



dr inż. Michał Malinowski
michal.malinowski@uth.edu.pl

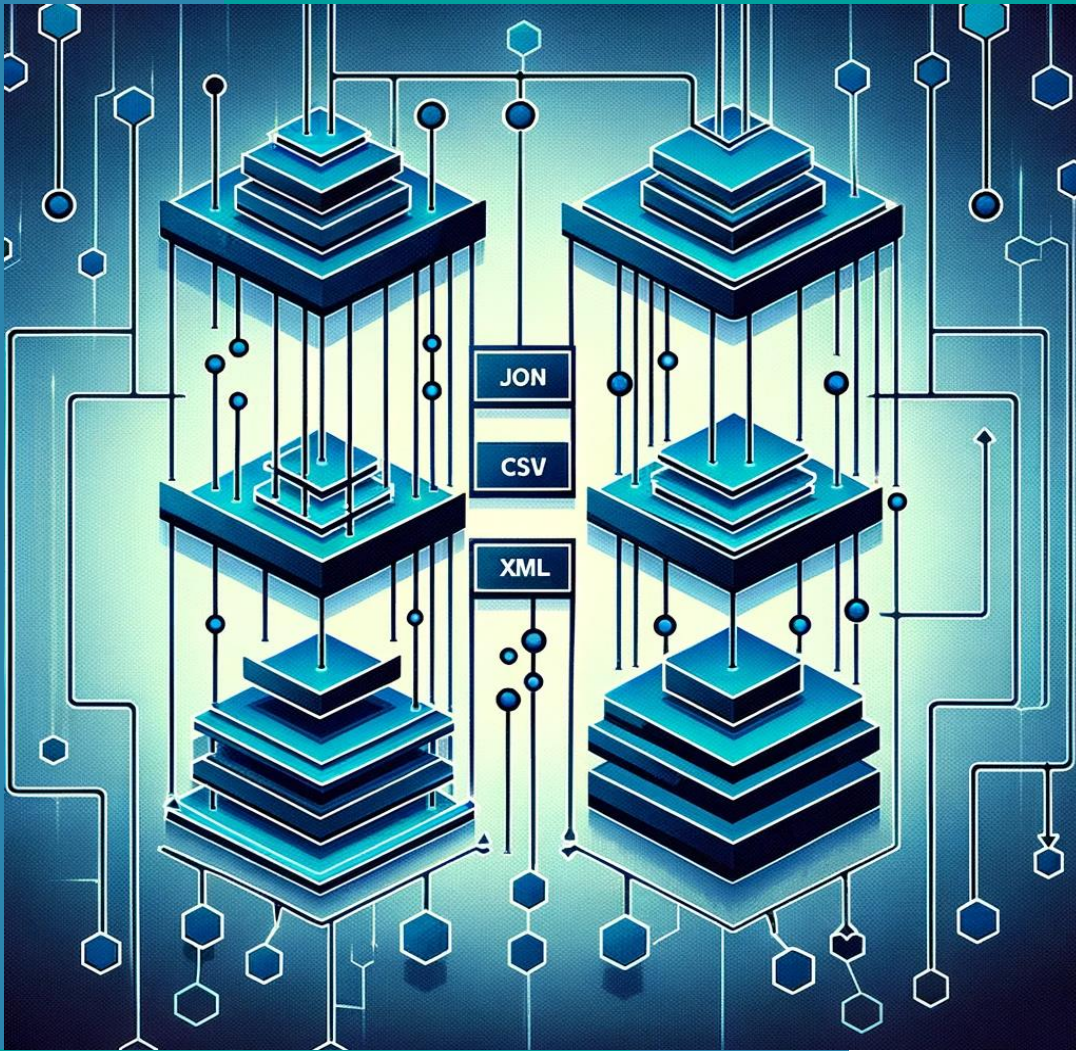


[https://www.linkedin.com/in/
michał-malinowski-malkon/](https://www.linkedin.com/in/michał-malinowski-malkon/)

