Aplikacje Internetowe 1

Laboratorium 7: REST API Client – OpenWeatherMap.org

|  |  |
| --- | --- |
| Data laboratorium: | 21.11.2022 r. |
| Termin sprawozdania: | 23.11.2022 r. |
| Imię i nazwisko: | Mikołaj Jóźwik |

# 0. Wprowadzenie

Tutaj należy umieścić 2-3 zdania opisujące zawartość sprawozdania i streszczające przebieg laboratorium. Wypełnienie tej rubryki warunkuje dalsze sprawdzenie sprawozdania.

Niniejsze sprawozdanie przedstawia działanie Pogodynki – strony, która pobiera dane z serwisu OpenWeatherMap.org i prezentuje je jako bloki z pogodą.

W dniu 21.11.2022 r., podczas zajęć, doktor Buczyński przedstawił jak powinna wyglądać strona-pogodynka, po czym przystąpiliśmy do implementacji.

# 1. Strona HTML ze wszystkimi elementami

Należy przedstawić HTML ze wszystkimi wymaganymi elementami. 1 punkt.

Table

Description automatically generated

Aktualna wersja strony zawiera następujące elementy:

* Pole w którym wpisuję się miasto
* Przycisk aktywujący – wyświetlenie pogody na daną chwilę wraz z prognozą na 5 dni
* Blok z informacjami o aktualnej pogodzie
* 40 bloków z informacjami o pogodzie na przyszłe 5 dni

**Kod HTML:**

**Text

Description automatically generated**

**Kod CSS:**

**Text

Description automatically generated**

Text

Description automatically generated

**Kod JS:**

**Text

Description automatically generated**

**Text

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated

# 2. Żądanie XMLHttpRequest pod endpoint current

Należy udokumentować pobieranie danych z endpointu current z wykorzystaniem tradycyjnego XMLHttpRequest. Wyświetlać dane z wykorzystaniem console.log(). 1 punkt.

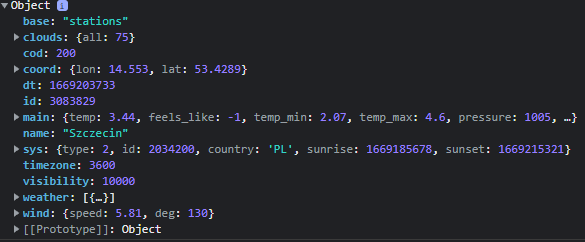
**Kod JS:**

****

Text

Description automatically generated

**Output:**



# 3. Żądanie Fetch API pod endpoint forecast

Należy udokumentować pobieranie danych z endpointu forecast z wykorzystaniem nowoczesnego FETCH API. Wyświetlać dane z wykorzystaniem console.log(). 1 punkt.

**Kod JS:**

****

**Text

Description automatically generated**

**Output:**

**A computer screen capture

Description automatically generated with medium confidence**

# 4. Wizualizacja otrzymanych zwrotek

Należy udokumentować funkcjonalność wyświetlania uzyskanych zwrotek poprzez modyfikację drzewa DOM. 1 punkt.

**Pogoda aktualna:**

**Kod HTML:**

**Text

Description automatically generated**

**Kod JS:**

****

****

****

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Wyjaśnienie:**

Do pierwotnie stworzonego elementu typu *div* z *id=weather-result-container* usuwane są wszystkie podrzędne elementy, a następnie tworzone nowe elementy typy *div* o klasie *class=weather-block*. Ten *div* reprezentuje kafelek z informacjami o pogodzie. Informacje te są pobierane z otrzymanych, za pomocą XMLHTTPRequest, danych w formie JSON. *Div* o klasie *class=weather-block* jest następnie dodawany do ojca *div* z *id=weather-result-container* za pomocą funkcji *appendChild()*.

**Prognoza na 5 dni:**

**Kod HTML:**

**Text

Description automatically generated**

**Kod JS:**

****

**Text

Description automatically generated**

**Text

Description automatically generated**

**Kod CSS:**

**Text

Description automatically generated**

**Output:**

**Diagram

Description automatically generated with medium confidence**

**Wyjaśnienie:**

Dane odebrane za pomocą FetchAPI są zamieniane na JSON, a potem na listę. Podczas rysowania jednego kafelka, wyciągane są dane z jednej pozycji w liście. Następnie kafelek ten, jest dodawany do *div* z *id=weather-forecast*, który ma atrybut CSS -> *flex-wrap: wrap* i *display: wrap* a swoim dzieciom daję atrybut *flex: 1 400px*.

# 5. Linki i uwagi

W tej sekcji należy umieścić link do działającej aplikacji listy zadań oraz dowolne uwagi.

…