AI1 **LAB D**	Jakubiak Hubert Album 50985 Grupa 4	**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…

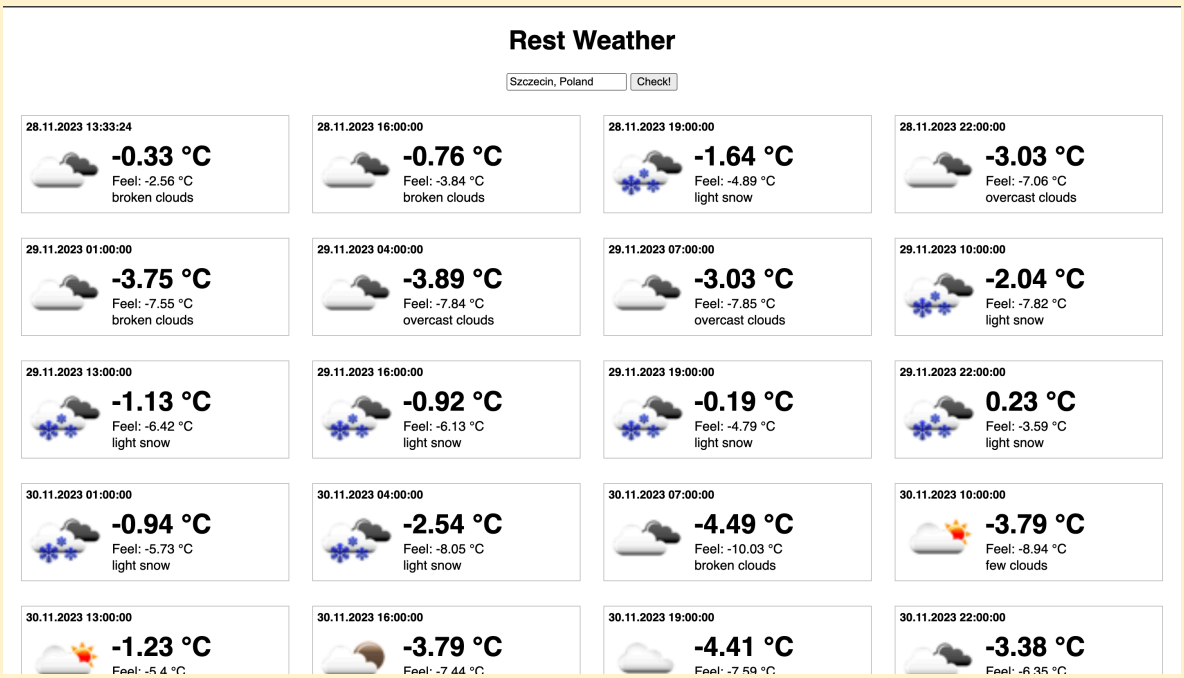
Badanie API

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:

```
1 usage
getCurrentWeather(query) : void  {
    let url : string  = this.currentWeatherLink.replace( searchValue: "{query}", query);
    let req : XMLHttpRequest  = new XMLHttpRequest();
    req.open( method: "GET", url,  async: true);
    req.addEventListener( type: "load",  listener: () : void  => {
        this.currentWeather = JSON.parse(req.responseText);
        this.drawWeather();
    });
    req.send();
}
```

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

```
▼ Object i
    base: "stations"
  ▶ clouds: {all: 51}
    cod: 200
  ▶ coord: {lon: 14.553, lat: 53.4289}
    dt: 1701174804
    id: 3083829
  ▶ main: {temp: -0.33, feels_like: -2.56, temp_min: -1.06, temp_max: -0.16, pressure: 999, …}
    name: "Szczecin"
  ▶ sys: {type: 2, id: 2078561, country: 'PL', sunrise: 1701154153, sunset: 1701183022}
    timezone: 3600
    visibility: 10000
  ▶ weather: [{…}]
  ▶ wind: {speed: 1.79, deg: 308, gust: 4.47}
  ▶ [[Prototype]]: Object
```

