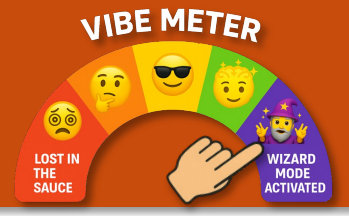
BONUS LEVELS: REFERENCE SLIDES

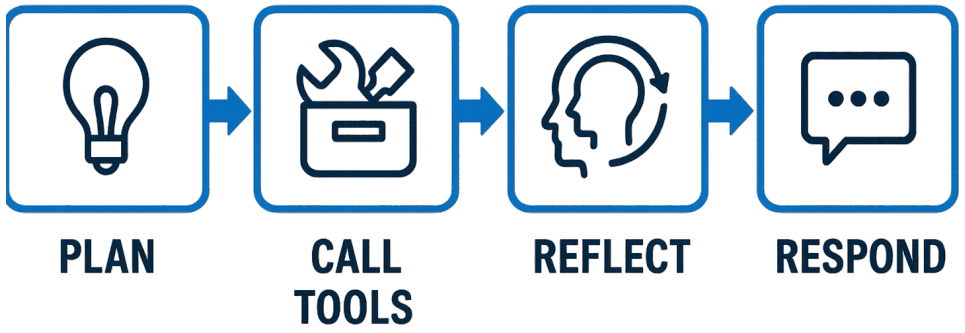


The following slides contain reference materials targeting specific issues you may want to address and extra tips and techniques you can use when Vibe Coding for XR:

- Plan, Call Tools, Reflect, Respond
- REST vs. MCP (Model Content Protocol)
- Generative AI Tools with APIs/MCP Support
- Expert Level Prompting with Plans
- Live Coding/Testing Setup
- Smart Asset Caching for XR
- Addressing API Rate Limitations
- Cheats and Shortcodes
- God Mode
- Boss Level

PLAN, CALL TOOLS, REFLECT, RESPOND

🛠️ Best Practices for AI Agents:



Don’t guess. Use tools to inspect files, read code, and verify assumptions.

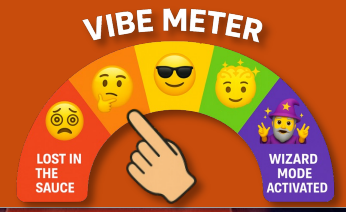
Plan first. Think through each step before taking action.

Reflect after. Review results before continuing … insight > autopilot.

Stay on task. Only stop when the problem is fully solved.

Chain smartly. Use persistent memory or prompt chaining for continuity.

REST vs MCP (Model Context Protocol)



🧠 **Quick Analogy:**

REST = Direct plug MCP = Smart USB hub for AI tools

Not the MCP we're talking about!

✅ **REST**

- One request → One response
- Great for static data (models, textures)
- Used in Web APIs & game engines

✅ **MCP**

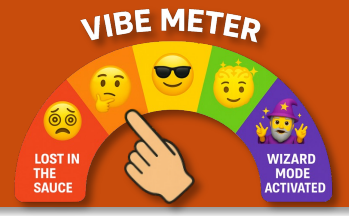
- Manages tools like a smart adapter
- Keeps memory + context
- Chains multiple AI calls seamlessly

📌 **Tip:**

Use REST for loading assets

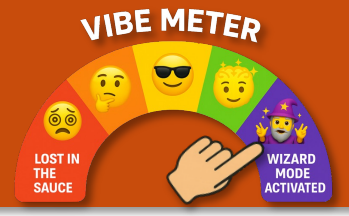
Use MCP for orchestrating logic + reasoning

GENERATIVE AI TOOLS WITH APIS/MCP



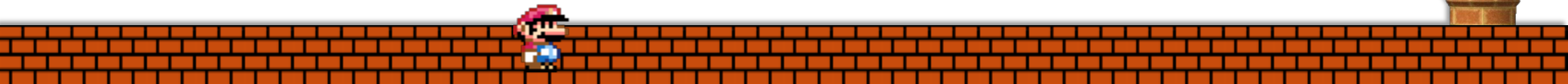
MEDIA TYPE	RECOMMENDED TOOLS
Image & Concept Art	DALL·E 3, SDXL, Stable Diffusion
360° Skyboxes	Blockade Labs, SDXL
3D Models & Meshes	Meshy.AI, Hunyuan3D 2.5, Sloyd, Anything World
SFX & Music	ElevenLabs (SFX), Loudly, Aimi
Voice & Narration	Voice.ai, Murf, Play.ht, ElevenLabs
Characters & NPCs	InWorld.ai, Replika, Joyland
Avatars & Faces	Ready Player Me, D-ID, Synthesia, ZMO.ai, Pinscreen
Video & Animation	Runway, Sora, Pika, Move.ai, DeepMotion
UI / Layout Tools	Uizard, Diagram, Locofy
Multiplayer / Agents	Liv.ai, Croquet, Convai