Formaty wymiany danych

Ćwiczenie 08

ALGORYTMY
I STRUKTURY DANYCH

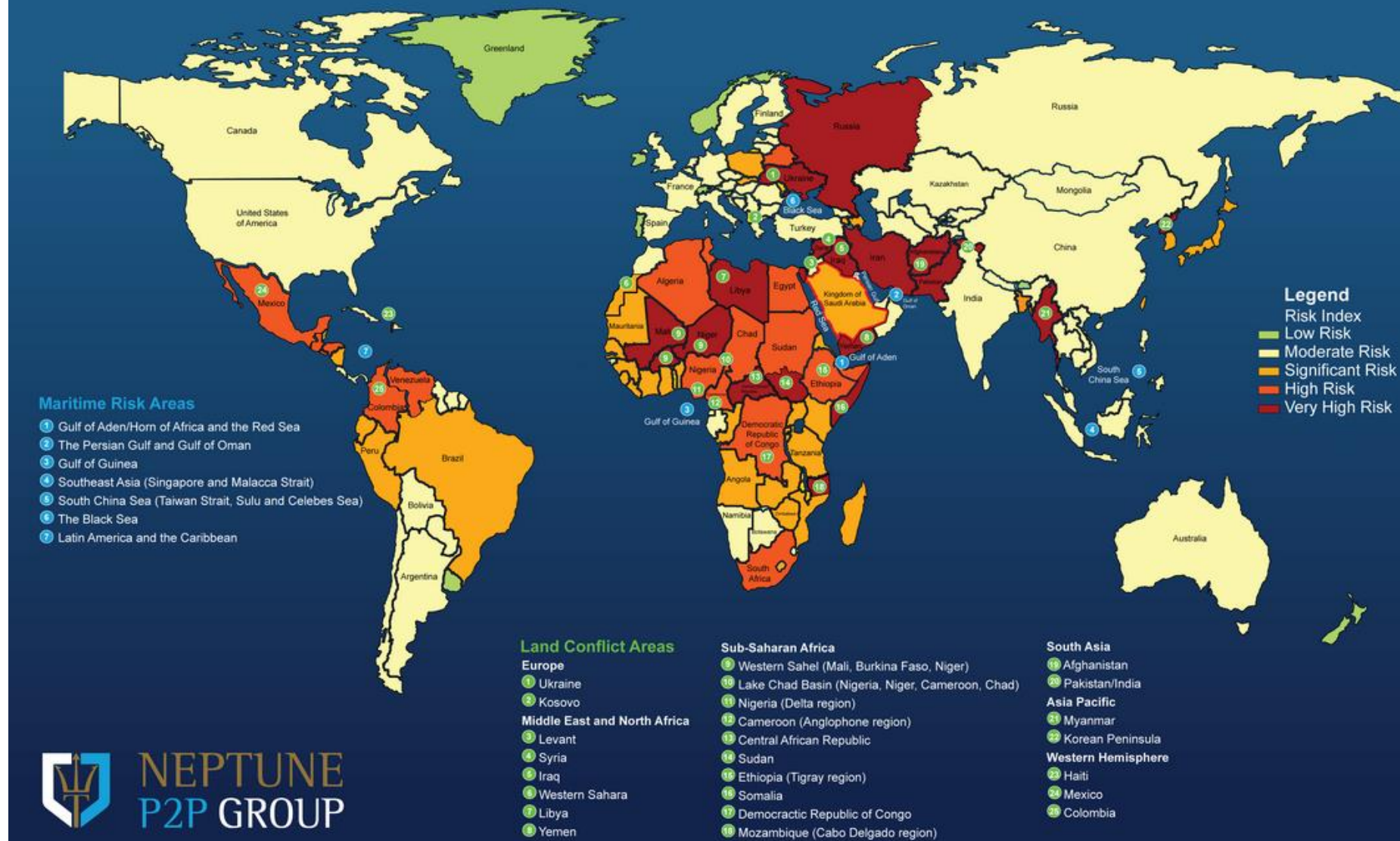


Zagadnienia (2h)

