Lesson Design Notes Template:

Collaborative Multilingual Search and Discovery Systems

Recommendation is to make PARTS our EPISODES and EPISODES into SECTIONS

Preparation
- Description
- Target Audience
- Prereq Knowledge
- Lesson
  - Questions
  - Episodes
  - Optional Elements
  - Key Points
  - Data Set/Narrative
  - Additional Design Notes
  - Instructions for Authors

Lesson Draft
- Title
- Collaborative Multilingual Search and Discovery Systems
- About this lesson (Lesson Description)
  - What will you learn? (Lesson Learning Objectives)
  - Who is this lesson for? (Target Audience)
  - Prerequisite Knowledge
- The Lesson
  - Part 1: Ethical and Multilingual Description Standards
    - Episode 1.1
    - Episode 1.2
  - Part 2: Database and Search System Development
    - Episode 2.1 Establish Functional Requirements
    - Episode 2.2 Transform a Google Sheet into a Database
    - Episode 2.3 Connect your Website and Database
    - Episode 2.4 Display your data on the site
    - Episode 2.5 Filter your data
    - Episode 2.6 Translate your site
    - Episode 2.7 Publish your site
  - Part 3: Develop an Equitable and Reciprocal Dissemination Plan
    - Episode 3.1.
    - Congratulations on developing a website to create a search and discovery system for your project! This is an impressive achievement. In this episode, you will explore how

to expand your reach and impact. After this episode, you will be able to:

■ Institute of Museum and Library Services (IMLS)

Preparation

Description

The lesson provides librarians with practical tools they can use to help research teams create a collaborative, multilingual, and open-source search and discovery system for a research corpus. It unfolds in three sections: (1) preparing researchers' data for open and multilingual distribution, (2) constructing an online search and discovery system, and (3) drafting a distribution plan.

The lesson is informed by the development of LACLI (lacli.info), the multilingual search and discovery system for free online resources for Latin American, Caribbean, Latinx, and Iberian studies. We intend to infuse this lesson with techniques gained from building LACLI and expertise working across borders in the spirit of international open science.

Made possible by generous support by the Lessons for Librarians in Open Science program at the UCLA Library Data Science Center.

This lesson equips librarians with tools to build collaborative, multilingual search systems for research corpora. It covers data prep, system building, and distribution planning, all informed by the development of the LACLI project. Made possible by generous support by the Lessons for Librarians in Open Science program at the UCLA Library Data Science Center.

Target Audience

Add your learner profile(s) below. See: Library Carpentry Audience for some ideas about roles in the Librarian profession or Software Carpentry Learner Profiles for some other examples.

- What is their background?
  - Subject liaison, digital scholarship, or data services librarians who work with researchers in social sciences and humanities.
- What do they already know how to do?
  - Subject librarians bring expertise in their area and in languages
  - Basic HTML/CSS/JavaScript syntax
  - Working with Google Sheets
- What do they want to do with skills in this lesson?
  - Prepare research data for open and multilingual distribution
  - Construct an online search and discovery system
  - Draft a distribution plan
- What problem will the lesson help them solve?
  - Create a website that lets users search and use research data that is generated in many different places (across cultures, languages, etc.)
  - Display and share research data on a user-friendly platform

---

Prereq Knowledge

Write a list of the skills/knowledge your learners will need to follow your lesson.

- Basic experience with HTML, CSS, and JavaScript syntax.
- Basic experience with Google Sheets
- Subject area expertise in the languages and content of your research data.

Lesson

INTRODUCTION:

Setup Instructions

- Students will create a copy of LACLI Abbreviated Data Google Sheet in their Google Drive.
- Students will download the HTML and CSS files.

Librarians will be able to…

- ***SECTION 1: Formulate a description standard for open social science and humanities research data***
    - **EPISODE 1.1: Analyze and cleanup your metadata**
        - **QUESTION:** Who is your intended audience? What are their needs?
        - Use existing tutorials for general spreadsheet cleanup
        - Appraise the strengths of different existing standards
        - Understand basic descriptive metadata ethics.
        - **ACTIVITY:** Imagine one target audience includes students, teachers, and librarians in Latin America.

- ***SECTION 2: Create a single page website that displays a search and discovery system for data on a Google Sheet using Javascript*.**
    - **EPISODE 2.1: Turn your Google App Script into a Database**
        - **QUESTION:**
        - **OBJECTIVE:** Librarians will be able to transform a Google Sheet into a simple API endpoint with Google App Script
        - **EXPLANATION:** Visual explanation of what is an API and how it brings data from Google Sheets to a website.
            - Diagram of API
            - Text definition of API and JSON
            - Assessment: Multiple choice on definition of API and JSON
        - **ACTIVITY:** Apply a Google App Script to turn a Google Sheet into an API. (1) Create App Script file in Google Drive next to Google Sheet (2) Copy and paste the provided script into the file. The script will give a line by line explanation of what is happening in the script. (3) Launch the web app, which creates a URL. (4) Test this URL in a browser to see the JSON displayed. This is their API Endpoint.
        -
    - **EPISODE 2.2: Build a Search and Discovery System**
        - **QUESTION:**

        - **OBJECTIVE:** Librarians will be able to apply three Javascript functions that filter and display their Google Sheet data on a website.
            - getData(): to request data from your Google Sheet
            - filterData(): filter that data according to user supplied keywords using the .filter() function
            - displayData(): format and display the filtered data on your website using the .map function.
    - **EPISODE 2.3: Build a Page Translation System**
        - **QUESTION**
        - **OBJECTIVE:** Librarians will be able to implement a system that dynamically translates web page content based on the user's selected language using a pre-defined translations object and event listeners.
            - A nested object of 1:1 translations for page content
            - An event listener that switches among translations provided in said nested object content from the nested object.
    - **EPISODE 2.4: Publishing your Site**
        - **QUESTION**
        - **OBJECTIVE:** Librarians will be able to consult resources on how to host a site for free using GitHub pages.
- ***SECTION 3: Develop an equitable and reciprocal dissemination plan for their system that appreciates linguistic and cultural diversity.***
    - **EPISODE 3.1: Promoting your Site and Encouraging Participation**
        - Identify the audience and potential collaborators for your multilingual search and discovery systems
        - Articulate (plan) an engagement program: webinars, edit-a-thons, ask for collaboration etc. Note: *How do you bring people in? (Think about this)*
        - Implement a promotion plan for various social media platforms in various languages.

  Note: *Authentic assessment.* Participants will write an action plan to promote your site and encourage participation.

Notes: Add any relevant information about how and why you defined these objectives. This information can be helpful for future collaborators/contributors/users to understand the scope of your lesson.

Questions

Questions are displayed at the beginning of the episode to prime the learner for the content.

- Question 1
- Question 2
-

Episodes

Each lesson should have more than one episode. Each episode is a separate file in the Carpentries Workbench **For each episode**, provide the following information:
Episode Title
FIXME - add Episode Title here
Episode Learning Objectives Look at your lesson learning objectives and break them down further.
After this episode, learners will be able to:

- objective 1
- objective 2
- …
- objective N

Challenges/Exercises/Activities Provide practical tasks for learners to complete.
Challenges with Solutions Provide solutions for the challenges to facilitate discussions and deepen understanding.
Discussion Facilitate discussions around the challenges to deepen understanding.

Optional Elements

Lessons do not have to contain the following structural elements, but they can be useful for highlighting particular content. When using these, please clearly indicate what's part of the block. For their appearance, please look at the Workbench Component Guide. Use them in line with your lesson content.
Callout Blocks One of the key elements of our lessons is our callout blocks, which give learners and instructors a bold visual cue to stop and consider a caveat or exercise.
CHECKLIST Callouts Created with the checklist tag, and are a way to emphasize particular steps more strongly.
TESTIMONIAL Callouts Created with the testimonial tag and are quotations from previous learners or instructors.
SPOILER Callouts Created with the spoiler tag and are a way to provide additional details/content that can be expanded and collapsed on demand.

Key Points

These appear at the bottom of each episode and provide some summary for the episode.
[Write your key points for the lesson below]

Data Set/Narrative

Add notes about any criteria you used when choosing a data set and/or narrative for your lesson.
Considerations:

- Which datasets and narratives did you consider?
- How and why did you choose between them?
- What implications do you think your choice of dataset and/or narrative will have for designing and implementing your lesson?