Punkty:	0	1

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

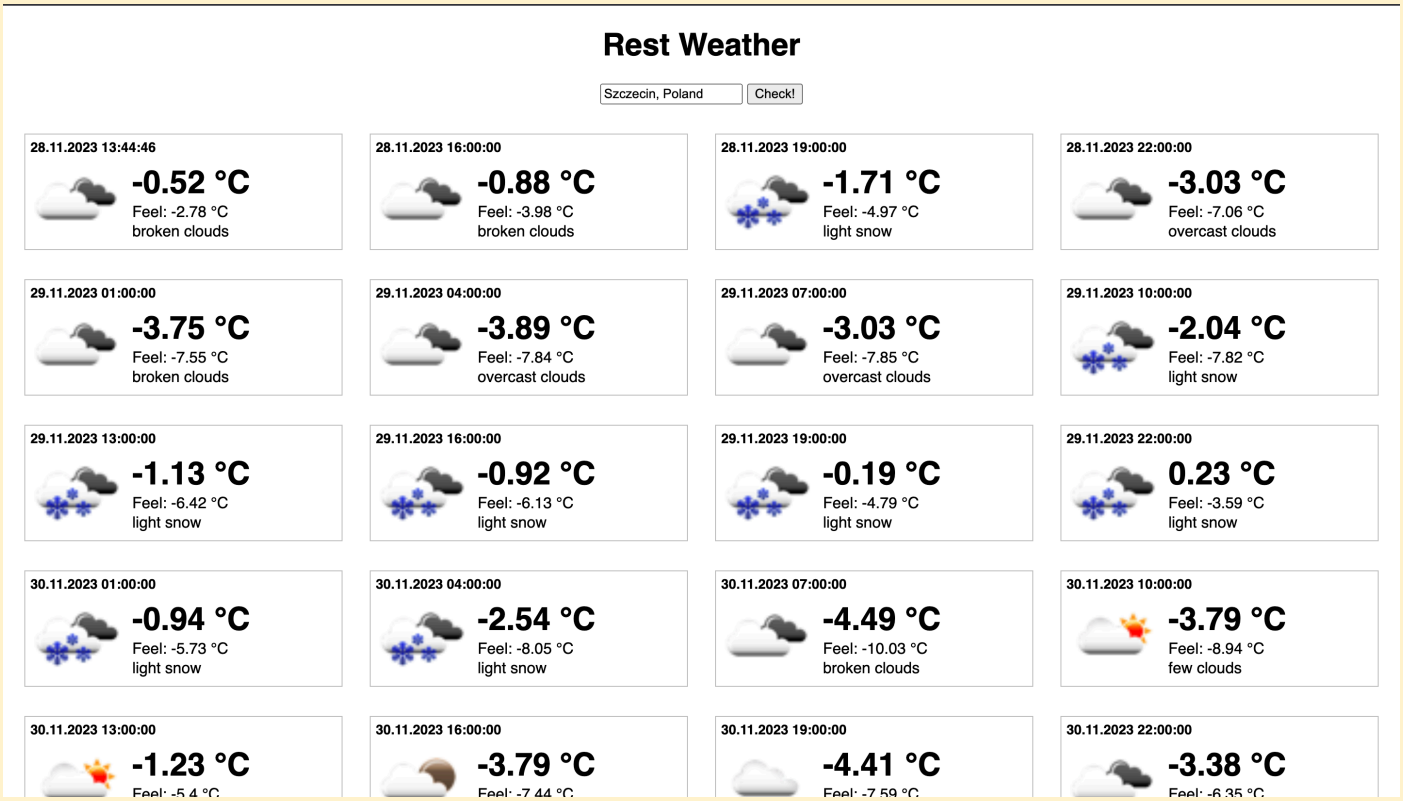
```
getForecast(query) : void  {
    let url : string  = this.forecastLink.replace( searchValue: "{query}", query);
    fetch(url) Promise<Response>
        .then((response : Response ) => {
            return response.json()
        }) Promise<any>
        .then((data) : void  => {
            this.forecast = data.list;
            this.drawWeather()
        })
    ;
}
```

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

```
▼ Array(40) i
  ▶ 0: {dt: 1701183600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 1: {dt: 1701194400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 2: {dt: 1701205200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 3: {dt: 1701216000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 4: {dt: 1701226800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 5: {dt: 1701237600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 6: {dt: 1701248400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 7: {dt: 1701259200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 8: {dt: 1701270000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 9: {dt: 1701280800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 10: {dt: 1701291600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 11: {dt: 1701302400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 12: {dt: 1701313200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 13: {dt: 1701324000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 14: {dt: 1701334800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 15: {dt: 1701345600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 16: {dt: 1701356400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 17: {dt: 1701367200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 18: {dt: 1701378000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 19: {dt: 1701388800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 20: {dt: 1701399600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 21: {dt: 1701410400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 22: {dt: 1701421200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 23: {dt: 1701432000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 24: {dt: 1701442800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 25: {dt: 1701453600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 26: {dt: 1701464400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 27: {dt: 1701475200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 28: {dt: 1701486000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 29: {dt: 1701496800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 30: {dt: 1701507600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 31: {dt: 1701518400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 32: {dt: 1701529200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 33: {dt: 1701540000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 34: {dt: 1701550800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 35: {dt: 1701561600, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 36: {dt: 1701572400, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 37: {dt: 1701583200, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 38: {dt: 1701594000, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
  ▶ 39: {dt: 1701604800, main: {…}, weather: Array(1), clouds: {…}, wind: {…}, …}
    length: 40
  ▶ [[Prototype]]: Array(0)
```

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-d` na podstawie głównej gałęzi kodu.

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

…link, np. https://github.com/jakhub21/jakubiak/tree/lab-d/AI1-LAB-D

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.
W trakcie tego laboratorium zdobyłem umiejętności związane z tworzeniem interaktywnego interfejsu webowego do sprawdzania aktualnej pogody. Nauczyłem się integrować zewnętrzne API, takie jak OpenWeatherMap, przy użyciu języka JavaScript. Tworzenie dynamicznych interfejsów, korzystając z XMLHttpRequest oraz funkcji fetch, umożliwiło mi pobieranie danych pogodowych i aktualizowanie interfejsu w czasie rzeczywistym. Ponadto, zyskałem doświadczenie w manipulowaniu elementami HTML i ich stylizacji za pomocą CSS, tworzeniu dynamicznych komponentów przy użyciu JavaScript oraz organizowaniu i prezentowaniu danych w czytelny sposób na stronie internetowej.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.