Perfect 👍 You’re moving into **client-side testing** now. Let’s prepare **WSTG notes for “Testing for DOM-Based Cross-Site Scripting (XSS)” (WSTG-CLNT-01)**.

**📝 Notes: Testing for DOM-Based XSS**

**🎯 Purpose**

DOM-Based XSS occurs when **JavaScript on the client side** processes untrusted input from the DOM (URL, fragment, cookies, localStorage, etc.) and writes it back to the page **without proper sanitization**.  
Unlike stored/reflected XSS, the payload is executed **entirely on the client side**, not involving the server’s HTTP response.

**⚡ Common Sources of DOM XSS**

* document.URL, document.documentURI
* document.location, location.hash, location.search
* document.referrer
* window.name
* localStorage / sessionStorage
* Insecure JS functions:
  + document.write(), innerHTML, outerHTML, eval(), setTimeout(), Function()

**🔍 How to Test**

1. **Identify Input Sources**
   * Look for parameters or fragments reflected in the DOM (e.g., ?q=test, #section=1).
2. **Inject Test Payloads**
   * Start simple:
   * #<img src=x onerror=alert(1)>
   * ?q=<svg/onload=alert(1)>
   * Observe in DOM Inspector (DevTools → Sources/Elements).
3. **Use Browser Developer Tools**
   * Set breakpoints in JavaScript code where DOM writes happen.
   * Check if untrusted data flows into dangerous sinks.
4. **Use Security Testing Tools**
   * **Burp Suite DOM Invader** (great for DOM XSS detection).
   * **OWASP ZAP** → DOM XSS plugin.
5. **Check Different Execution Paths**
   * On-page load, on-click, or async JS may trigger delayed execution.

**🛡️ Mitigation**

* **Avoid dangerous sinks** → innerHTML, document.write, eval.
* Use **safe DOM APIs**: textContent, setAttribute().
* Apply **output encoding/escaping** for dynamic DOM updates.
* Use **trusted libraries** (DOMPurify for HTML sanitization).
* Apply **Content Security Policy (CSP)** to restrict inline script execution.
* Validate & sanitize **all client-side data sources**.

✅ **Key Takeaway:**  
DOM XSS happens when **client-side JavaScript takes untrusted input from the DOM and executes it**. It requires **manual testing with payloads and browser tools**, since it may not appear in server responses.

👉 Do you want me to also prepare a **ready-to-use payload cheat sheet** (common DOM XSS test strings) so you can plug them directly into parameters and fragments during testing?