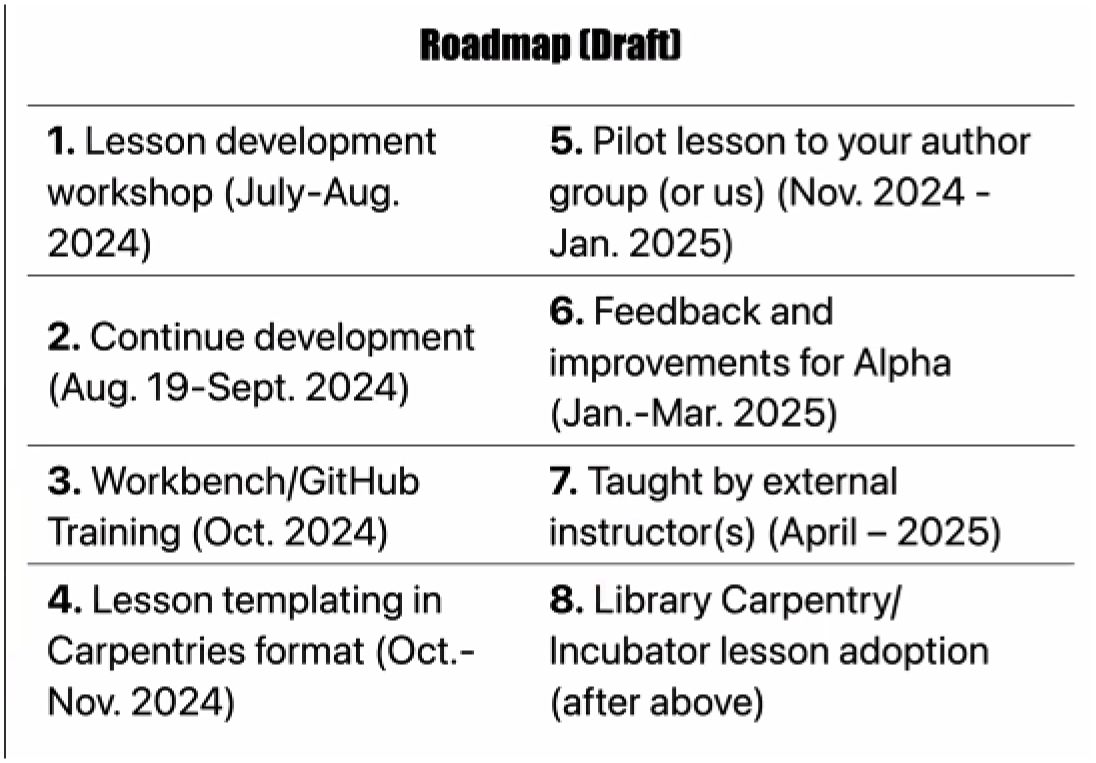
Additional Design Notes

Add notes to this section that do not fit elsewhere in the page. Topics for this section might include:

- What has been tried that did not work
- Learning objectives that you decided to remove (e.g., to trim down the content) and why
- Concept maps for all or part of your lesson (see the section below)

Instructions for Authors

1. **Lesson Title and Description:** Begin by filling in the title and a brief description of your lesson. This helps learners understand what the lesson is about and what they can expect to learn.
2. **Target Audience and Prerequisites:** Clearly define who the lesson is intended for and any prerequisite knowledge they should have. This ensures that the lesson is tailored to the appropriate audience and that learners are adequately prepared.
3. **Learning Objectives and Questions:** Use SMART criteria to write clear and achievable learning objectives. These should guide the development of your lesson content and exercises. Display questions at the beginning of each episode to prime the learners.
4. **Episode Structure:** Divide your lesson into multiple episodes, each focusing on a specific topic. For each episode, provide a title, detailed learning objectives, and associated exercises or activities.
5. **Challenges, Solutions, and Discussions:** Use practical tasks to engage learners. Provide solutions for the challenges and facilitate discussions to deepen understanding.
6. **Optional Elements:** Utilize callout blocks, checklists, testimonials, and spoilers to enhance the learning experience. Clearly label these elements to maintain structure and clarity.
7. **Key Points:** Write the key points at the end of each episode to summarize the main takeaways and reinforce learning outcomes.



Roadmap (Draft)

1. Lesson development workshop (July-Aug. 2024)	**5.** Pilot lesson to your author group (or us) (Nov. 2024 - Jan. 2025)
2. Continue development (Aug. 19-Sept. 2024)	**6.** Feedback and improvements for Alpha (Jan.-Mar. 2025)
3. Workbench/GitHub Training (Oct. 2024)	**7.** Taught by external instructor(s) (April – 2025)
4. Lesson templating in Carpentries format (Oct.-Nov. 2024)	**8.** Library Carpentry/ Incubator lesson adoption (after above)

AS OF AUGUST 12, 2024

NEXT STEPS

- ☐ Diagrams
- ☐ Code segments
- ☐ Data set
- ☐ Citations, suggested readings, etc.
- ☐ Overview section (call out blocks) for each episode (includes essential question(s) and objective(s) See example here.

Lesson Draft

Title

Collaborative Multilingual Search and Discovery Systems

About this lesson (Lesson Description)

Many research teams in the social sciences and humanities generate a wealth of materials, including text, videos, and images. Often, these teams want to share their research collections (corpora) with the public. This is where librarians play a crucial role, as they are increasingly helping researchers share their institutions' scientific knowledge with a wider audience.[1]

This lesson will guide you on how to assist research teams in creating a collaborative, multilingual, and open-source search and discovery system for their research corpora. The lesson draws on the experience of building LACLI (lacli.info), a

[1] Dempsey, L. (2016). Library collections in the life of the user: Two directions. *LIBER Quarterly*, *26*(4). https://doi.org/10.18352/lq.10170

multilingual search and discovery system for free online resources related to Latin American, Caribbean, Latinx, and Iberian studies.

The lesson begins by exploring the steps involved in getting researchers' data ready for public access and use in multiple languages. It will then move to constructing a free, online system that allows users to easily find and explore the research corpus. Finally, you'll create a plan for sharing the search and discovery system.

What will you learn? (**Lesson Learning Objectives)**

- Part 1: Ethical and Multilingual Description Standards
  - Learning Objective: Understand how a description standard ensures ethical representation of data and the importance of multilingual access.
- Part 2: Database and Search System Development
  - Learning Objective: Create a JavaScript-based website search system that allows users to effectively search and retrieve information from a Google Sheets database.
- Part 3: Dissemination Plan
  - Learning Objective: Consider linguistic and cultural diversity in the design and implementation of the dissemination plan that works toward reciprocity and collaboration in sharing research findings.

Who is this lesson for? (Target Audience)

This lesson is designed for subject liaison, digital scholarship, or data services librarians working with researchers in the social sciences and humanities. While these librarians possess domain expertise, they seek theoretical and technical guidance in creating a collaborative, multilingual search and discovery system for their research community.

Prerequisite Knowledge

A foundational understanding of HTML, CSS, and JavaScript is necessary, along with basic experience using spreadsheets like Google Sheets.

The Lesson

Part 1: Ethical and Multilingual Description Standards

In this part, we will:

- Describe the intended audience for your data, and their needs.
- Appraise the strengths of different existing standards

- Understand basic descriptive metadata ethics

Episode 1.1

In this episode we will evaluate different methods to cleanup and enhance your data, all while keeping in mind an ethical obligation to provide inclusive access to information. We will then apply these techniques to our sample data set.

QUESTION: Who is your intended audience? What are their needs?

Knowing your audience is the first step to ensure you have adequate data categories. In our spreadsheet we started with typical access points such as title, creator (institutional host), resource type, subject, language, and URL. Then we thought of some of our target audiences. What access points would students, faculty, or other librarians want to search by? So we added more detailed subject analysis with columns such as Country Coverage and Time Period.

Exercise: Imagine one target audience includes students, teachers, and librarians in Latin America. What kind of metadata might enhance access to your data? (Choose all that apply)

A. Terms in Spanish & Portuguese

B. Country Coverage

C. Resource type

Answer: A & B. While A, B, and C are all useful metadata, adding additional columns for subject access in Spanish and Portuguese provides a broader audience the ability to keyword search in multiple languages. And adding the ability to narrow by specific countries, rather than a broader Latin America or by continent, gives the user more precise searching capabilities.

POP OUT BOX:

Don’t start from scratch!

Find a cheat sheet for cleaning up spreadsheets, such as:

https://drive.google.com/file/d/15KBbdxUDpEqr7zE7P4ad2LSdweYmw4J5/view

Use existing tutorials for general spreadsheet cleanup

There are free tutorials and other tools to help you clean your data quickly. For example, Google Sheets has a built-in Smart Cleanup tool. You can find it under Data–Data Cleanup–Cleanup Suggestions. [insert screenshot]
For more info, see https://support.google.com/docs/answer/10098582?hl=en

Appraise the strengths of different existing standards

Metadata standards already exist for all kinds of data. If you can find one that matches some of your needs, you will save yourself a lot of time. One area that we recommend outsourcing when possible is the use of controlled vocabularies. Instead of coming up with your own list of terms to use consistently (and thus enhance collocation of results), see if one exists.