EXPERT LEVEL PROMPTING (EXAMPLE)



You are a vibe coding expert using Windsurf to generate a complete project plan for an immersive application. The project is a(n) **Gaming and Entertainment** that enables users to **Interact with Objects**, **Navigate Scenes**, **Talk to Characters**, **Play / Compete**, **Roleplay / Embody Character**. Target deployment includes: **Quest** using interface types: **Gaze**, **Voice**, **Hand Tracking**, **Controller**. Performance priority is: **Balanced**. UX style is: **Sci-Fi / Futurist**. Recommended framework: **WebXR - Babylon.js**. Generative AI tools should support: **3d models**, **voiceovers**, **ui panels**, **environments**, **characters**, **code**, **skyboxes**. Suggested GenAI services: **Inworld AI**, **Blockade Labs**. Online features to consider: **Multi-user support**, **Real-time interaction**.

Generate a multi-phase project plan with: 1. Structured build sequence with vibe-coded prompt examples per phase. 2. Required tools, APIs, formats (e.g., .glb, .cbr, .mp3). 3. Sub-prompts for Windsurf to generate each asset and module. 4. Recommended folder structure. 5. Developer tips and warnings. Summarize the app in one sentence, then proceed with phase-specific breakdowns.



LIVE CODING/TESTING SETUP

Enable Developer Mode

↓

Connect Your Quest

↓

Run Your Local Server

↓

Bridge Localhost with

↓

Visit in Quest Browser

Use the Meta Quest app to enable Developer Mode and USB debugging.

Plug your headset into your computer via USB (easiest)

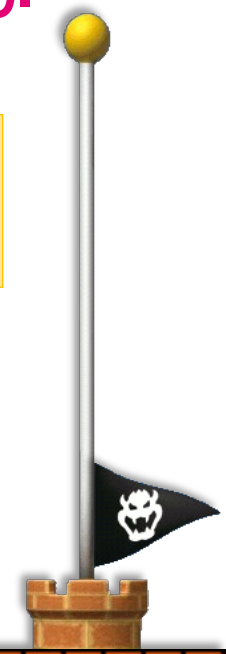
Start Vite, Webpack, or similar on a unique port (5173 in our example).

Run the following in Terminal:

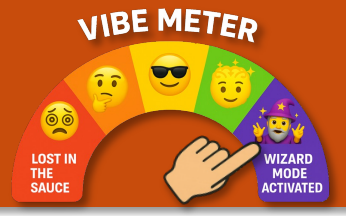
```
adb devices
adb reverse tcp:5173 tcp:5173
```

⚠ Browsers only allow non-HTTPS content from localhost - using ADB reverse tricks your Quest into accepting your local dev server.

On your Quest, go to http://localhost:5173



SMART ASSET CACHING FOR XR



Cascade Base lets you auto-build caching logic into your scene with smart prompting.

1. Cache on Generation

“Save each generated asset to the `/cache/assets/` folder using a hash of its prompt string as the filename. Use subfolders for each media type.”

2. Load if Exists, Else Generate

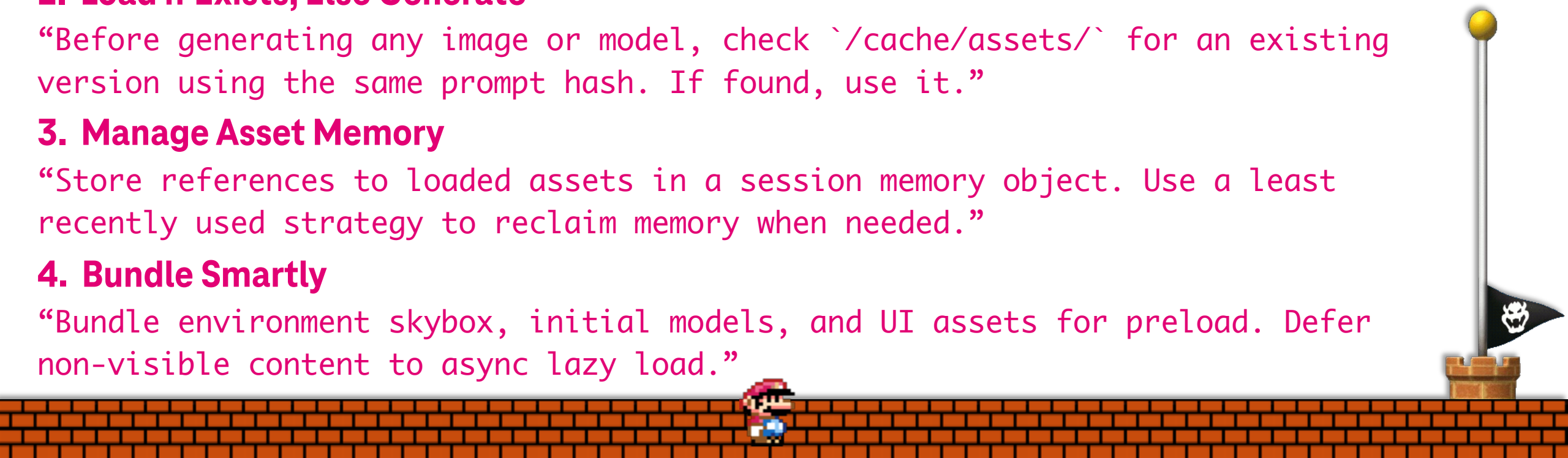
“Before generating any image or model, check `/cache/assets/` for an existing version using the same prompt hash. If found, use it.”

