composite. Missing this budget results in visible jank and UI lag.

2. Long Tasks

A long task is any JavaScript execution lasting more than 50ms. These tasks block rendering and user interactions, causing freezes, delayed input, and unresponsive UI moments.

3. Microtask Floods

Overusing microtasks (via Promises or queueMicrotask) can block rendering entirely because all microtasks must finish before the browser paints. This leads to severe jank and delayed UI updates.

4. Ordering with fetch / then / await

fetch() places its .then handler in the microtask queue. await expressions also enqueue microtasks. Since microtasks run before rendering or macrotasks, long async chains can delay repainting and cause UI stalls.

5. Performance Patterns

Debounce: waits until user input stops before executing a function.
Throttle: ensures a function executes at fixed intervals despite frequent events.
Chunking Work: breaks heavy operations into smaller units using setTimeout or requestAnimationFrame, allowing rendering between chunks and preventing jank.