Exercise: The sample spreadsheet is part of our larger project featuring free online resources relating to Latin American, Caribbean, Latinx and Iberian Studies. Which of the following controlled vocabularies best fits our data?

A. Library of Congress Subject Headings (LCSH)
B. Hispanic American Periodicals Index (HAPI) Subject Headings
C. Medical Subject Headings (MeSH)

Answer: B. HAPI Subject Headings align neatly with our defined area. While based in part on LCSH, HAPI is more focused vocabulary and also has the flexibility to update their terminology without the long proposal process at the Library of Congress. Updating the names of indigenous tribes, for example, is a multi-year project for LC while HAPI can update those names more quickly.

POP OUT BOX

Don't want to create your own controlled vocabulary? Find an existing one! Search for one related to your scholarly area or start here for suggestions: https://www.getty.edu/research/publications/electronic_publications/intro_controlled_vocab/index.html

Understand basic descriptive metadata ethics.

When choosing which metadata to capture, how to organize it, and how to use it, we necessarily apply our own perspectives and experiences to those choices. Thus it is important to conscientiously work to avoid bias as much as possible. Take a step back and think about how the structure of your metadata may help or hinder access by your target audience.

For more guidance, see The Cataloging Code of Ethics:

http://hdl.handle.net/11213/16716

Statement of Internaitonal Cataloging Principles:

https://repository.ifla.org/handle/20.500.14598/80

Exercise: Imagine one target audience includes students, teachers, and librarians in Latin America. What kind of metadata might enhance access to your data? (Choose all that apply)

A. Terms in Spanish & Portuguese

B. Country Coverage

C. Resource type

Answer: A & B. While A, B, and C are all useful metadata, adding additional columns for subject access in Spanish and Portuguese provides a broader audience the ability to keyword search in multiple languages. And adding the ability to narrow by specific countries, rather than a broader Latin America or by continent, gives the user more precise searching capabilities.

At this point, go ahead and copy the `lacli-sample-data` spreadsheet to your Google Drive.

Question: There are typos that may result in inaccurate search results. How might you edit this?

Answer: use tutorials and other existing google sheet shortcuts. Filtering within a column, for example, is a fast way to pull out terms only used once, some of which may in fact be typos. [insert screenshot?]

Question: There is a problematic or outdated term for a group of people in your subject columns. How might you quickly fix a term quickly for the whole column?

Answer: Use Find & Replace [screenshot]

Part 2: Database and Search System Development

In this part we will create a single page website that displays a search and discovery system for data on a Google Sheet using Javascript.

Episode 2.1 Establish Functional Requirements

In any technical project, it's essential to establish clear functional requirements before diving into the technical details. This ensures that the team stays focused on the core objectives of the project, rather than getting sidetracked by fancy features or bells and whistles.

During the development of the LACLI project, we arrived at a series of functional requirements. Many of which are shared by digital projects in the humanities and social sciences:

Functional Requirement	**Solution**
A free, intuitive, and collaborative database solution that can be easily edited by multiple volunteers without prior database knowledge.	Google Sheets is a free, cloud-based spreadsheet solution that is intuitive to use, supports real-time collaboration by multiple users, and does not require extensive database knowledge.
Free web hosting services to store and access website files.	GitHub Pages allows you to easily publish static websites directly from your GitHub repositories and use custom URL.
The client-side technology should be open-source and avoid the use of front-end frameworks, which have a steep learning curve and can dissuade collaboration among those unfamiliar with a given framework.	Vanilla JavaScript is the core programming language of the web and is open-source. It provides for collaboration and easy maintenance as it is framework independent.
The user interface should allow users to search through the data using keywords and present resources in an attractive, easy-to-read format that includes links to the original resources.	JavaScript can make API calls, filter data, and display data on a webpage by injecting HTML into the site at run time.
The website must be translatable.	JavaScript can create a translation table inside of an object to switch the site's text content.

With this solution, the website data is edited and stored using Google Sheets. The added benefit is using Google's security via its login credentials and you can also manage access of users. Then we use JavaScript to get the data from the spreadsheet and display it on our website. The JavaScript file that does that data manipulation, as well as the HTML and CSS files that give structure and style to our site, all live in a GitHub repository and are served to the public via their GitHub Pages service.

Assessment: Based on the functional requirement assessment created above, name three best practices for writing effective functional requirements.

Episode 2.2 Transform a Google Sheet into a Database

The first technological hurtle is the fact that a Google Sheet is not automatically set up like a database. We will do this in three steps:

1. Create a Google App Script file that uses the Google Sheets API
2. Write a script in the file that wraps all data up in a JSON
3. Launch our script as a web app, which creates a unique API endpoint for our data and will permit our website to get that data and bring it to the user's browser.

An API is a set of protocols that delivers data to a website. We want our site to receive the data in our spreadsheet as JSON (JavaScript Object Notation), which is a text-based data format to transmit data objects between a web server and a client-side application. When we talk about server-side, we are talking about operations that happen in the database on the servers. When we talk about client-side, we are talking about operations that occur in the user's browser.

To activate the Google Sheet API and transform our Spreadsheet into JSON data requires creating a Google App Script file. In the Google Drive folder where you copied `lacli-sample-data` create a Google App Script File and name it `Convert Sheet to JSON`. When you open it, add the Google Sheets API under Services:

CreateAppScriptFile.mp4

Now that the Sheets API is set up, we will instruct it to wrap up our spreadsheet data in JSON.

First, in the text editing window of Google App Script, delete the default function:

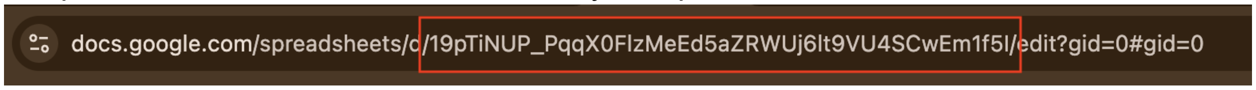
```
function myFunction() {
 }
```

We will replace it with the following code:

```
function doGet(request) {
  var spreadsheetId = '1v6xA8q23YJjS8koBD8Bq-233uaDflyETA4RXHReUZkk'; // Replace with Google Sheet ID
  var sheetName = 'Sheet1'; // Replace with the sheet you want to retrieve data from
  var sheet = SpreadsheetApp.openById(spreadsheetId).getSheetByName(sheetName);
  var dataRange = sheet.getDataRange();
  var dataValues = dataRange.getValues();

  // Change the index from 0 to 2 to use row 3 as column headings
  var headers = dataValues[2];

  var rows = [];
  for (var i = 3; i < dataValues.length; i++) { // Start from row 4, as row 3 is the column headings
    var row = {};
    for (var j = 0; j < headers.length; j++) {
      row[headers[j]] = dataValues[i][j];
    }
    rows.push(row);
  }

  var output = JSON.stringify(rows);
  return ContentService.createTextOutput(output).setMimeType(ContentService.MimeType.JSON);
}
```

Let's take this code step by step:

1. `function doGet(request)`Is a stock function in Google App Script language that handles a GET request made to your script's web app URL.
2. Your spreadsheetid is available in the URL for the spreadsheet. Copy the string of alphanumeric characters in the URL of your spreadsheet:



docs.google.com/spreadsheets/d/19pTiNUP_PqqX0FlzMeEd5aZRWUj6lt9VU4SCwEm1f5I/edit?gid=0#gid=0

3. var headers = dataValues[2], set the index of where the JSON should read header names. Because in our spreadsheet we have 2 rows of contextual information, we want the JSON to start on row 3 to find column headers.
4. Var rows = [], the script will loop over all rows to add them to the JSON.

The last step is launching the web app:
DeployWebApp.mp4

At the end of the video, you see that a red box appears around the Webapp URL. This URL will act as our API endpoint and enable our discovery system to import all the data from our Google Sheet in JSON. Take a moment to copy this URL into your browser. You'll see that now all our spreadsheet data is represented in JSON.

Keep this URL handy! We will use it soon!