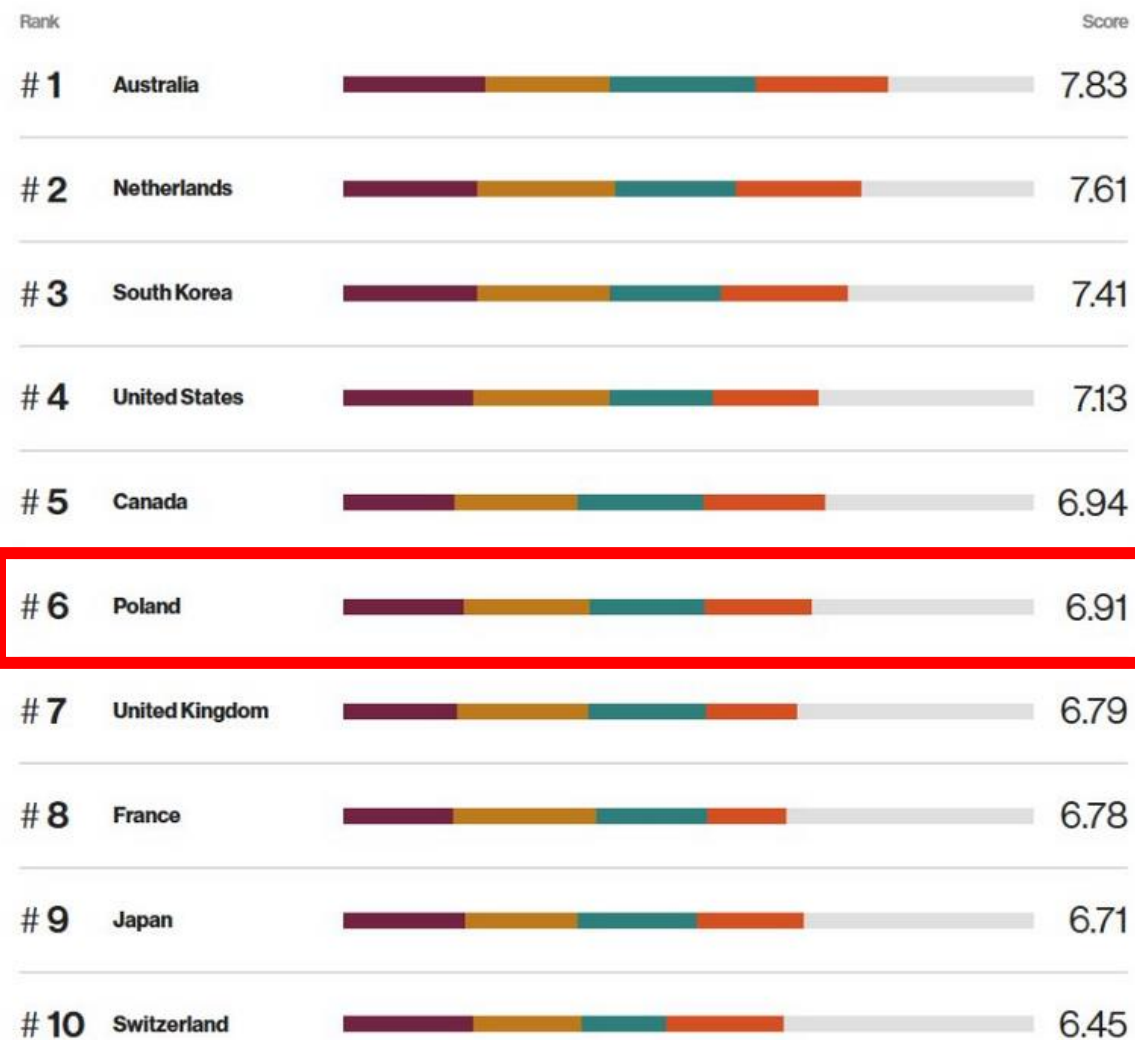
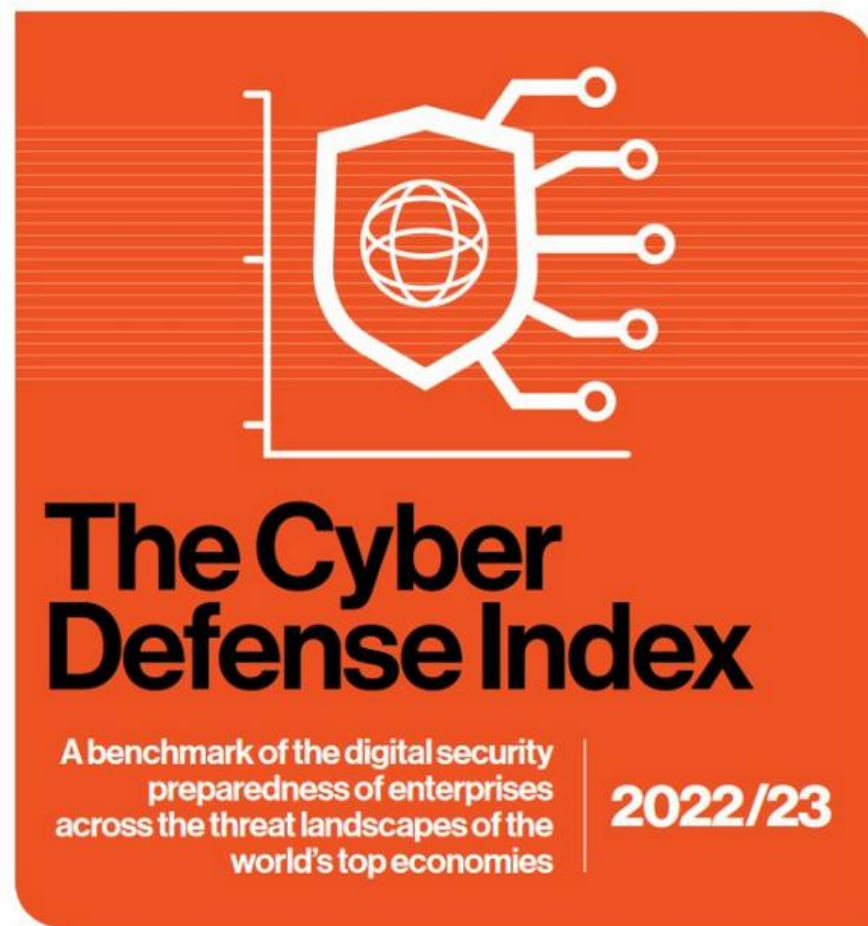
- [Formaty wymiany danych](#)
- [CSV](#)
 - [Zadanie: CSV](#)
 - [Algorytm parsowania: CSV](#)
- [INI](#)
 - [Zadanie: INI](#)
 - [Algorytm parsowania: INI](#)
- [XML](#)
 - [Zadanie: XML](#)
 - [Algorytm parsowania: XML](#)
- [JSON](#)
 - [Zadanie: JSON](#)
 - [Algorytm parsowania: JSON](#)
- [Zadanie: Parsowanie – XML, CSV, INI i JSON](#)

