🔹 Perché serve l'HTML

1. I browser **non permettono di trasformare XML con XSLT direttamente dai file `.xml` aperti con `file://`** (per motivi di sicurezza).

2. Se vuoi vedere il risultato in Chrome:

    - Devi **usare HTML + JavaScript** come intermediario.

    - L'HTML carica il tuo XML e XSL, applica la trasformazione e mostra il risultato.

🔹 Cosa fa l'HTML

- Contiene un `<div>` dove inserire il risultato della trasformazione:
  `<div id="output"></div>`
- Contiene il `<script>` che:

    1. Carica `dati.xml` e `stile.xsl`.

    2. Applica la trasformazione XSLT.

    3. Inserisce il risultato dentro il `<div>`.

💡 **In sintesi:**

- Serve **un file HTML** per far girare XML + XSL in Chrome.

- HTML + JS è il "ponte" tra i tuoi file e il browser.

- Una volta creato il file HTML, puoi riutilizzarlo per qualsiasi progetto XML + XSL, cambiando solo i file XML/XSL da caricare.

```
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <title>Test XML + XSL</title>
</head>
<body>

  <!--
    Questo div è dove verrà inserito il risultato della trasformazione XML + XSL
  -->
  <h1>Risultato trasformazione XML + XSL</h1>
```

```
  <div id="output"></div>

          <!--
            Script JavaScript:
          - Carica i file XML e XSL
           - Applica la trasformazione XSLT
           - Inserisce il risultato nel div #output
          -->
  <script>
           // Funzione per caricare un file XML o XSL dal server locale
    async function loadXML(url) {
           // fetch() legge il file dal server
      const res = await fetch(url);
            // res.text() converte il contenuto in testo
      const text = await res.text();
          // DOMParser trasforma il testo in documento XML leggibile da JS
      return new DOMParser().parseFromString(text, "application/xml");
    }

          // Funzione principale per applicare la trasformazione XSLT
    async function applyXSLT() {
      try {
            // Carichiamo i file dati.xml e stile.xsl
        const xml = await loadXML("dati.xml");
        const xsl = await loadXML("stile.xsl");

         // Creiamo un processore XSLT
        const processor = new XSLTProcessor();
         // Importiamo il foglio di stile XSL
        processor.importStylesheet(xsl);

        // Applichiamo la trasformazione al documento XML
        const resultDoc = processor.transformToFragment(xml, document);
          // Inseriamo il risultato dentro il div #output
        document.getElementById("output").appendChild(resultDoc);

      } catch (e) {
          // Se c'è un errore (file non trovato, sintassi sbagliata, ecc.)
            // Mostriamo un messaggio di errore rosso nel div #output
        document.getElementById("output").innerHTML =
          "<p style='color:red'>Errore: " + e.message + "</p>";
      }
    }

          // Chiamiamo subito la funzione per eseguire la trasformazione
    applyXSLT();
  </script>
</body></html>
```

🔹 Come funziona passo passo

1. **HTML** crea la struttura della pagina, con un `<div id="output">` dove comparirà il risultato.
2. **JavaScript**:
    - Carica `dati.xml` e `stile.xsl` dal server locale (Live Server).
    - Applica XSLT per trasformare XML in HTML leggibile.
    - Inserisce il risultato nel div `#output`.
    - Gestisce eventuali errori mostrando un messaggio rosso.
3. Il file HTML **è necessario** perché Chrome non permette di trasformare XML + XSL direttamente da file locali.

1️⃣ Se l'estensione **Live Server** non è installata in Visual Studio Code, nella barra di ricerca in alto scrivi:
Live Server
Cerca l'estensione chiamata **"Live Server"** di **Ritwick Dey** (è la più popolare e affidabile)

2️⃣ Installare l'estensione

3️⃣ Avviare Live Server

1. Apri la cartella del tuo progetto (`index.html`, `dati.xml`, `stile.xsl`) con **File → Open Folder…..**
2. Apri il file `index.html`.
3. In basso a destra della finestra di VS Code comparirà un pulsante **"Go Live"** → cliccaci sopra.
4. Chrome si aprirà automaticamente con un URL locale tipo:

    http://127.0.0.1:5500/index.html

Ora puoi vedere la trasformazione XML + XSL direttamente nel browser