Secure patch for `index.html` and `app.js`

Questo documento contiene le versioni aggiornate e hardenate dei file `index.html` e `app.js` della tua app client-side. Copia i file nel repository per applicare le patch.

---

index.html

```
<!doctype html>
<html lang="it">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Weather App</title>
  <!-- Content Security Policy: limita le sorgenti e connect solo alle API
attese -->
  <meta http-equiv="Content-Security-Policy" content="default-src 'self';
script-src 'self'; style-src 'self' 'unsafe-inline'; img-src 'self' data:
https:; font-src 'self' https://cdnjs.cloudflare.com; connect-src 'self'
https://geocode.xyz https://api.open-meteo.com;">
  <link rel="stylesheet" href="style.css">
  <script defer src="app.js"></script>
</head>
<body>
  <main>
    <h1>Weather App</h1>

    <label for="cityField">Città</label>
    <input id="cityField" maxlength="50" name="city" aria-label="Città"
placeholder="Inserisci una città" />
    <button id="submitBtn" type="button">Cerca</button>

    <div id="loading" class="hidden">Caricamento...</div>
    <div id="errorMessage" role="alert" aria-live="polite"></div>

    <section id="resultContainer" class="hidden" aria-live="polite">
      <h2 id="cityName"></h2>
      <div class="weather">
        <i id="weatherIcon" class="weather-icon fas" aria-hidden="true"></i>
        <p id="temperature"></p>
        <p id="condition"></p>
        <p id="windSpeed"></p>
      </div>
    </section>
  </main>
</body>
</html>
```

app.js

```
// Hardened app.js
const searchBtn = document.getElementById("submitBtn");
const cityField = document.getElementById("cityField");
const cityName = document.getElementById("cityName");
const temperature = document.getElementById("temperature");
const condition = document.getElementById("condition");
const windSpeed = document.getElementById("windSpeed");
const errorMessage = document.getElementById("errorMessage");
const loading = document.getElementById("loading");
const resultContainer = document.getElementById("resultContainer");
const weatherIcon = document.getElementById("weatherIcon");

const apiUrl = "https://api.open-meteo.com/v1/forecast?latitude={lat}
&longitude={lon}&current_weather=true";

function showLoading() {
  loading.classList.remove("hidden");
  resultContainer.classList.add("hidden");
  errorMessage.textContent = "";
}

function hideLoading() {
  loading.classList.add("hidden");
}

function showResults() {
  resultContainer.classList.remove("hidden");
}

function getWeatherIcon(code) {
  const weatherIcons = {
    0: 'fa-sun',
    1: 'fa-cloud-sun',
    2: 'fa-cloud-sun',
    3: 'fa-cloud',
    45: 'fa-smog',
    48: 'fa-smog',
    51: 'fa-cloud-rain',
    53: 'fa-cloud-rain',
    55: 'fa-cloud-rain',
    61: 'fa-cloud-rain',
    63: 'fa-cloud-rain',
    65: 'fa-cloud-showers-heavy',
    71: 'fa-snowflake',
    73: 'fa-snowflake',
    75: 'fa-snowflake',
    77: 'fa-snowflake',
```

```
    80: 'fa-cloud-rain',
    81: 'fa-cloud-showers-heavy',
    82: 'fa-cloud-showers-heavy',
    95: 'fa-cloud-bolt',
    96: 'fa-cloud-bolt',
    99: 'fa-cloud-bolt',
  };

  return weatherIcons[code] || 'fa-question';
}

// Simple client-side rate-limit and debounce (prevent rapid repeated clicks)
let lastSearch = 0;
const MIN_SEARCH_INTERVAL = 3000; // ms

searchBtn.addEventListener("click", async () => {
  const now = Date.now();
  if (now - lastSearch < MIN_SEARCH_INTERVAL) {
    errorMessage.textContent = "Attendi alcuni secondi prima di effettuare
una nuova ricerca.";
    return;
  }
  lastSearch = now;

  const rawCity = cityField.value || "";
  const city = rawCity.trim();

  // Basic input validation: only letters, spaces, hyphen and apostrophe, max
50 chars
  const cityRegex = /^[\p{L}\s\-']{1,50}$/u;
  if (!cityRegex.test(city)) {
    errorMessage.textContent = "Nome città non valido. Usa solo lettere e
spazi (max 50).";
    return;
  }

  showLoading();

  try {
    // Encode user input for use in URL
    const geoUrl = `https://geocode.xyz/${encodeURIComponent(city)}?json=1`;
    const response = await fetch(geoUrl);

    if (!response.ok) {
      throw new Error(`Geocode service error: ${response.status}`);
    }

    const data = await response.json();

    // Validate coordinates returned by external API
    const latt = data && data.latt;
```

```
    const longt = data && data.longt;

    if (!latt || !longt || isNaN(Number(latt)) || isNaN(Number(longt))) {
      errorMessage.textContent = "Coordinate non valide restituite dal
servizio.";
      hideLoading();
      return;
    }

    const weatherResponse = await fetch(apiUrl.replace("{lat}",
encodeURIComponent(latt)).replace("{lon}", encodeURIComponent(longt)));
    if (!weatherResponse.ok) {
      throw new Error(`Weather service error: ${weatherResponse.status}`);
    }

    const weatherData = await weatherResponse.json();

    if (!weatherData || !weatherData.current_weather) {
      errorMessage.textContent = "Impossibile ottenere i dati meteo.";
      hideLoading();
      return;
    }

    hideLoading();
    showResults();

    const weatherCode = weatherData.current_weather.weathercode;
    const iconClass = getWeatherIcon(weatherCode);

    // Safely set text content (prevents XSS) and use classList for icon
classes
    cityName.textContent = city;

    // reset base classes and add icon class safely
    weatherIcon.className = "weather-icon fas"; // reset to known safe base
    if (typeof iconClass === 'string' && /^[a-zA-Z0-9\-]+$/.test(iconClass))
{
      weatherIcon.classList.add(iconClass);
    }

    temperature.textContent = `Temperatura: $
{String(weatherData.current_weather.temperature)}°C`;
    condition.textContent = `Condizione: $
{String(weatherData.current_weather.weathercode)}`;
    windSpeed.textContent = `Velocità del vento: $
{String(weatherData.current_weather.windspeed)} km/h`;
    errorMessage.textContent = "";
  } catch (error) {
    hideLoading();
    console.error(error);
    errorMessage.textContent = "Si è verificato un errore. Riprova più
```

```
tardi.";
  }
});
```

Note

- Il codice applica validazione dell'input, encodeURIComponent per tutte le parti dinamiche dell'URL, controllo dello stato delle risposte HTTP e validazione dei dati restituiti dalle API.
- Ho aggiunto una Content Security Policy di base in `index.html`. Se la app verrà servita da un server, è preferibile impostare la CSP come header HTTP lato server.

Se vuoi, posso: - generare un file `patch.diff` (formato patch/PR) pronto da applicare con `git apply`; - creare un `docker-compose` leggero per eseguire la pagina in modo isolato; - o preparare una PR textuale con i cambiamenti se mi dici il branch di destinazione.
