[A] Web Scraper

In this assignment, the task is to write a web scraper that scrapes and analyzes information on some web sites built especially for this assignment. The idea is that you are going to write a scraper/agent that is designed to solve a specific problem.

You will get the main page to proceed from which links to three different web sites. You don't have to care about how they work internally, just the HTML they are rendering and how to form your HTTP request to get the data you want for analyzing.

Your starting point is https://courselab.lnu.se/scraper-site-1, which should also be the starting point in your scraping script, meaning that no more hardcoded URLs should be used in your code (except for the AJAX call in the *cinema* site). Your scraping script should also be able to handle the alternative server (see below).

A friendly hint, make sure to make the exercise "Promising Web Scraper" before moving into this assignment.

Goals for this assignment

These are the main goals for this assignment.

- Get practical experience in building a web scraper.
- Get knowledge about HTTP and use it when building an application in Node.js.
- Analyze the traffic between the client and the server.
- Get practical knowledge of asynchronous programming in Node.js.
- Analyze and solve a problem with JavaScript code.
- Using Git to show progress in your work.

Grading

This is an individual assignment.

This assignment is graded as Fail (U) or Pass (G) and it is worth 1 credit/hp.

You have access to the course environment on GitLab.

You have knowledge of programming JavaScript with Node.

Get going

Get going with the assignment by cloning and preparing the assignment repo.

Clone the assignment repo

1. You have a git repo on GitLab named "b1-scraper", available below your lnu student acronym in the course at GitLab. Start of by cloning the repo to your computer ("xxx" is your lnu-username):

```
# Using ssh
git clone git@gitlab.lnu.se:1dv528/student/xxx/b1-scraper.git

# Move into the repo
cd b1-scraper
```

Add the development tools

Setup the repo with the same development environment used. The document "Development environment" contains the details on how to do that.

Scenario

The three friends Peter, Paul, and Mary usually get together one weekend every month to see a movie and, after that, eat at a restaurant. The problem is that it is hard to plan this event since they must find a time slot when all three are available, look for a movie that plays at the cinema that day, and finally see if they can book a table at their favorite restaurant. Since all this information is available through HTTP requests it would be nice to have a script that automates this workflow!

And that's your task...

The web sites

are going to examine your scraper against another server when grading your assignment. As mentioned before, from this URL your application/web scraper should be able to crawl all three applications by itself. The scraper should be able to scrape all information, analyze it and present a solution to the user in a good way. Of course, there will be some points internally in the web sites where you will have to hardcode, but try to write it as general as possible (see examinations for more info).

The calendar web site

The first web site is where the three friends are syncing their *calendar*. Each of the friends has their page, where he/she can edit the information to let others know what day of the weekend is free. These pages are built with simple HTML and the task is to scrape the pages and analyze on what (if any) day(s) all three friends are free. The friends are only available to see each other on the weekends (Friday, Saturday, Sunday) so there is no need to handle other days.

The cinema web site

The *cinema* web site is a simple web site that displays the cinema's shows for the weekend. You can get which day and at which time a specific movie is running, and if it is fully booked or not. **By analyzing the traffic between the client and the server you should be able to find a way to request this information** and use it in your code, together with the data from the *calendar* site. Use the browser's inspector to analyze the traffic.

The restaurant web site

The third web site is the three friends' favorite *restaurant* (the only one they visit..!). To see this site, you must log in first. For this you can use the credentials below:

- username: **zeke**
- password: **coys**

The site will use session **cookies** for authorization which your application must handle in some way. After this, you can see the available booking times which you should analyze with the other data to propose a final solution.

The workflow to automate

- Check which day or days all friends are available; if none - output this on screen
- Get the available movies for that day(s)

hours after the movie starts.

- Present the solution(s) as output in your terminal/console window (or as an HTML view)

What the application should look like

Start the application passing the start URL as an argument to the process.

```
npm start https://courselab.lnu.se/scraper-site-1
```

The output of your application should look similar to this:

```
Scraping links...OK
Scraping available days...OK
Scraping showtimes...OK
Scraping possible reservations...OK

Recommendations
===============
* On Friday the movie "Keep Your Seats, Please" starts at 16:00 and there is a free table
* On Friday the movie "A Day at the Races" starts at 16:00 and there is a free table betw
```

The output should not be more "verbose" than this. Be sure to remove all your other `console.log` calls before making your release. The recommendations shown above is the correct one for the current state of the sites, and you can use it to check your solution. Be sure to test your application using the alternative server.

Using the alternative server

We have provided an alternative server where we have made some changes on the information and some URLs. Your application should also pass this server. The alternative start URL is scraper-site-2.

```
npm start https://courselab.lnu.se/scraper-site-2
```

The output should be similar to this.

```
Scraping links...OK
Scraping available days...OK
```

```
Recommendations
==============
* On Saturday the movie "Keep Your Seats, Please" starts at 18:00 and there is a free tab
* On Sunday the movie "Keep Your Seats, Please" starts at 18:00 and there is a free table
```

Requirements of your solution

1. The application should be written as a Node.js application in JavaScript using ESM modules.
2. Any external modules should be included in the `package.json` and installed through `npm install`.
3. The only command the examiner should use to run your application after cloning it from GitHub is `npm install` and `npm start` (with the starting URL as a parameter).
4. The application **should be able to take a parameter with the start URL** so one easy could change servers when running the examination.
5. You must structure your code using ESM modules.
6. Try to make a solution that is as general as possible to solve both sites. You can think like this: **The HTML structure will never be changed** but there could be changes in:

- `href` attributes in HTML: To check that your scraper doesn't use hardcoded URLs.
- URLs only defined in JavaScript code (as in the AJAX and *cinema* example) will not be changed, so you can hardcode these.
- The day(s) all three friends will be available (remember: if none, the application should give the end-user a message about that).
- The movie titles, their time and if they are fully booked or not.
- The availability of tables at the restaurant and the redirect URL we get when we log in.

Submission

This is how to submit this assignment.

2. Ensure you have committed and pushed all your changes, including the tags, to GitLab.

3. Create an issue on the repo and answer the following questions in it. Write freely with 15 to 30 sentences of text.

    1. Describe the architecture of your application. How have you structured your code with modules and what are your thoughts behind the architecture?

    2. What Node concepts are essential to learn as a new programmer, when diving into Node? What would your recommendations be to a new wanna-be-Node-programmer?

    3. Are you satsified with your application? Does it have some improvement areas? What are you especially satisfied with?

    4. Ensure that the README.md contains any essential details for grading, installing, starting, configuring your submission.

    5. What is your TIL for this course part?

4. When you are done, assign the issue to the teacher, to show that you are ready for grading. The teacher will be notified by mail.

What is a TIL? TIL is an acronym for "Today I Learned" which playfully indicates that there are always new things to learn, every day. You usually pick up things you have learned and where you might have hiked to a little extra about its usefulness or simplicity, or it was just a new lesson for the day that you want to note.

Previous
[E] Web servers