Episode 2.3 Connect your Website and Database

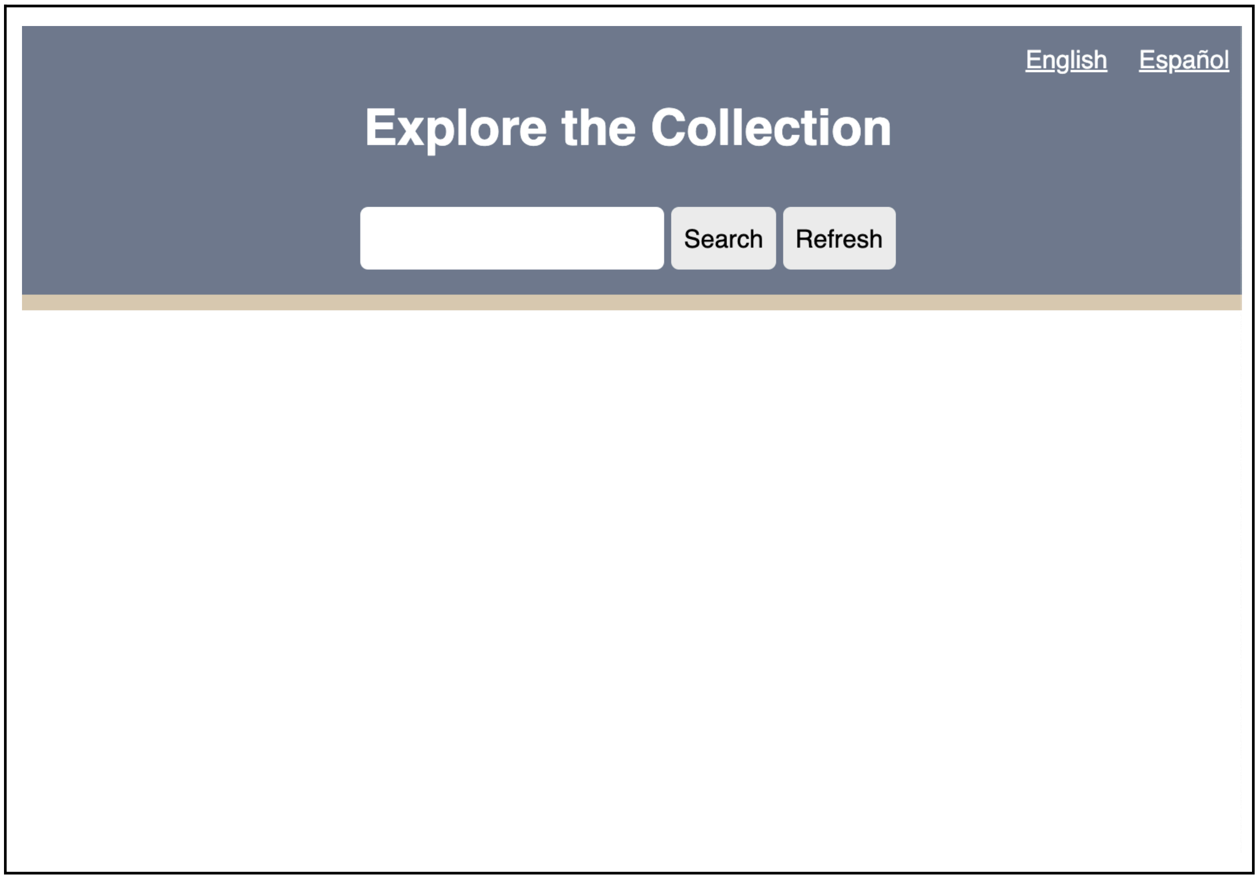
In this section we will take the API endpoint we created in episode 2.2 and connect it to our website. We'll then display that data on our website.

To get started, download CMSDS, which contains the starting files for our website. Your file structure will look like this:

```
–CMSDS
        –app.js
        –index.html
        –style.css
```

Open index.html in the browser and you will see a basic structure for our discovery system. I've already given you some basic styling. We won't be looking at style.css during this lesson, but feel free to modify it after.

From top to bottom: we have our language selection buttons, which we will use in a later episode to translate our page. Then we have our search bar, search button, and refresh button. We will use these in the next episode to enable users to search through the research data by with keywords. Last, in the blank white space is where we will populate our website with our research data.



Now open the CMSDS folder in your preferred IDE. Let's take a look at app.js, which will drive our site's functionality. Right now, we only have a road map of how we will build our system:

```
app.js

// Define API endpoint and DOM elements
const googleSheet = '';

// Get Data

// Filter Data

// Display Data
```

We will build three main functions that will, as stated in our road map, (1) get data from our Google Sheet, (2) filter that data according to a user's keywords, and then (3) display the data in a useful way on the page. In this episode we will build functions 1 and 2.

To get the data, we will use the webapp we built in Episode 2.2. In app.js, we have already defined a variable googleSheet. Let's set this to the URL of the Google webapp by copying the URL between the single quotes.

```
const googleSheet = 'https://script.google.com/macros/s/....';
```

Now we need to have app.js request that our webapp send our site the JSON data. We will do this by creating our first function, getData().

1. Create an object called apiData where we will store the incoming data.

```
let apiData = [];
```

2. Create an async function that will make a fetch request to our Google Sheet.

```
// Get data
let apiData = [];
```

```
async function getData(url) {
   const response = await fetch(url);
   const data = await response.json();
   apiData = data;
}
```

The getData function will take a URL (our Webapp) and then fetch that data and save it in a object called response. Next, we will format the response as JSON in an object called data. Lastly, we assign that formatted data from the fetch request as our apiData object.

A unique attribute of JavaScript are promises. Promises are objects that represents the eventual completion (or failure) of an asynchronous operation. Promises typically do not happen as soon as a page loads, but take some time. In our case, we assign the async keyword before declaring the getData function because we know that the request for the Google Sheet data will take some time to reach our webpage. Similarly, the await keyword is used when we make the fetch request and format our fetch response in JSON because we are anticipating that the data will not arrive when the page loads.

3. As a result of using promises, it is best practice too use the try/catch error handling block in our function in case there is some error retrieving the data. Our final getData() function should look like this:

```
// Get data
let apiData = [];

async function getData(url) {
 try {
   const response = await fetch(url);
   const data = await response.json();
   apiData = data;
 } catch (error) {
   window.alert(error.message);
 }
}
```

4. Lets test to make sure we are getting the data from our Google Sheet. We can log our incoming data to the console. Right below line 7, let's add `console.log(apiData);` Our full app.js should look like this:

```
// Define API endpoint and DOM elements
const googleSheet = 'https://script.google.com/macros/s/...';



// Get data
let apiData = [];

async function getData(url) {
 try {
   const response = await fetch(url);
   const data = await response.json();
   apiData = data;
   console.log(apiData); //Test if data fetch is working correctly
 } catch (error) {
   window.alert(error.message);
 }
};

getData(googleSheet);
```

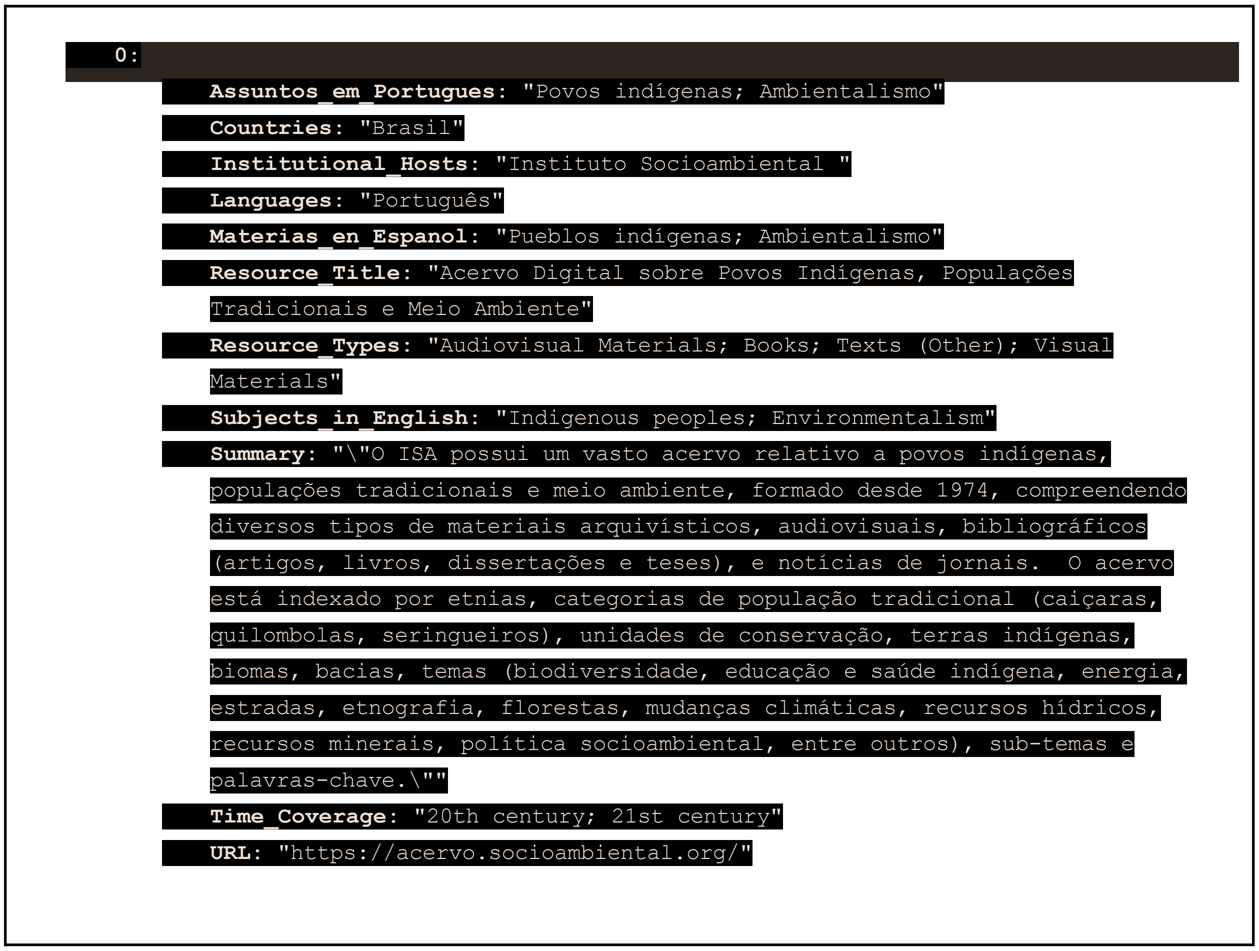
5. We now need to link our app.js file and our index.html. Open up index.html and just before the end body tag, link the app.js file:

```

   </main>
<script src="app.js"></script>
</body>
</html>
```