Global Risk and Security Assessment

Visualising Maritime and Terrestrial Risk Q4 2023



MIT Technology Review

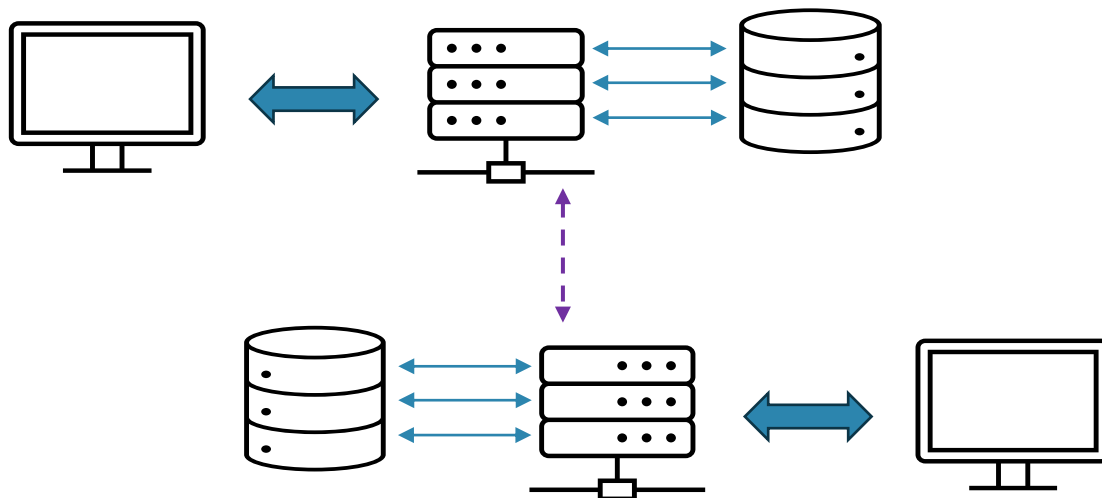


Formaty wymiany danych

Formaty wymiany danych

to standardy, które pozwalają na zorganizowany transport różnych struktur danych pomiędzy różnymi systemami informatycznymi i ich elementami.

