

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.....	5
Podsumowanie.....	5

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 posłkować 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:

PL Stargard


Pogoda

POGODA W MIEŚCIE STARGARD


14.11.2023 20:07:38
7.79°C
Feels like 3.96°C

 scattered clouds


14.11.2023 22:00:00
8.45°C
Feels like 4.95°C

 light rain


15.11.2023 01:00:00
8.91°C
Feels like 6.08°C

 light rain


15.11.2023 04:00:00
8.24°C
Feels like 5.8°C

 light rain


15.11.2023 07:00:00
8.15°C
Feels like 5.63°C

 light rain

15.11.2023 10:00:00
7.78°C
Feels like 4.94°C

 light rain

15.11.2023 13:00:00
8.94°C
Feels like 6.65°C

 scattered clouds

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

```

async function getCurrentWeather(lat, lon) : Promise<XMLHttpRequest> {
  let xhr : XMLHttpRequest = new XMLHttpRequest()
  xhr.open( method: "GET", url: 'https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${apiKey}&units=metric', async: true)
  xhr.send()
  return xhr;
}

```

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

```

{"coord":{"lon":15.0316,"lat":53.309},"weather":[{"id":804,"main":"Clouds","description":"overcast clouds","icon":"04d"}],"base":"stations","main":{"temp":10.37,"feels_like":9.6,"temp_min":9.84,"temp_max":10.81,"pressure":996,"humidity":82,"sea_level":996,"grnd_level":994},"visibility":10000,"wind":{"speed":9.69,"deg":248,"gust":18.28},"clouds":{"all":100},"dt":1699961813,"sys":{"type":2,"id":2089756,"country":"PL","sunrise":1699942928,"sunset":1699974392},"timezone":3600,"id":7531735,"name":"Stargard","cod":200}

```

Punkty:	0	1
---------	---	---

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

```
1 usage  1Batrex1
async function getForecastForNextDays(lat, lon) : Promise<Response> {
  return await fetch(
    'https://api.openweathermap.org/data/2.5/forecast?lat=${lat}&lon=${lon}&appid=${apiKey}&units=metric'
  );
}
```

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

```
Object {
  city: {id: 7531735, name: 'Stargard', coord: {}, country: 'PL', population: 0, ...},
  cnt: 40,
  cod: "200",
  list: (40) [{}],
  message: 0,
  [[Prototype]]: Object
}
```

Punkty:

0

1

Wstaw zrzut ekranu przedstawiający 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/1Batrex1/ai1-lab1/tree/lab-d>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Nauczyłem się jak korzystać z api i robić asynchroniczne requesty oraz tego że nie jest to za bardzo intuicyjne
...podsumowanie...

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.