6. Open index.html in the browser (I suggest Chrome) and open developer tools. Then click on the console tab and you should see your data in an array:

🎬 ep3-2-devTools.mp4

We have success! Lets look at this data to get a sense of what is going on. Let's expand the first item in the array:



```
0:
        Assuntos_em_Portugues: "Povos indígenas; Ambientalismo"
        Countries: "Brasil"
        Institutional_Hosts: "Instituto Socioambiental "
        Languages: "Português"
        Materias_en_Espanol: "Pueblos indígenas; Ambientalismo"
        Resource_Title: "Acervo Digital sobre Povos Indígenas, Populações
        Tradicionais e Meio Ambiente"
        Resource_Types: "Audiovisual Materials; Books; Texts (Other); Visual
        Materials"
        Subjects_in_English: "Indigenous peoples; Environmentalism"
        Summary: "\"O ISA possui um vasto acervo relativo a povos indígenas,
        populações tradicionais e meio ambiente, formado desde 1974, compreendendo
        diversos tipos de materiais arquivísticos, audiovisuais, bibliográficos
        (artigos, livros, dissertações e teses), e notícias de jornais.  O acervo
        está indexado por etnias, categorias de população tradicional (caiçaras,
        quilombolas, seringueiros), unidades de conservação, terras indígenas,
        biomas, bacias, temas (biodiversidade, educação e saúde indígena, energia,
        estradas, etnografia, florestas, mudanças climáticas, recursos hídricos,
        recursos minerais, política socioambiental, entre outros), sub-temas e
        palavras-chave.\""
        Time_Coverage: "20th century; 21st century"
        URL: "https://acervo.socioambiental.org/"
```

JSON works in key-value format. This format will be the same for every entry in our data. All the keys are exactly the column headings in our Google Sheet. This is very useful because we can call specific pieces of data about a resource in our data table by using the column headings. The values, presented within quotes, is the data per row of our data. So here we are look at all the data, for the first row in our table. So, for example, if we want to display the title of a resource in our data table, we can ask JavaScript to show us the Resource_Title for a specific row.

Episode 2.4 Display your data on the site

Now let's move to displaying this data on the website in a user-friendly format. In our index.html file, you will see we have a div element with the id of "display" within the <main> part of our document.

```
index.html
   <main>
       <div id="display"></div>
   </main>
```

To display our data, we will target this div in our app.js file and dynamically inject html containing our data here. To target this div, we will get this element by its id in app.js:

```
app.js
const display = document.getElementById('display');
```

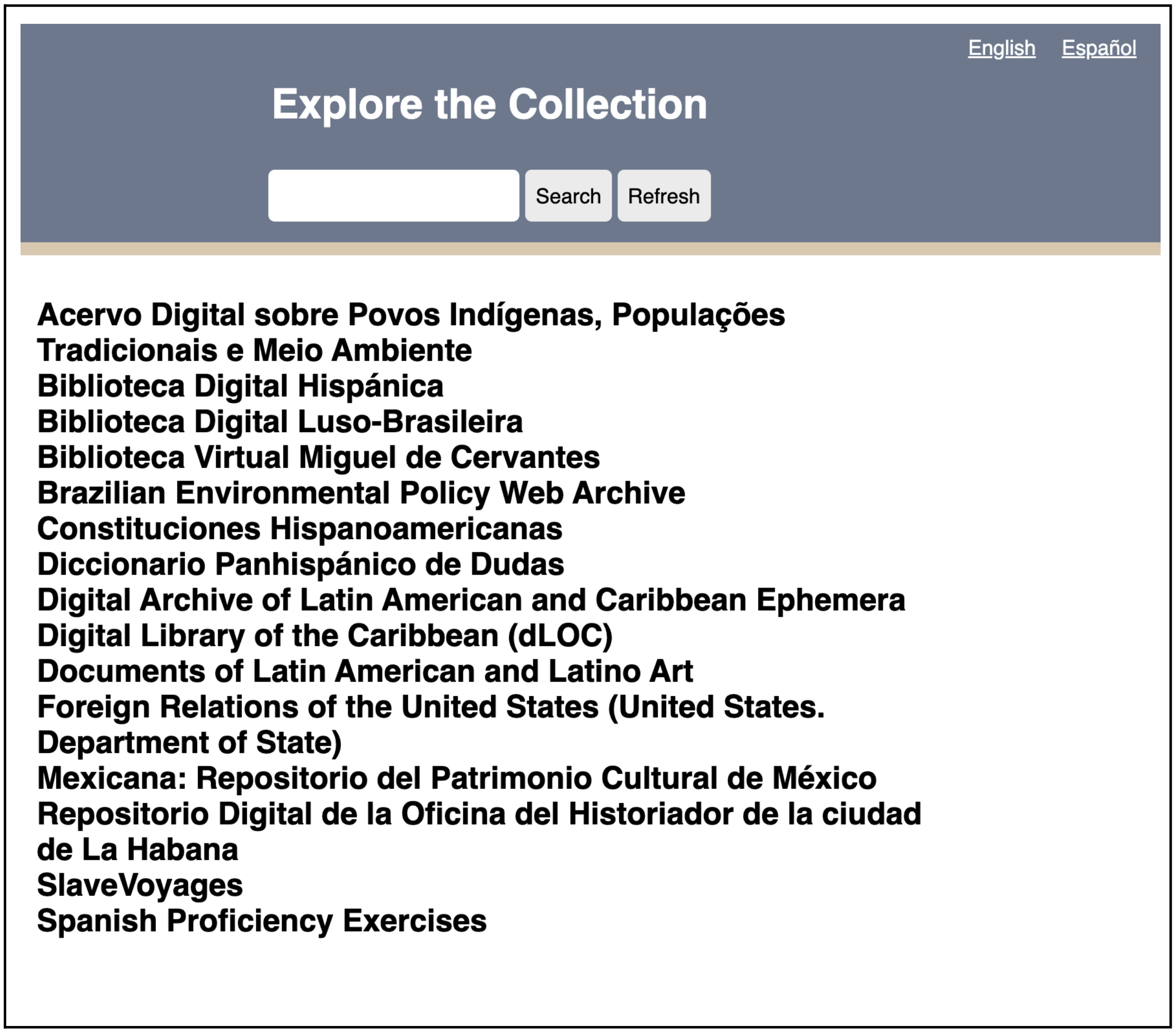
Now that it's targeted, we can control the html injection with a new function in app.js called displayData():

```
app.js

const display = document.getElementById('display');

// Display data
function displayData(data) {
 let dataDisplay = data.map((object) => {
   return `
<h2>${object.Resource_Title}</h2>
`;
 }).join('');

 display.innerHTML = dataDisplay;
};
```

This function takes our data as input. It then creates an object called dataDisplay where we will store our formatted data for display to the user. The key part of displayData() is the use of the .map() method. In Javascript, .map() creates a new array from an existing array by applying a provided function to each element in the existing array. In our project, .map will take our data array, a create a new array, dataDisplay, that will take each object and format it in html, ready to be injected into index.html, which occurs in the last line of the code above, `display.innerHTML = dataDisplay;`

You can see what we are iterating over all the data and placing each object's Resource_Title within <h2> tags. The map function makes it easy to specify where you'd like what piece of data within the html by calling: object.<insert column name from spreadsheet>. Here we are using object.Resource_Title.

Open index.html in the browser and you will see the Resource_Title of each object in our data array now populating the formerly blank white space below the search bar.



