AI1 **LAB D**	Żółkiewski Filip Album 51163 Grupa 5	**Wersja 1**

REST API CLIENT

SPIS TREŚCI

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie następujących umiejętności:

- pobieranie danych z zewnętrznych zasobów za pomocą REST API
- zdobywanie wiedzy na temat zewnętrznych API za pomocą dokumentacji typu Swagger
- wysyłanie asynchronicznych żądań z wykorzystaniem XMLHttpRequest i Fetch API

W praktycznym wymiarze uczestnicy stworzą dynamiczną stronę HTML pozwalającą na wyświetlanie bieżącej informacji pogodowej oraz prognoz dla zadanej przez użytkownika miejscowości.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie wykonywania połączeń synchronicznych i asynchronicznych z poziomu JS na stornie.

Wejściówka?

UWAGA

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do `Plik` -> `Informacje` -> `Właściwości` -> `Właściwości zaawansowane` -> `Niestandardowe` i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub `Ctrl+A` -> `F9`.

Wymagania

W ramach LAB D przygotowane powinny zostać:

- pojedyncza strona HTML ze skryptem ładowanym z zewnętrznego pliku JS
- pole tekstowe (input typu „text”) do wprowadzania adresu
- przycisk „Pogoda”, po kliknięciu którego wykonywane jest zapytanie asynchroniczne:
  - do API Current Weather: https://openweathermap.org/current za pomocą XMLHttpRequest
  - do API 5 day forecast: https://openweathermap.org/forecast5 za pomocą Fetch API
- obsługa zwrotki z obu API – wypisanie pogody bieżącej oraz prognoz poniżej pola wyszukiwania.

Wygeneruj własny lub wykorzystaj gotowy klucz do API: 7ded80d91f2b280ec979100cc8bbba94

W przypadku blokady można posiłkować się filmem: https://www.youtube.com/watch?v=WoKp2qDFxKk jednakże spróbuj rozwiązać ten problem samodzielnie!

Prowadzący omówi powyższe wymagania. Upewnij się, czy wszystko rozumiesz.

Tu umieść swoje notatki:

…notatki…

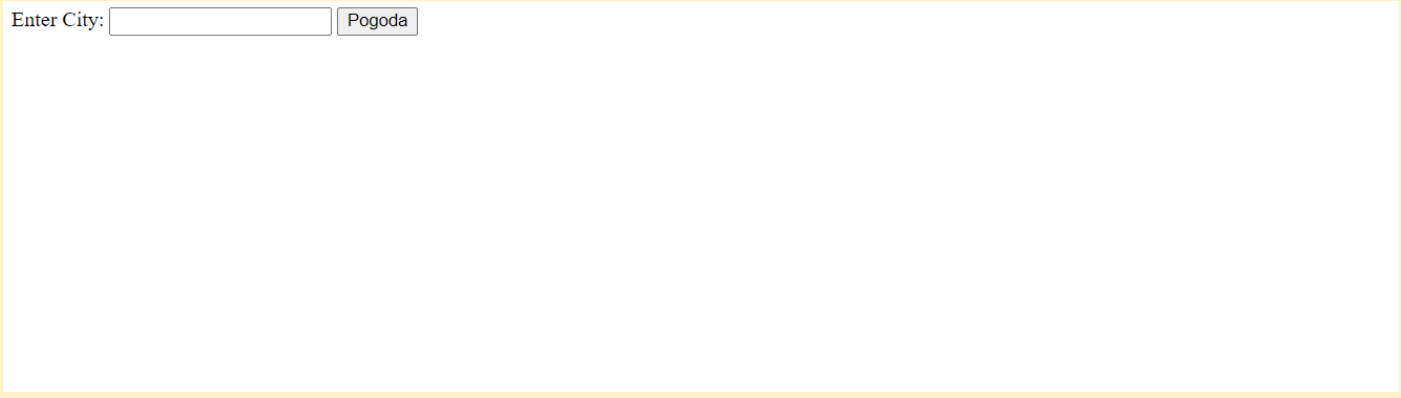
Badanie API

Poświęć kilka minut na wykonanie przykładowych zapytań do API z poziomu pasku adresu przeglądarki. Podaj wymagane parametry dla osiągnięcia różnych wyników. Zbadaj odpowiedzi API, aby uzyskać pełen obraz wymagań i możliwości API.

Implementacja

Tradycyjnie implementację należy zacząć od zbudowania w HTML + CSS wszystkich wymaganych elementów / placeholderów na te elementy. Następnie krok po kroku należy implementować poszczególne zachowania.

