Overview and Project Objective

1. Project Overview

This script is designed to scrape mobile subscription data from the Swisscom website. It automates data extraction, organizes the information into structured JSON, and makes it available via an API endpoint.

2. Objectives

- Extract comprehensive and accurate data about mobile subscriptions.
- Make the data easily accessible for analysis or integration into other systems.
- Automate the process to keep data updated daily across multiple websites.

3. Tools & Technologies Used

- **Node.js & Playwright**: For web scraping and browser automation.
- **Express.js**: To create a REST API for data access.
- **dotenv**: For managing environment variables.
- **JavaScript**: For data processing and extraction logic.

Data Extraction Process

1. Steps for Data Extraction

1. Navigate to the target website.
2. Handle cookies consent modal and UI navigation.
3. Extract main titles, subtitles, and offer details (e.g., header text, prices, features).
4. Use dynamic filtering to categorize data into packages (e.g., "blue Abos," "Kids Abos").
5. Access additional data from external pages linked to each subscription.

2. Sample Extracted Data

Below is an example of the structured data extracted:

json

```json
{
  "page_title": "Swisscom Mobile Subscriptions",
  "title": "Discover Our Offers",
  "subtitle": "Find the best plan for you",
  "offers": [
    {
      "header_text": "Special Offer",
      "offerText": "Save 20%",
      "linkText": "Learn More",
      "linkUrl": "https://www.swisscom.ch/mobile/special"
    }
  ],
  "packages_options": [
    {
      "title": "blue Abos",
      "cards": [
        {
          "title": "Blue Premium",
          "features": {
            "Surfing": ["Unlimited 5G"],
            "Calling": ["Unlimited in CH & EU"]
          },
          "strikethroughPrice": "129.90",
          "originalPrice": "149.90",
          "discountedPrice": "119.90",
          "bestSeller": true,
          "extraData": "Details about Blue Premium"
        }
      ]
    }
  ]
}
```

Automation and Multi-Source Integration

1. Automating Daily Updates

To ensure the data is updated daily:

- **Scheduling with Cron Jobs**:
  - Use a Node.js library like `node-cron` to schedule scraping tasks.
- Example: Schedule the script to run every day at midnight.
  javascript
  CopyEdit
  `const cron = require("node-cron");`
- `cron.schedule("0 0 * * *", scrapeSwisscomMobileSubscriptions);`
  -
- **Database Integration**: Save the scraped data into a database (e.g., MySQL, MongoDB) to track changes and allow historical analysis.
- **Notification System**: Set up alerts to monitor failures or significant changes in data.

2. Multi-Source Scraping

For scraping multiple websites:

- **Modularize the Scraper**: Create separate functions for each website's specific scraping logic.
- **Data Normalization**: Standardize data structure across different sources for consistency.
- **Concurrency Management**: Use libraries like `Promise.all` or `bull` for handling concurrent scraping jobs efficiently.

3. Example Workflow

1. Fetch target URLs from a database or configuration file.
2. Loop through each URL and call its respective scraping function.
3. Aggregate the results into a unified data format.
4. Save and expose data via APIs or dashboards.

4. Conclusion

This scraper offers a robust foundation for dynamic data extraction and automation. Expanding it to support multi-source scraping and daily updates will ensure it remains scalable and efficient for larger applications.