Acervo Digital sobre Povos Indígenas, Populações Tradicionais e Meio Ambiente
Biblioteca Digital Hispánica
Biblioteca Digital Luso-Brasileira
Biblioteca Virtual Miguel de Cervantes
Brazilian Environmental Policy Web Archive
Constituciones Hispanoamericanas
Diccionario Panhispánico de Dudas
Digital Archive of Latin American and Caribbean Ephemera
Digital Library of the Caribbean (dLOC)
Documents of Latin American and Latino Art
Foreign Relations of the United States (United States. Department of State)
Mexicana: Repositorio del Patrimonio Cultural de México
Repositorio Digital de la Oficina del Historiador de la ciudad de La Habana
SlaveVoyages
Spanish Proficiency Exercises

In our `lacli-sample-data` spreadsheet, we have a lot of other data about each resource that is useful to users:

- Resource_Types
- Languages

- Countries
- URL
- Subjects_in_English
- Materias_en_Espanol
- Assuntos_em_Portugues
- Time_Coverage
- Institutional_Hosts
- Summary

Thus the hierarchy we apply to this data when we populate it on our website should reflect some kind of logic. For example, the Resource_Title should serve as a URL link to the resource. Below the title, we should have the Institutional_Hosts. The rest of the data is there to describe the resource so that a user can quickly evaluate whether they are interested in a given resource. Here is one possible arrangement of this data in html:

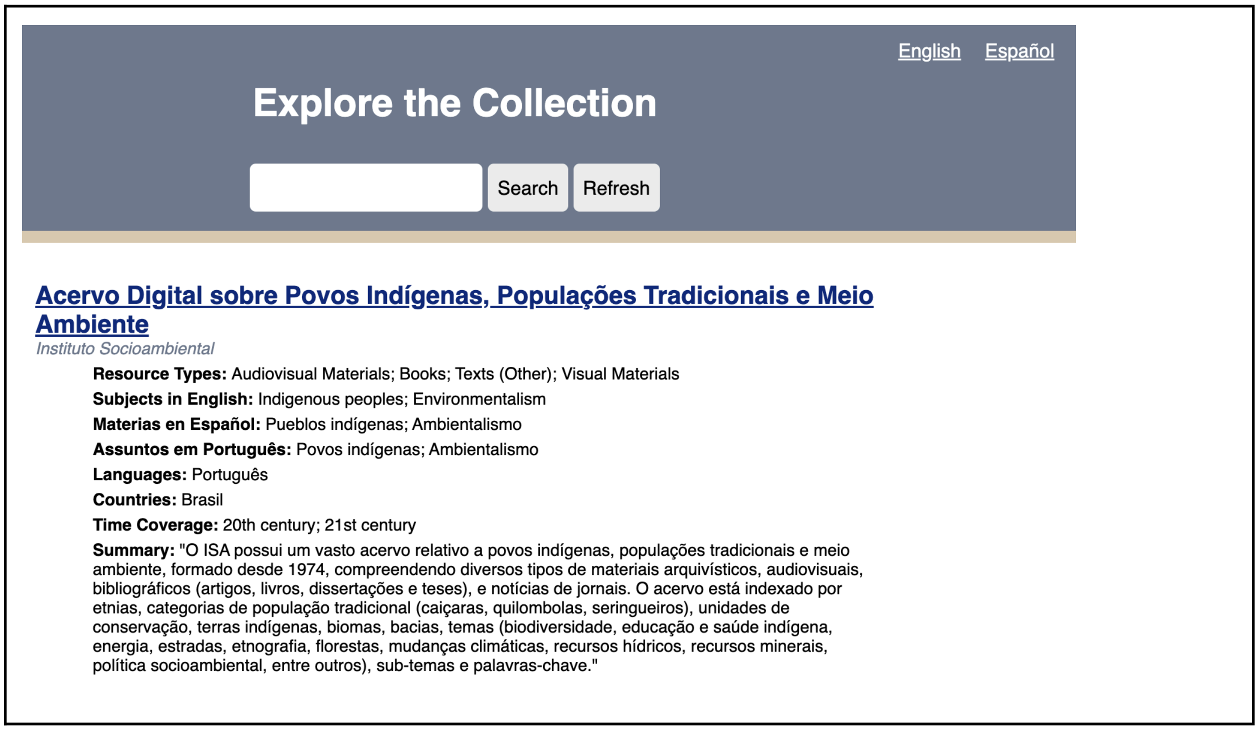
```
<article class="item">
   <div class="item-header">
      <h2><a href="${object.URL}" target="_blank" rel="noopener
noreferrer">${object.Resource_Title}</a></h2>
      <p>${object.Institutional_Hosts}</p>
   </div>
   <div class="item-description">
      <p><span class="inline-label">Resource Types: </span>${object.Resource_Types}</p>
      <p><span class="inline-label">Subjects in English:
</span>${object.Subjects_in_English}</p>
      <p><span class="inline-label">Materias en Español:
</span>${object.Materias_en_Espanol}</p>
      <p><span class="inline-label">Assuntos em Português:
</span>${object.Assuntos_em_Portugues}</p>
      <p><span class="inline-label">Languages: </span>${object.Languages}</p>
      <p><span class="inline-label">Countries: </span>${object.Countries}</p>
      <p><span class="inline-label">Time Coverage: </span>${object.Time_Coverage}</p>
      <p><span class="inline-label">Summary: </span>${object.Summary}</p>
   </div>
</article>
```

As we discussed above when we were just applying object.Resource_Title, here you can see that we can call all the different data elements per resource with this same notation. Want to call the language of the resource? object.Languages. Materias en Español? object.Materias_en_Espanol. Again, all borrowing the column headings from our Google Sheet. We wrap each resource in an article tag as it defines independent content and thus improves accessibility by providing clear structure for assistive technologies.

We can copy this html into our displayData() function like this:

```
app.js

const display = document.getElementById('display');

// Display data
function displayData(data) {
 let dataDisplay = data.map((object) => {
   return `
<article class="item">
   <div class="item-header">
       <h2><a href="${object.URL}" target="_blank" rel="noopener
noreferrer">${object.Resource_Title}</a></h2>
       <p>${object.Institutional_Hosts}</p>
   </div>
   <div class="item-description">
       <p><span class="inline-label">Resource Types:
</span>${object.Resource_Types}</p>
       <p><span class="inline-label">Subjects in English:
</span>${object.Subjects_in_English}</p>
       <p><span class="inline-label">Materias en Español:
</span>${object.Materias_en_Espanol}</p>
       <p><span class="inline-label">Assuntos em Português:
</span>${object.Assuntos_em_Portugues}</p>
       <p><span class="inline-label">Languages: </span>${object.Languages}</p>
       <p><span class="inline-label">Countries: </span>${object.Countries}</p>
       <p><span class="inline-label">Time Coverage:
</span>${object.Time_Coverage}</p>
       <p><span class="inline-label">Summary: </span>${object.Summary}</p>
   </div>
</article>
   `;
 }).join('');

 display.innerHTML = dataDisplay;
};
```

The CSS I've already provided will handle the formatting of our data, to aid in readability and reinforce our hierarchy of information.