3. Manage Asset Memory

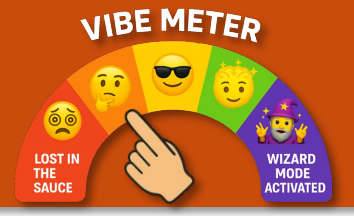
“Store references to loaded assets in a session memory object. Use a least recently used strategy to reclaim memory when needed.”

4. Bundle Smartly

“Bundle environment skybox, initial models, and UI assets for preload. Defer non-visible content to async lazy load.”



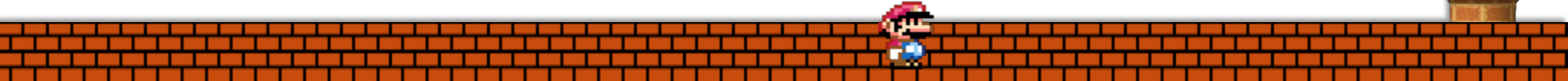
ADDRESSING API RATE LIMITATIONS



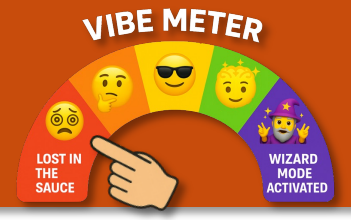
You may encounter API rate limits. APIs usually limit how many requests you can send in a given time frame (per minute/hour/day). **You may get 429 errors, see request failures or unusually long wait times.**

Here's a Windsurf rate limit handler designed to retry failed API calls on a "429 Too Many Requests" error **using exponential backoff: (Wait 1s → 2s → 4s → 8s).**

```
async function callWithRateLimit(apiCall, args, options = {}) {
 const maxRetries = options.retries || 5;
 const baseDelay = options.delay || 1000;

 let attempt = 0;

 while (attempt < maxRetries) {
  try {
   const result = await apiCall(...args);
   return result;
  } catch (err) {
   if (err.status === 429) {
    const waitTime = baseDelay * Math.pow(2, attempt); // exponential backoff
    console.log(`[Rate limit hit. Retrying in ${waitTime}ms...`);
    await new Promise((res) => setTimeout(res, waitTime));
    attempt++;
   } else {
    console.error(`API call failed: ${err.message}`);
    throw err; // Re-throw if not a rate limit issue
   }
  }
 }

 console.error("API call failed after multiple retries due to rate limits.");
 return { status: "error", result: "Rate limit failure" };
}
```



TRAPS, GOTCHAS, AND PITFALLS

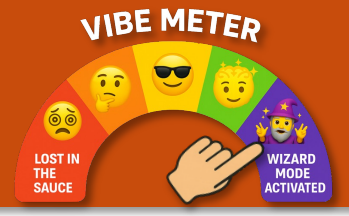


- 😵 **Prompt fatigue:** Too many tweaks = creative burnout. Version early.
- 🎭 **Unclear roles:** Be consistent: dev voice, not wizard → designer → poet.
- 🔒 **Leaking secrets:** Never put API keys in prompts. Use `.env` files.
- 🧩 **Goal drift:** Cool ideas multiply fast. Finish one core feature at a time.
- 🙈 **Perfection trap:** Don't stall at 90%. Test, then tweak.
- 🧟 **Context bleed:** The AI is still thinking about something you said 10 prompts ago. Fix with:

```
Ignore all previous instructions and start fresh. You are now acting as an XR developer assistant focused on generating optimized scene setup code. Respond only based on the new prompt below, and do not reference earlier conversations.
```

"Code responsibly. Vibe freely."

CHEATS AND SHORTCODES



⛓️ **Use Prompt Chaining:** Break large tasks into stepwise prompts.

💾 **Cache Heavy Assets:** Avoid regenerating large files on reload.

🌀 **Set** `temperature = 0.6–0.8`**:** Best balance between creativity & logic.

🧠 **Context Snippets:** Feed LLM real-time XR state, user data, and scene graph.