Umożliwiają one różnym aplikacjom, które mogą być napisane w różnych językach programowania i działać na różnych platformach, na wzajemne rozumienie i używanie przekazywanych informacji mających różne struktury.



CSV

CSV (Comma-Separated Values)

- Struktura: Format tekstowy reprezentujący dane w formie tabelarycznej. Poszczególne wartości są oddzielone przecinkami (lub innymi separatorami), a każdy wiersz odpowiada jednemu rekordowi.
- Czytelność: Czytelny dla ludzi i prosty do przetwarzania, ale nie obsługuje hierarchii czy złożonych struktur danych.
- Użycie: Często stosowany do eksportu i importu danych z baz danych, arkuszy kalkulacyjnych i innych narzędzi analitycznych.

CSV

Budowa

```
Imię i nazwisko,Adres email,Numer telefonu,Miejsce zamieszkania  
"Jan Kowalski",jankowalski@example.com,555-1234,Warszawa  
"Anna Nowak",annanowak@example.com,555-5678,Kraków  
"Paweł Wiśniewski",pawelwisniewski@example.com,555-9012,Wrocław
```

- Plik tekstowy z rozszerzeniem .CSV
- Pierwsza linia to nagłówek, który definiuje nazwy kolumn.
- Każda kolejna linia reprezentuje jeden rekord, czyli jedną osobę z jej danymi kontaktowymi.
- Wartości są oddzielone przecinkami.
- Każda linia kończy się znakiem nowej linii, oddzielając rekordy od siebie.

Zadanie: CSV

Zadanie

Stwórz plik CSV, który będzie zawierał informacje o książkach w Twojej domowej bibliotece. Każdy rekord powinien zawierać następujące informacje o książce: Tytuł, Autor, Rok wydania, ISBN

- Plik powinien zawierać nagłówek z nazwami kolumn.
- Wartości w poszczególnych kolumnach powinny być oddzielone przecinkami.
- Plik powinien zawierać co najmniej 5 rekordów (książek).

Algorytm parsowania: CSV

Prasowanie JavaScript

Skopiować do pliku .html
i przetestować

```
<script>
// Przykładowe dane CSV w postaci tekstu
const csvString = `Imię,Nazwisko,Wiek
John,Doe,30
Jane,Smith,25
Bob,Johnson,35`;

// Parsuj dane CSV
function parseCSV(csv) {
  const lines = csv.split('\n');
  const result = [];
  // Pobierz nagłówek (pierwsza linia)
  const headers = lines[0].split(',');
  for (let i = 1; i < lines.length; i++) {
    const line = lines[i].split(',');
    if (line.length === headers.length) {
      const entry = {};
      for (let j = 0; j < headers.length; j++) {
        entry[headers[j]] = line[j];
      }
      result.push(entry);
    }
  }
  return result;
}

const parsedData = parseCSV(csvString);
// Dostęp do pól obiektów
parsedData.forEach(entry => {
  console.log("Imię:", entry.Imię); console.log("Nazwisko:", entry.Nazwisko); console.log("Wiek:", entry.Wiek);
});
</script>
```

XML

XML (eXtensible Markup Language)

- Struktura: Format oparty na znacznikach, pozwalający na definiowanie niestandardowych tagów, co zapewnia dużą elastyczność. Struktura dokumentu XML jest reprezentowana przez drzewo.
- Czytelność: Czytelny dla ludzi, ale mniej zwarty niż JSON.
- Użycie: Używany w wielu systemach do przechowywania i transportu danych, w usługach webowych, w dokumentach (np. Microsoft Office) oraz w konfiguracjach.