Explore the Collection

Acervo Digital sobre Povos Indígenas, Populações Tradicionais e Meio Ambiente

Instituto Socioambiental

Resource Types: Audiovisual Materials; Books; Texts (Other); Visual Materials

Subjects in English: Indigenous peoples; Environmentalism

Materias en Español: Pueblos indígenas; Ambientalismo

Assuntos em Português: Povos indígenas; Ambientalismo

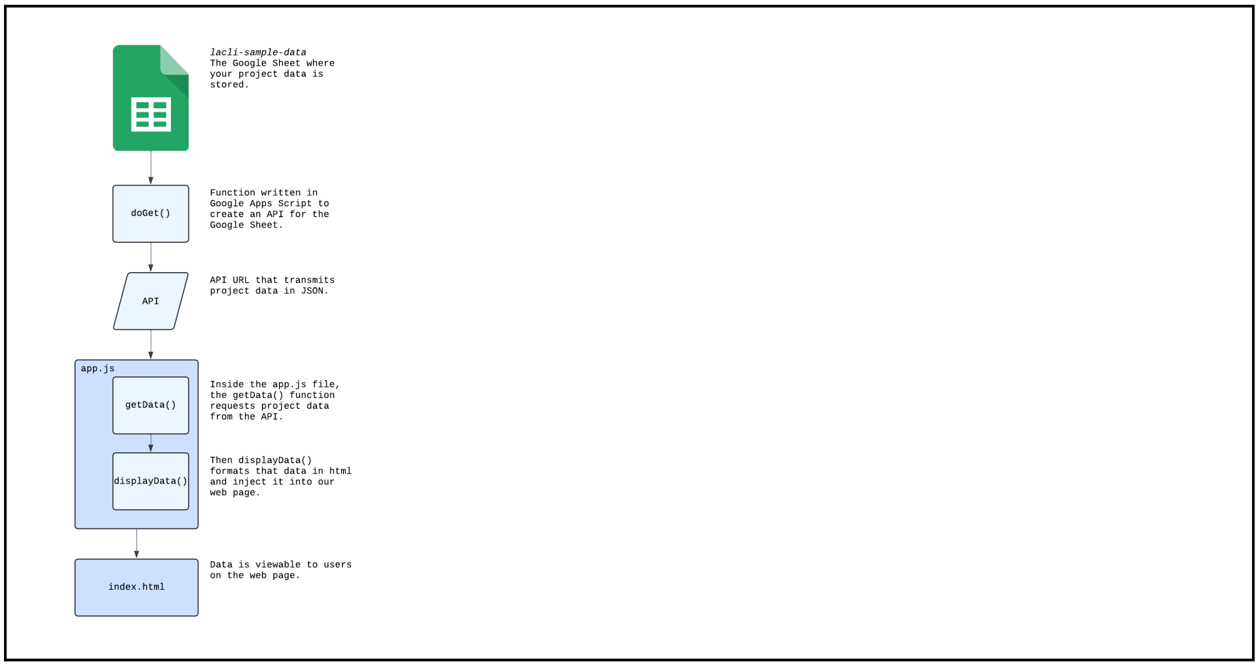
Languages: Português

Countries: Brasil

Time Coverage: 20th century; 21st century

Summary: "O ISA possui um vasto acervo relativo a povos indígenas, populações tradicionais e meio ambiente, formado desde 1974, compreendendo diversos tipos de materiais arquivísticos, audiovisuais, bibliográficos (artigos, livros, dissertações e teses), e notícias de jornais. O acervo está indexado por etnias, categorias de população tradicional (caiçaras, quilombolas, seringueiros), unidades de conservação, terras indígenas, biomas, bacias, temas (biodiversidade, educação e saúde indígena, energia, estradas, etnografia, florestas, mudanças climáticas, recursos hídricos, recursos minerais, política socioambiental, entre outros), sub-temas e palavras-chave."

Here is a recap of what we've accomplished so far:


lacli-sample-data
The Google Sheet where your project data is stored.
doGet()
Function written in Google Apps Script to create an API for the Google Sheet.
API
API URL that transmits project data in JSON.
app.js
getData()
Inside the app.js file, the getData() function requests project data from the API.
displayData()
Then displayData() formats that data in html and inject it into our web page.
index.html
Data is viewable to users on the web page.

Episode 2.5 Filter your data

Up to this point, we've focused on connecting our site to our Google Sheet API and displaying that data in the browser: we haven't touched the search functionality. To call this functionality searching very much anthropomorphizes the code: humans search using a variety of emotion, reason, and what we might call algorithms to sift through information to determine what is most relevant at a given time. What our will do is much simpler and a better way to describe it is filtering the data. We will filter the data according to whether the data we have describing a resource matches or does not match the user's keywords. For example, if a user enters the keyword "Periodicals," we want to return all resources whose Resource_Types include "periodicals."

Filtering the data will occur after the browser has received data from the Google Sheet API, the getData() function, and before that data is displayed on the website to the user, the displayData() function.

It is very important to remember that after getData() is called, all of the data is loaded into the user's browser. This means that once the page loads, and the getData() promise is resolved, our website will not request data again from our Google Sheet API. All our filtering will occur on the client-side, which is to say in the browser and not in our Google Sheet database. This organization helps us achieve the Functional Requirements we established in Episode 2.1 and has a few benefits:

1. After getData() loads the data, filtering data is almost instantaneous, because we do not need to keep request new data from the Google Sheets API.
2. As laid out in the Functional Requirements in Episode 2.1, we want free database and web hosting solutions. Limiting our API requests ensures that we can continue with Google Sheets as our simple database.
3. Security: we want a simple web solution, and a single get request from the API excludes any two way communication with our server, such as Post requests or the ability of users to inject malicious code back into our database. Our system is very simple, and this simplicity is one of its strengths.

Let's take a look at our search bar in our html:

```
index.html

<div class="search-bar">
               <input id="input" autocomplete="off" type="text" aria-label="Enter
keywords to search">
               <button id="search-btn" aria-label="Run search
button">Search</button>
```

```
               <button id="refresh-btn" title="Refresh" aria-label="Refresh search
results button">Refresh</button>
</div>
```

In html, search bars are created using <input> elements. Ours additionally has the id of “input.” We also have a search button, this is the first <button> element with an id of “search-btn.” We need to capture these two elements in our app.js file in order to process a user’s search.

Switching to app.js, let’s target these two html elements:

```
app.js

// Define API endpoint and DOM elements
const googleSheet = '<your Google Sheet API here>';
const display = document.getElementById('display');
const input = document.getElementById('input');
const searchBtn = document.getElementById('search-btn');
```

The first goal is to capture a user’s keywords in the search bar.

1. Let’s create a function called runSearch() that will process a user’s keywords. searchTerms will get us the value inside the <input> html element as well as trim off any incidental white space at the beginning or end of the user’s keywords. For now, we will console.log the user’s keywords.

```
app.js

// Filter data
function runSearch() {
 const searchTerms = input.value.trim();
 console.log(searchTerms)
}
```

2. Now we need to create an event listener that will trigger the runSearch(). An event listener in JavaScript is a function that is triggered when a user interacts with an HTML element. In our

case, we want to trigger runSearch() when the user hits the return key or clicks the search button.

```
app.js

// Event listeners for search bar
// When the user clicks the search button
searchBtn.addEventListener('click', runSearch);
// When the user presses the return key in the search window
input.addEventListener('keypress', (event) => {
   if (event.key === 'Enter') {
       runSearch();
   }
});
```

🎬 ep2-5.mp4

In this video you can see that when there is text in the search bar and the user either hits the return key or presses the search button, our runSearch() function is activated and in turn logs the text in the console.

Now we can create our filterData() function. First, create a function called filterData() and have it accept as an argument the user's query. We capture this query in the runSearch() function. We want to first have some logic here. We want to check whether or not a user even has a keyword in the search bar. Maybe they hit the search button without entering any keywords.

```
app.js

function filterData(query) {
 if (query) {
 // Our filter function will go here

 } else {
   displayData(apiData);
 }
};
```

In this case, if the user does provide a query, our function will do something with that query to filter our data. If the user does not provide a query, our function does nothing and passes along the full dataset from our API to our displayData() function.

We need to consider the variety of ways our users might search for material.

1. **Capitalization:** Some users may capitalize words differently. They might use CAPSLOCK or all lowercase or Title Case Their Searches. Our filter function needs to be ready to handle all of these cases.

2. **Word Order:** We want to search each keyword as if it were a unique search term as opposed to searching with one long string. For example, if we search as one string, and a user searches “art in mexico” or “mexico art,” the filter function will only return results that also have the words in the same order. What the user wants is all resources that mention art *and* mexico.

3. **Diacritics:** Diacritics are the accent marks used in Spanish. Given that we are looking at a bilingual database and expect Spanish speaking users, we need to manage diacritics. It is best to make our filtering diacritic agnostic. For example, someone could misspell Pueblos indígenas as Pueblos indigenas.

The process to manage all the variety of ways users might enter search terms is called normalization. We will normalize all search input to prevent our filter function from excluding results because of capitalized letters, word order, or missing diacritics.

JavaScript makes normalizing for capitalization and word order easy. Let’s return to our filterData() function, and add this code if a user supplies a query:

```
app.js

function filterData(query) {
 if (query) {
     const searchTerms = query.toLowerCase().split(/\s+/)

 } else {
   displayData(apiData);
 }
};
```

We created a new variable called searchTerms. Inside this variable, we will first apply the .toLowerCase() method which makes the user’s input keywords in lowercase and should eliminate any problems around mismatched capitalization. Then we use the .split() method to split up the user’s input on each space (`/\s+/`) which is a regular expression that identifies spaces and creates a new search term on the space. In the end, searchTerms will be an array of search terms that we will use to filter our data.

Diacritics are require an additional step, we must create our own function that will strip the diacritics from the user’s keywords:

```
app.js

function removeDiacritics(str) {
 return str.normalize("NFD").replace(/[\u0300-\u036f]/g, "");
}
```

This function takes a string, the user’s input, and applies .normalize("NFD"). This command normalizes the input string using the "Normalization Form D" (NFD) algorithm, which separates base characters and diacritic markings. For example, "é" becomes "e" and ‘. Then the .replace() method replaces all diacritic marks with an empty string. The g flag ensures that all markings within the string are replaced.

