Language Idea

**Goal:** Build a web scraping language that will help automate and make web scraping more accessible and easier. It’s already been done I know but it will be fun.

**Ideation:**

IN ENGLISH, what can it do? …

* Very broadly - gather information from online sources and manipulate/store/output the data you find.
* Narrowed down slightly – write a script to get some data from a website.
* Slightly more – write “I want to find out the score of this game.”
* “I want to know what time this game starts”
* “I want to set an outlook reminder to watch this game.”
* “I want to output some game data – stats, scores, player data etc. To a file” - script will create a new folder and a .txt file inside for data output. No idea how to do this but it would be neat.
* More brain dump: If working with a single website (nfl.com) for your script you could type team->player->lastGameStats->mystats.txt -- or something like that -- this assumes our language only interfaces with nfl.com, but what the script does is treats the site like a file tree. Access the team, then look at the player on that team, then look at their statistics from the last game, then output the stats to a .txt file. At each step along the way there could be various options for what you want to do. Pick what elements in the site you want to look at I.e. images, links, tables.

Idea:   
 1. A foundation of basic scripts that would allow for common web scraping functions that can be applied to multiple different sites.

2. Libraries would be added to the language to scrape for more selective data on particular sites

Some Possible common scripts:

* Let users pick what they want to keep tabs on for a specific website. Might require the user to look at the website beforehand to know what they’re looking for so that they can write a script to monitor an element in the html or something. Check when a website is modified, ...
* Get user-specified data and save/append to a file. Script would open a new folder and .txt file and store html content of a website, for example.
* CTRL+f on tren: find words in the text content of a website, but not for one page of the site, for all subpages/relative links too.
* CSS Selector Extraction: A function to extract data using CSS selectors. This allows for flexible and consistent extraction of specific elements across different websites.
* XPath Extraction: Similarly, a function for XPath extraction can be useful for navigating and extracting data from the XML-like structure of HTML documents.
* Image Download: A function to download images from a webpage based on their URLs. This can be useful for scraping and storing images from various sources.
* Delay: Functions to introduce delays or throttling between requests to comply with website policies and avoid overloading servers.

Main thought: Be able to provide a robust foundation of tools that will allow people to build the libraries needed to do site/data specific scraping?

**The Design:**

**Components:**

1. Perl Integration:

* Utilizing Perl's efficiency in handling regular expressions, HTTP requests, and data manipulation makes it an ideal backend for web scraping. The language leverages Perl's strengths to streamline operations such as parsing HTML, extracting data, and interacting with web pages.

1. Compilation:

* The compilation process analyzes the high-level language code, identifying syntactic structures and converting them into equivalent Perl code. During this phase, the compiler optimizes the code for better performance, ensuring that the resulting Perl script faithfully represents the user's intent.

1. Bison and Flex:
   * Bison: Used for parsing the high-level language, generating an abstract syntax tree (AST) that represents the hierarchical structure of the code. The AST serves as an intermediate representation, aiding in subsequent analysis and translation into Perl code.
   * Flex: Used for lexical analysis, breaking down the code into tokens. Tokenization simplifies syntax analysis, making it easier to identify and interpret different components of the high-level language code.
2. Library integration:
   * The language supports the integration of external libraries, particularly those designed for web scraping. Users can import libraries directly into the language, expanding its capabilities and incorporating specialized functionalities for work or specific websites.

**Functionalities:**

1. Web Object Creation:

* new obj = https://www.website.com: Creates a new website object with the specified URL.
* Example Use: new myWebsite = https://www.example.com
  1. Automated Scripts:
* Enable the creation of automated scripts that run at user-specified times.
* Schedule repetitive tasks to take specific actions on the defined website object.
  1. Common Functions:
* Develop a set of basic, common functions applicable to multiple sites.
  1. Examples:
     + get HTML:
* Retrieve the HTML content of the webpage.
* HTML Parsing:
  + A generic HTML parser for extracting information from the HTML structure.
    - delay: Introduce delays between requests to comply with website policies.
    - CSS Selector Extraction:
      * Implement a function for extracting data using CSS selectors for flexible and consistent data extraction.
    - XPath Extraction:
      * Provide a function for XPath extraction, particularly useful for navigating and extracting data from HTML documents.
    - Image Download:
      * Function to download images from a webpage based on their URLs.
      * Useful for scraping and storing images from various sources.

**Tokens:**

* Assignment:
  + =: Used for assignment.
  + Example: new myWebsite = https://www.example.com
* Function Calling:
  + .: Used for calling functions on website objects.
  + Example: myWebsite.getHTML()
* Object Navigation:
  + ->: Used to navigate through the website object.
  + Example: myWebsite->team->player->lastGameStats

**Keywords:**

1. New **–** creating new website object:
   * Example use: new myWebsite = arbitraryName

**Types:**

1. Strings:
   * Strings are used for representing textual data.
   * Example:
2. integers:
   * Integers are used for representing whole numbers.
   * Example:Website objects –
3. lists/Arrays:
   * lists or arrays to store multiple values.
   * Example:
4. HTML Elements:

* Types to represent HTML elements, allowing users to navigate and manipulate the document structure.
* Example:

1. Regular Expressions:
   * Types for working with regular expressions, useful for pattern matching in scraped data.
   * Example:

ArbitraryName.getHTML("query”) arbiraryName is the object, which can be used to interface with website content. Query is optional/not needed?

**Grammar:**

**program : script\_instructions END;**

**script\_instructions : instruction | instruction script\_instructions;**

**instruction : ID ASSIGNOP WEBOBJECT**

**| WEBOBJECT FUNC1GETHTML { genCode($1); }**

**| WEBOBJECT FUNC2SECONDFUNCTION { genCode($1); }**

**| WEBOBJECT FUNC3THIRDFUNCTION { genCode($1); }**

**Built-in functions:**

1. GetHtml:
   * grabs the html of the page and returns it for work
2. HTML Parsing:
   * A generic HTML parser that can extract information from the HTML structure of a webpage. Libraries like Beautiful Soup (Python) to provide robust HTML parsing capabilities.
3. Get user-specified data and save/append to a fil:
   * Script would open a new folder and .txt file and store html content of a website, for example.
4. Image Download:
   * A function to download images from a webpage based on their URLs. This can be useful for scraping and storing images from various sources.