XML

Budowa

```
<?xml version="1.0" encoding="UTF-8"?>
<kontakty>
  <osoba>
    <imie_i_nazwisko>Jan Kowalski</imie_i_nazwisko>
    <adres_email>jankowalski@example.com</adres_email>
    <numer_telefonu>555-1234</numer_telefonu>
    <miejsce_zamieszkania>Warszawa</miejsce_zamieszkania>
  </osoba>
  <osoba>
    <imie_i_nazwisko>Anna Nowak</imie_i_nazwisko>
    <adres_email>annanowak@example.com</adres_email>
    <numer_telefonu>555-5678</numer_telefonu>
    <miejsce_zamieszkania>Kraków</miejsce_zamieszkania>
  </osoba>
</kontakty>
```

- Plik tekstowy z rozszerzeniem .xml
- Pierwsza linijka to deklaracja informuje parser, że dokument jest w formacie XML wersji 1.0 i używa kodowania UTF-8.
- Każdy kontakt jest reprezentowany przez element <osoba>.
- Dane są wydzielone do odpowiednich elementów potomnych, takich jak <imie_i_nazwisko>, <adres_email>, <numer_telefonu>, i <miejsce_zamieszkania>.
- Elementy te są zagnieżdżone wewnątrz głównego elementu <kontakty>, który zawiera całą listę kontaktów.

Zadanie: XML

Zadanie

Stwórz plik XML, który będzie zawierał informacje o książkach w Twojej domowej bibliotece. Każdy element XML powinien zawierać następujące informacje o książce: Tytuł, Autor, Rok wydania, ISBN

- Każda książka powinna być reprezentowana przez element <książka>.
- Dane są wydzielone do odpowiednich elementów potomnych, takich jak <tytuł >, <autor>, <rok_wydania> i <isbn>.
- Plik powinien zawierać co najmniej 3 elementy książka.

Algorytm parsowania: XML

Prasowanie JavaScript

Skopiować do pliku .html
i przetestować

```
<script>
// Przykładowe dane XML w postaci tekstu
const xmlString = `
<person>
  <name>John Doe</name>
  <email>johndoe@example.com</email>
  <age>30</age>
</person>`;

// Parsuj dane XML
function parseXML(xml) {
  const parser = new DOMParser();
  const xmlDoc = parser.parseFromString(xml, "text/xml");
  return xmlDoc;
}

const parsedData = parseXML(xmlString);

// Teraz 'parsedData' zawiera obiekt dokumentu XML, który można przeglądać
const name = parsedData.querySelector("name").textContent;
const email = parsedData.querySelector("email").textContent;
const age = parsedData.querySelector("age").textContent;

console.log("Imię:", name);
console.log("Email:", email);
console.log("Wiek:", age);
</script>
```

INI

INI (Initialization File Format)

- Struktura: Format tekstowy używany do przechowywania konfiguracji w prostych parach klucz-wartość, z danymi zgrupowanymi w sekcje. Sekcje są zaznaczone w nawiasach kwadratowych [], a każda para klucz-wartość jest oddzielona znakiem równości =.
- Czytelność: Format INI jest łatwy do odczytania i edycji przez ludzi, z wyraźnie oddzielonymi sekcjami i prostymi parametrami konfiguracji. Nie obsługuje jednak zagnieżdżonych struktur danych ani bardziej złożonych typów danych.
- Użycie: Format INI jest szeroko stosowany w konfiguracji oprogramowania, szczególnie na platformach Windows, ale także w różnych aplikacjach na innych systemach operacyjnych. Jest też często używany w starszym oprogramowaniu i grach komputerowych.

INI

Budowa

```
; Komentarz - zaczyna się od znaku średnika
```

```
[OpcjeAplikacji]
```

```
klucz1 = wartość1
```

```
klucz2 = wartość2
```

```
klucz3 = wartość3
```

```
[InnaSekcja]
```

```
ustawienieA = true
```

```
ustawienieB = 42
```

- Plik tekstowy z rozszerzeniem .ini
- Linie zaczynające się od znaku średnika (;) są komentarzami i są ignorowane podczas analizy.
- Sekcje są otoczone nawiasami kwadratowymi ([nazwa_sekcji]) i zawierają pary klucz-wartość związanymi z tą sekcją.
- Każda para klucz-wartość składa się z klucza (po lewej stronie znaku równości =) i wartości (po prawej stronie znaku równości =).

Zadanie: INI

Zadanie

Stwórz plik INI, który będzie zawierał konfigurację aplikacji. Twoja aplikacja ma następujące ustawienia konfiguracyjne:

- Ścieżka do bazy danych.
- Tryb debugowania (true lub false).
- Maksymalna liczba wyników na stronie.

