HTML learning roadmap

Alright 🚀 we've covered:

1. **DNS & Networking**
2. **TCP Handshake**
3. **Rendering: Parsing → Layout → Compositing → Painting/Rasterization**

👉 Now comes the **final stage of the browser pipeline**:

🖥️ Step 5: Compositing & Display on Screen

1️⃣ Composition → The "Stage Director" 🎭

- Remember: in **Rasterization**, each layer got turned into **tiles of pixels**.
- Now the **Compositor** takes all those tiles and says:

  > "Put this tile here, that tile above it, this one behind that one, etc."

It's like a **stage director** arranging props, actors, and backgrounds in the right order before the curtain goes up.

2️⃣ GPU's Role → The Projector 🎥

- The GPU receives the final **list of tiles** from the compositor.
- It draws them **really fast** using hardware acceleration.
- Example: scrolling a long webpage
  - You don't repaint everything.
  - The GPU just shifts existing tiles and loads new ones when needed.

3️⃣ Display Refresh Cycle ⏱️

- Your monitor refreshes **60 times per second** (60Hz) or higher.
- The browser must "feed" frames to the GPU **in sync** with this refresh.
- If the browser misses a frame (too slow to composite), you get **jank/stutter**.

👉 Example:

- Smooth animation = browser delivers 60 frames in 1 second.
- Laggy animation = browser delivers only 40 frames → feels choppy.

4️⃣ Developer Relevance 🧑‍💻

- **GPU Compositing is cheap**: moving pre-rasterized layers (like images, transformed divs) is super fast.
- **CPU-bound operations are costly**: forcing layout/paint makes the browser redo a lot of work.
- **Best Practices for Smooth UI:**
  - Use `transform` + `opacity` for animations (GPU-optimized).
  - Avoid triggering layout/paint repeatedly in JS loops.
  - Use `will-change` or `translateZ(0)` carefully to promote layers (but don't overuse → memory overhead).

✅ **In short:**

- The compositor arranges rasterized tiles like a **collage**.
- The GPU rapidly draws them to your screen in sync with monitor refresh.
- Smoothness depends on hitting frame deadlines (16.6ms per frame for 60Hz).

Do you want me to now **wrap everything from Step 1 → Step 5 into a single "browser lifecycle" notes document** for you, or should I expand more on **developer pitfalls (like what causes reflow, repaint, jank)** before making notes?