

REST API CLIENT

SPIS TREŚCI

Spis treści.....	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga.....	1
Wymagania.....	2
Badanie API	2
Implementacja.....	2
Commit projektu do GIT	4
Podsumowanie	4

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 posilkować 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...

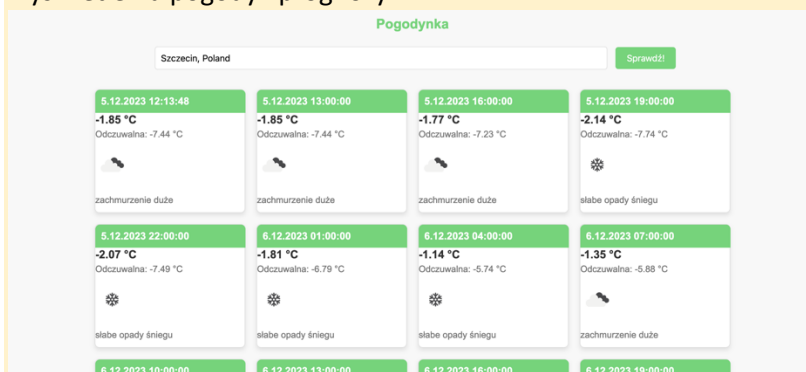
BADANIE API

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:

```
getCurrentWeather(query) {
  let url = this.currentWeatherLink.replace("{query}", query);
  let req = new XMLHttpRequest();
  req.open("GET", url, true);

  req.addEventListener("error", () => this.handleRequestError());
  req.addEventListener("load", () => {
    this.currentWeather = JSON.parse(req.responseText);
    console.log(this.currentWeather);
    this.drawWeather();
  });
  req.send();
}
```

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

```
Object { coord: { _: }, weather: (1) [ _ ], base: "stations", main: { _ }, visibility: 10000, wind: { _ }, clouds: { _ }, dt: 1701770064, sys: { _ }, timezone: 3600, ... }
  base: "stations"
  clouds: Object { all: 99 }
  cod: 200
  coord: Object { lon: 14.553, lat: 53.4289 }
  dt: 1701770064
  id: 3083829
  main: Object { temp: -2.63, feels_like: -9.15, temp_min: -2.93, ... }
  name: "Szczecin"
  sys: Object { type: 2, id: 19799, country: "PL", ... }
  timezone: 3600
  visibility: 10000
  weather: Array [ { _ } ]
  wind: Object { speed: 6.71, deg: 110 }
  <prototype>: Object { _ }
```

Punkty:

0

1

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

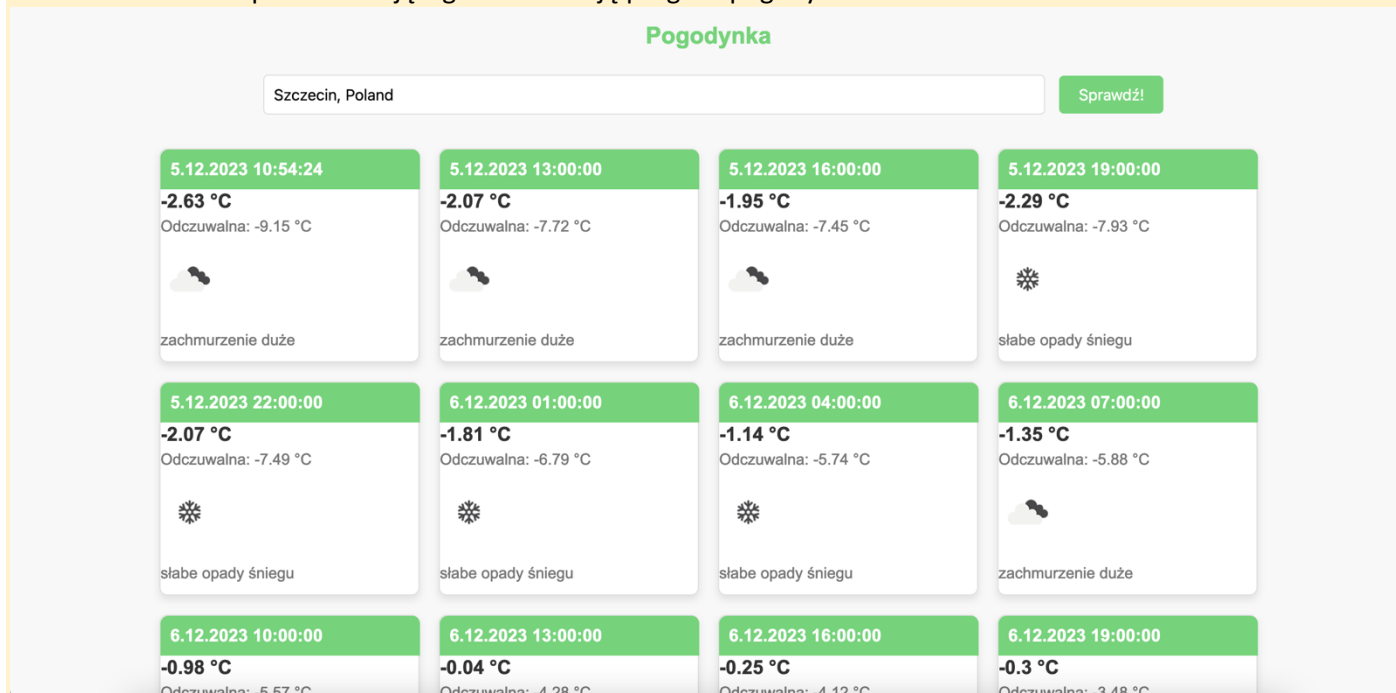
```
getForecast(query) {
  let url = this.forecastLink.replace("{query}", query);
  fetch(url).then((response) => {
    return response.json();
  }).then((data) => {
    console.log(data);
    this.forecast = data.list;
    this.drawWeather();
  });
}
```

