

REST API CLIENT

SPIS TREŚCI

Spis treści1

Cel zajęć1

Rozpoczęcie1

Uwaga1

Wymagania2

Badanie API2

Implementacja2

6

Podsumowanie6

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 klucz do API. Ponieważ aktywacja może chwilę potrwać, na czas trwania laboratorium możesz wykorzystać „służbowy” klucz: `7ded80d91f2b280ec979100cc8bbba94`. **UWAGA!** Klucz zostanie dezaktywowany niedługo po zajęciach. Musisz wygenerować swój własny.

W przypadku blokady twórczej 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...

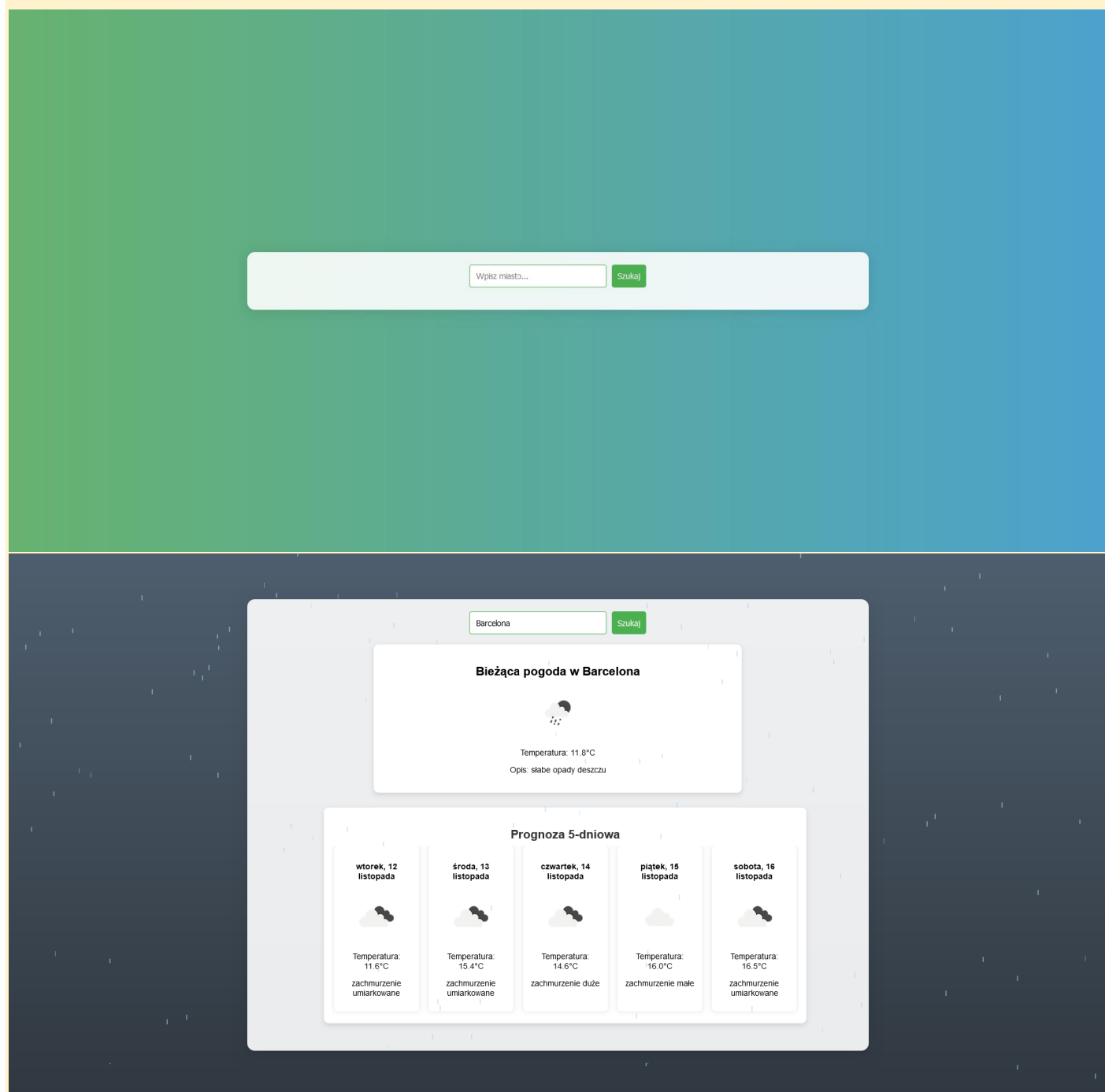
BADANIE API

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: **Na początku widać tylko wyszukiwanie, miejsca na pogodę tworzą się dopiero po wyszukaniu miasta. Stwierdziłem, że będzie to wyglądało estetyczniej.**



Te białe kropki to padający deszcz...

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

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

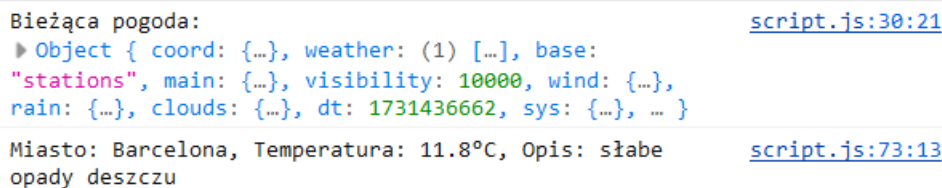
```
const xhr = new XMLHttpRequest();
xhr.open("GET", currentWeatherUrl, true);

xhr.onload = function() {
  if (xhr.status === 200) {
    const data = JSON.parse(xhr.responseText);
    console.log("Bieżąca pogoda:", data); // Logowanie danych o pogodzie
    displayWeather(data);
  } else {
    weatherResult.innerHTML = "Nie udało się znaleźć miasta.";
  }
};

xhr.onerror = function() {
  weatherResult.innerHTML = "Wystąpił błąd połączenia.";
};

xhr.send();
```

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



Bieżąca pogoda: [script.js:30:21](#)
 ▶ Object { coord: {...}, weather: (1) [...], base:
 "stations", main: {...}, visibility: 10000, wind: {...},
 rain: {...}, clouds: {...}, dt: 1731436662, sys: {...}, ... }
 Miasto: Barcelona, Temperatura: 11.8°C, Opis: słabe
 opady deszczu [script.js:73:13](#)

Punkty:

0

1

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

```
fetch(forecastUrl)
  .then(response => {
    if (!response.ok) throw new Error("Nie udało się pobrać prognozy.");
    return response.json();
  })
  .then(data => {
    console.log("Prognoza 5-dniowa:", data); // Logowanie danych o prognozie
    displayForecast(data);
  })
  .catch(error => {
    forecastResult.innerHTML = "<p>Nie udało się pobrać prognozy pięciodniowej.</p>";
    console.error("Błąd:", error);
  });
```