Wstaw zrzut ekranu zawierającego stronę ze wszystkimi elementami, tj. pole tekstowe, przycisk, miejsce do wyświetlenia pogody i prognozy:



Punkty:	0	1

Wstaw zrzut ekranu kodu odpowiedzialnego za wysyłanie żądania do current za pomocą XMLHttpRequest:

```
function makeRequest(url, callback) {
    const xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.onload = function () {
        if (xhr.status === 200) {
            const response = JSON.parse(xhr.responseText);
            console.log(response);
            callback(response);
        } else {
            console.error('Request failed. Status:', xhr.status);
        }
    };

    xhr.onerror = function () {
        console.error('Request failed');
    };
    xhr.send();
}
```

Wstaw zrzut ekranu pokazujący otrzymaną odpowiedź za pomocą `console.log()` w przeglądarce.

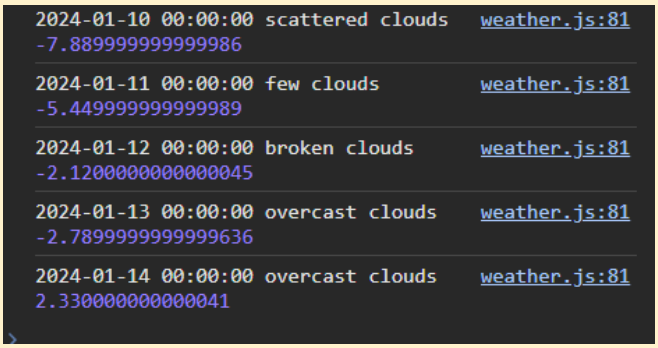
```
                                                    weather.js:46
▶ {message: 'like', cod: '200', count: 0, list: Arra
  y(0)}
                                                    weather.js:46
▶ {message: 'like', cod: '200', count: 3, list: Arra
  y(3)}
                                                    weather.js:46
▶ {message: 'like', cod: '200', count: 3, list: Arra
  y(3)}
```

Punkty:	0	1

Wstaw zrzut ekranu kodu odpowiedzialnego za wysyłanie żądania do forecast za pomocą Fetch:

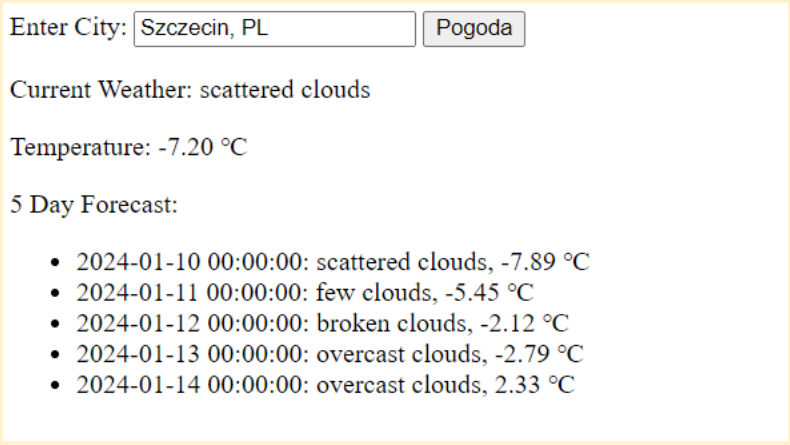
```
function displayForecast(data) {
    const weatherInfo = document.getElementById('weatherInfo');
    const forecast = data.list;
    let forecastHtml = '<p>5 Day Forecast:</p><ul>';

    for (let i = 0; i < forecast.length; i += 8) {
        const date = forecast[i].dt_txt;
        const weather = forecast[i].weather[0].description;
        const temperatureKelvin = forecast[i].main.temp;
        const temperatureCelsius = temperatureKelvin - 273.15;

        forecastHtml += `<li>${date}: ${weather}, ${temperatureCelsius.toFixed(2)} &#8451;</li>`;
        console.log(date,weather,temperatureCelsius);
    }

    forecastHtml += '</ul>';
    weatherInfo.innerHTML += forecastHtml;
}
```

Wstaw zrzut ekranu pokazujący otrzymaną odpowiedź za pomocą `console.log()` w przeglądarce.



Punkty:	0	1

Wstaw zrzut ekranu przedstawiającego wizualizację prognoz pogody:



Upewnij się, że widoczne są pasek wyszukiwania ze wskazaną miejscowością, a także zarówno pogoda bieżąca jak i prognozy pogody.

Punkty:	0	1

COMMIT PROJEKTU DO GIT

Zacommituj i pushnij swoje rozwiązanie do repozytorium GIT.

Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie `lab-c` na podstawie głównej gałęzi kodu.

Podaj link do brancha `lab-d` w swoim repozytorium:

…link, np. https://github.com/zf51163/Aplikacje-Internetowe-Lab1/tree/main/LAB-04

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

…podsumowanie…

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.