🎯 **Align Prompt Intent:** Be explicit about desired output formats (JSON, code, etc.)

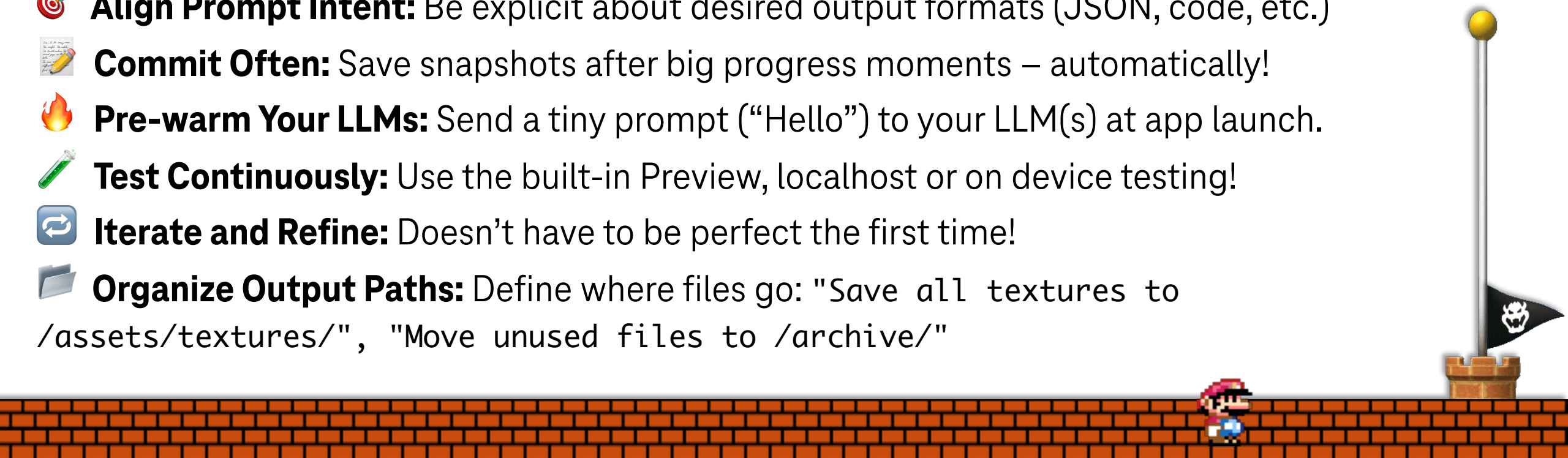
📝 **Commit Often:** Save snapshots after big progress moments – automatically!

🔥 **Pre-warm Your LLMs:** Send a tiny prompt (“Hello”) to your LLM(s) at app launch.

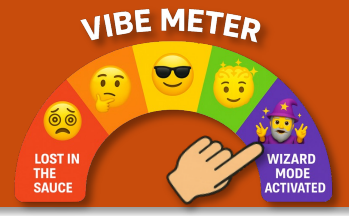
🧪 **Test Continuously:** Use the built-in Preview, localhost or on device testing!

🔁 **Iterate and Refine:** Doesn’t have to be perfect the first time!

📂 **Organize Output Paths:** Define where files go: `"Save all textures to /assets/textures/", "Move unused files to /archive/"`



GOD MODE



Use LLMs (ChatGPT 4o) to create prompts for itself to create prompts for other AI services

Use Perplexity to find designs and APIs (via internet).

Create an "empty" project for specific types of app/targets, clone

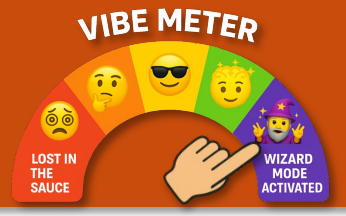
Use Wispr Flow for faster input and vibing. https://wisprflow.ai/terry (3 months free!)

Windsurf: Use persistent memories and user-defined rules to define custom behavior

Windsurf: Copy errors (from any console!) and paste as-is into Cascade input field for immediate fixing

Windsurf: Use Workspace Rules (.windsurfrules) and Global Rules (global_rules.md) to define guidelines for project continuity and to tailor AI behavior.

BOSS LEVEL (GRAB THESE POWER-UPS!)



🧠 **Windsurf**

Your XR co-pilot. Visual prompting, tool orchestration, versioning, auto-commits.

🔗 https://windsurf.app

📦 **OpenAI API Playground**

Explore prompt behavior, model settings, and output formatting.

🔗 https://platform.openai.com/playground

📖 **Prompt Engineering Guide**

The fundamentals of writing structured, modular, and scoped prompts.

🔗 https://github.com/dair-ai/Prompt-Engineering-Guide

👥 **Multi-player Coding (using Lovable)**

Collaborate in real time - edit, debug, and prompt together in the same coding space.

🔗 https://lovable.so