Plik INI powinien zawierać sekcję [Konfiguracja].

W sekcji [Konfiguracja] powinny być zawarte klucze DatabasePath, DebugMode i MaxResultsPerPage z odpowiednimi wartościami.

Algorytm parsowania: INI

Prasowanie JavaScript

Skopiować do pliku .html
i przetestować

```
<script>
// Przykładowe dane INI w postaci tekstu
const iniString = `
[Section1]
key1 = value1
key2 = value2

[Section2]
key3 = value3
key4 = value4
`;

// Parsuj dane INI
function parseINI(ini) {
  const lines = ini.split('\n');
  const result = {};
  let currentSection = null;
  lines.forEach(line => {
    line = line.trim();
    if (line.startsWith('[') && line.endsWith(']')) {
      currentSection = line.substring(1, line.length - 1);
      result[currentSection] = {};
    } else if (line.includes('=')) {
      const [key, value] = line.split('=');
      if (currentSection) {
        result[currentSection][key.trim()] = value.trim();
      }
    }
  });
  return result;
}

const parsedData = parseINI(iniString);

// Dostęp do pól obiektu
console.log("Section1 key1:", parsedData.Section1.key1);
console.log("Section2 key4:", parsedData.Section2.key4);
</script>
```

JSON

JSON (JavaScript Object Notation)

- Struktura: Lekki format tekstowy, bazujący na składni obiektów JavaScript. Używa nawiasów klamrowych {} do reprezentacji obiektów (czyli zestawów par klucz-wartość) i nawiasów kwadratowych [] dla tablic.
- Czytelność: Bardzo czytelny zarówno dla ludzi, jak i maszyn.
- Użycie: Szeroko stosowany w aplikacjach internetowych dla wymiany danych między klientem a serwerem, a także jako format konfiguracyjny i w web APIs.

JSON

Budowa

```
{
  "imie": "Jan",
  "nazwisko": "Kowalski",
  "wiek": 30,
  "miasto": "Warszawa",
  "znajomi": ["Anna", "Paweł", "Ewa"],
  "adres": {
    "ulica": "Krakowska 123",
    "kod_pocztowy": "00-000",
    "kraj": "Polska"
  }
}
```

- Plik tekstowy z rozszerzeniem .json
- Dane są przedstawione w formie par "klucz"-"wartość".
- Obiekty (zestawy danych) są otoczone klamrami {}.
- Klucze są zawsze w podwójnych cudzysłowach (np. "imie").
- Wartości mogą być typu tekstowego, liczbowego, logicznego, tablicy lub innego obiektu JSON.
- Tablice są listami wartości otoczone kwadratowymi nawiasami [].

Zadanie: JSON

Zadanie

Stwórz plik JSON, który będzie zawierał listę Twoich ulubionych książek wraz z informacjami o każdej z nich. Każda książka powinna być reprezentowana jako obiekt JSON z następującymi polami:

- tytuł (string)
- autor (string)
- rok_wydania (number)

Utwórz co najmniej trzy obiekty reprezentujące różne książki.
Ostateczny plik JSON powinien zawierać tablicę tych obiektów.

Algorytm parsowania: JSON

Prasowanie JavaScript

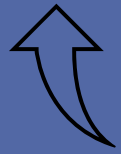
Skopiować do pliku .html
i przetestować

```
<script>
// Przykładowe dane JSON w postaci tekstu
const jsonString = '{"name": "John", "age": 30, "city": "New York"}';

// Parsuj dane JSON
const parsedData = JSON.parse(jsonString);

// Teraz 'parsedData' zawiera obiekt JavaScript z danymi JSON
console.log(parsedData);

// Dostęp do pól obiektu
console.log("Imię:", parsedData.name);
console.log("Wiek:", parsedData.age);
console.log("Miasto:", parsedData.city);
</script>
```



Zadanie: Parsowanie – XML, CSV, INI i JSON

Zadanie (2 pkt)

Dostosuj algorytmy parsowania XML, CSV, INI i JSON

Do swoich danych z 4 zadań



Przedstawienie wyniku

- Dokumenty HTML / ZIP
- Umieszczenie wyniku w Moodle

Termin

- Podany przy zadaniu w Moodle (około 2 tygodnie)