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

```
Object { cod: "200", message: 0, cnt: 40, list: (40) [ _ ], city: { _ } }
  city: Object { id: 3083829, name: "Szczecin", country: "PL", ... }
  cnt: 40
  cod: "200"
  list: Array(40) [ { _ }, { _ }, { _ }, ... ]
    0: Object { dt: 1701776000, visibility: 10000, pop: 0, ... }
    1: Object { dt: 1701780400, visibility: 10000, pop: 0.11, ... }
    2: Object { dt: 1701792000, visibility: 197, pop: 0.59, ... }
    3: Object { dt: 1701810000, visibility: 815, pop: 0.79, ... }
    4: Object { dt: 1701820000, visibility: 349, pop: 0.82, ... }
    5: Object { dt: 1701831600, visibility: 10000, pop: 0.33, ... }
    6: Object { dt: 1701842400, visibility: 3744, pop: 0.25, ... }
    7: Object { dt: 1701853200, visibility: 250, pop: 0.65, ... }
    8: Object { dt: 1701864000, visibility: 196, pop: 0.83, ... }
    9: Object { dt: 1701874800, visibility: 141, pop: 0.96, ... }
    10: Object { dt: 1701885600, visibility: 201, pop: 0.98, ... }
    11: Object { dt: 1701896400, visibility: 77, pop: 0.42, ... }
    12: Object { dt: 1701907200, visibility: 24, pop: 0.29, ... }
    13: Object { dt: 1701918000, visibility: 23, pop: 0.35, ... }
    14: Object { dt: 1701928800, visibility: 23, pop: 0.36, ... }
    15: Object { dt: 1701939600, visibility: 23, pop: 0.28, ... }
    16: Object { dt: 1701950400, visibility: 25, pop: 0, ... }
    17: Object { dt: 1701961200, visibility: 41, pop: 0, ... }
    18: Object { dt: 1701972000, visibility: 231, pop: 0, ... }
    19: Object { dt: 1701982800, visibility: 10000, pop: 0, ... }
    20: Object { dt: 1701993600, visibility: 10000, pop: 0, ... }
    21: Object { dt: 1702004400, visibility: 10000, pop: 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-c` na podstawie głównej gałęzi kodu.

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

...link, np. <https://github.com/gracjanjasnos/main/tree/main/lab04>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

AI1 LAB D – Jasnos Gracjan – Wersja 1

W trakcie laboratorium nauczyłem się pracy z zewnętrznymi API. Użyłem klucza API OpenWeatherMap do asynchronicznego pobierania danych za pomocą XMLHttpRequest i Fetch API. Dostosowałem interfejs użytkownika, prezentując informacje pogodowe w czytelny sposób. Dodatkowo, zoptymalizowałem styl strony.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.