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

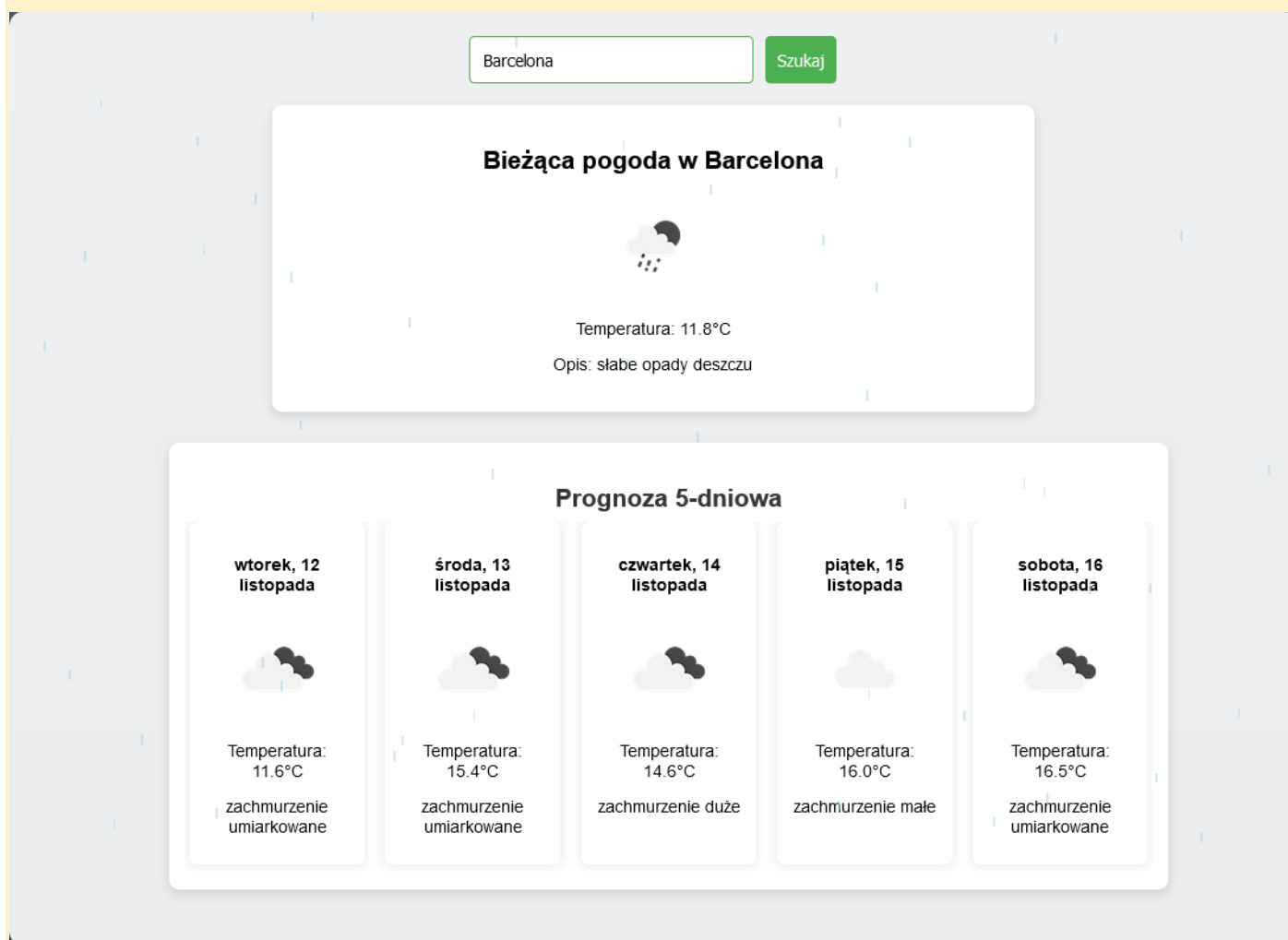
```
Progniza 5-dniowa: script.js:49:21  
▶ Object { cod: "200", message: 0, cnt: 40, list: (40)  
[...], city: {...} }  
Progniza dla wtorek, 12 listopada: 11.6°C, zachmurzenie script.js:108:17  
umiarkowane  
Progniza dla środa, 13 listopada: 15.4°C, zachmurzenie script.js:108:17  
umiarkowane  
Progniza dla czwartek, 14 listopada: 14.6°C, script.js:108:17  
zachmurzenie duże  
Progniza dla piątek, 15 listopada: 16.0°C, zachmurzenie script.js:108:17  
małe  
Progniza dla sobota, 16 listopada: 16.5°C, zachmurzenie script.js:108:17  
umiarkowane
```

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/kapikoks1/a1/tree/main/l4>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

...podsumowanie...

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.