Now let’s add this removeDiacritics function and finish normalizing the user’s input:

```
app.js

function filterData(query) {
 if (query) {
     const searchTerms = query.toLowerCase().split(/\s+/).map(term =>
removeDiacritics(term));

 } else {
   displayData(apiData);
 }
};
```

The .map() function is very useful in JavaScript. This function in JavaScript creates a new array by applying a provided function to each element of the original array. That's important because remember that searchTerms is actually an array of every term, separated by spaces, from the user’s input. We then use the arrow function, which takes each term in the array and applies the removeDiacritics function on that argument.

Now that we've normalized the user's search terms, we can use it to filter the data. We will use the JavaScript filter() function to return only those results in our data that match the user's search keywords:

```
app.js

// Filter data
function filterData(query) {
 if (query) {
   const searchTerms = query.toLowerCase().split(/\s+/).map(term =>
removeDiacritics(term));

   const filteredData = apiData.filter(allData => {
     return searchTerms.every(term => {
       return (
         Object.values(allData).some(value => {
           if (value && typeof value === 'string') {
             return removeDiacritics(value.toLowerCase()).includes(term);
           }
           return false;
         })
       );
     });
   });

   displayData(filteredData);
 } else {
   displayData(apiData);
 }
};

     });
   });

   displayData(filteredData);
 } else {
   displayData(apiData);
 }
}
```

We use three important methods to conduct the search: filter(), every(), and some().
1. **Filter()** We use to iterate over every object in our API data to determine if it matches with the user's search keywords.
2. **Every()** Iterates over each search term in the searchTerms array. If all search terms match, the every() method returns true, indicating that the object should be included in the filtered results.

3. **Some()** Here is where the real matching occurs. It is iterating over all the values for each object in the array. We can think of each object in the array of API data as one resource in our dataset. The values of the object are those we have in our dataset per resource. For example, Resource_Title, Materias_en_Espanol, Languages, and Summary. The Some() function will take each search term and see if it matches within the values of an object and returns a boolean true or false. All search terms be true meaning that a resource must contain all the search terms in order for it to be included in the filteredData object. Within this some function, we include a bit of logic and processing on the API data. First, we check that the data is a string, this way we can match the string of the user's input to the string in the data. If it is a string, we apply the same normalization of capitalization and diacritics we applied to the keywords to the data. This way we remove any possibility for a missed match due to differences in capitalization or diacritic placement. In other words, we compare lowercase, diacritic-free search terms with lowercase, diacritic-free data.

In the end, we will pass our filteredData array to our displayData() function to display the search results to our user.

Now we are ready to test it. We need to make a small change to our runSearch() function, which at this time is logging our searchTerms to the console. We want to now pass the user's supplied keywords to our new filterData() function like this:

```
app.js

// Filter data
function runSearch() {
 const searchTerms = input.value.trim();
 filterData(searchTerms);
}
```

The last feature we need to add is a refresh button. This button will take users back to the full API dataset without forcing them to refresh the page.

Let's identify the refresh button in app.js:

```
app.js

const refreshBtn = document.getElementById('refresh-btn');
```

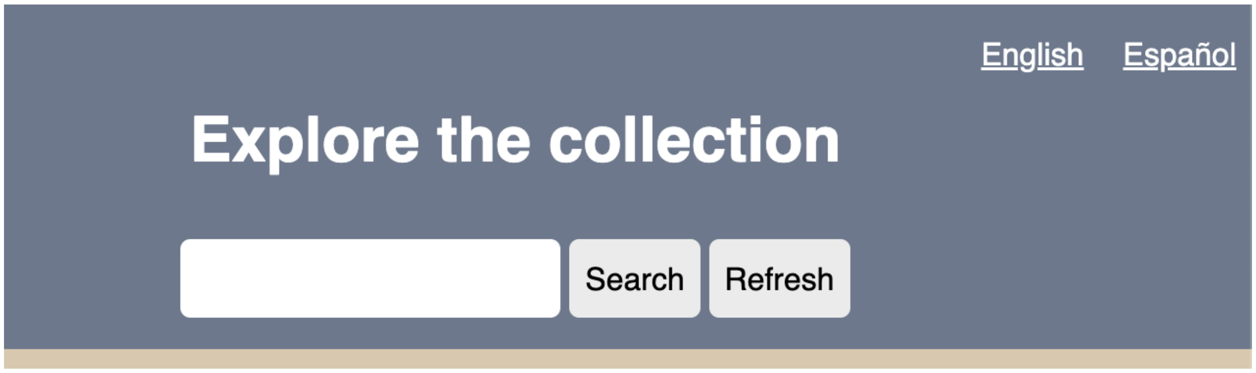
This button will clear all user keywords so they see all resources with no filtering. We will do this by running the runSearch() function with no user keywords.

```
app.js

refreshBtn.addEventListener('click', () => {
 input.value = '';
 runSearch();
});
```

Episode 2.6 Translate your site

Translation remains an elusive problem on the web. Browsers permit the use of 3rd party tools like Google Translate to translate a web page on the fly. There are also subscription AI tools like Transifex. Translation has come a long way and it is now very good and enable users to access content from a variety of languages. But language is complicated by regionalisms, varieties, and problems between literal and figurative speech. At times, AI translation services miss these subtleties. JavaScript enables us a straightforward way to customize our translation for users and be sure that we have full control of the content user's see.

In this episode we will focus on setting up our language selection buttons in the top right corner of our page.



We will first create a new file called translation.js where we will create our language selection app. Your file structure should look like this:

```
–CMSDS
        –app.js
        –index.html
        –style.css
        –translation.js
```

Next we will need to link this new JavaScript file with our webpage. Open up index.html and add a link using a <script> tag at the bottom of the file:

```
index.html


[...]
<main>
       <div id="display"></div>
   </main>
<script src="translation.js"></script>
<script src="app.js"></script>
</body>
</html>
```

Within our index.html file, we should also examine the structure of the language selector:

```
index.html

[...]
<div class="translation">
      <a href="" data-lang="en">English</a>
      <a href="" data-lang="es">Español</a>
</div>
```

We see that these are actually anchor elements <a>. We have left the href empty because these links will not direct users to a new page, but instead will trigger our translation.js file. Also, we have this custom element data-lang, which we will use to cue our translation.js script, through the two-character language codes en and es, as to what language the user prefers to read in.

Next, lets take a look at the HTML elements we want to translate:

```
index.html

[...]
<h1 id="welcome-msg">Explore the Collection</h1>
           <div class="search-bar">
               <input id="input" autocomplete="off" type="text" aria-label="Enter
keywords to search">
               <button id="search-btn" aria-label="Run search
button">Search</button>
               <button id="refresh-btn" title="Refresh" aria-label="Refresh search
results button">Refresh</button>
           </div>
```

We have our welcome message in the <h1>, search button, and refresh button. All of these elements have an id element that will let us target and translate them in translation.js.

Now let's switch over to translation.js. We will first target all three of these elements:

```
translation.js

const welcomeMsg = document.getElementById('welcome-msg');
const searchButton = document.getElementById('search-btn');
const refreshButton = document.getElementById('refresh-btn');
```

Translation using JavaScript works by creating a 1:1 translation table. When our script receives the language code, en or es, it will then look what text to display for the elements we just targeted above. We will create an object called translations that has a set of content for English (en) and Spanish (es).

```
translation.js

const translations = {
   en: {
       welcomeMsg: 'Explore the collection',
       searchBtn: 'Search',
       refreshBtn: 'Refresh',
   },
   es: {
       welcomeMsg: 'Explorar la colección',
       searchBtn: 'Buscar',
       refreshBtn: 'Recargar',
   }
};
```

As you can see, we reference the element we want to target, for example welcomeMSG. And then under each language, we put the content we want to display.

Now we will create a function that will execute our translation:

```
translate.js

let currentLanguage = 'en'; // Default language

function translatePage(language) {
   currentLanguage = language;
   // Update text content for the specified language
   welcomeMsg.textContent = translations[language].welcomeMsg;
   searchButton.textContent = translations[language].searchBtn;
   refreshButton.textContent = translations[language].refreshBtn;
}

// Initial translation based on the default language
translatePage(currentLanguage);
```

We start by setting our default language to English (en). Then we create the function translatePage(), which takes as an argument the two-character language code. Then for each element on our page we want to translate, JavaScript first manipulates the textContent in the HTML by looking up in the translations object which nest table of translations to use. It then within that nest table for either en or es, finds the prescribed content to push to our HTML.

Finally, we need to create an event listener that will pass the two-character language code to our translatePage() function when a user clicks on their preferred language:

```
translation.js

// Handle language selection from translation language links
document.querySelector('.translation').addEventListener('click', function (event) {
   const selectedLanguage = event.target.getAttribute('data-lang');
   if (selectedLanguage) {
       event.preventDefault(); // Prevent default page reload
       translatePage(selectedLanguage);
   }
});
```

When a user clicks on a language link, the script will retrieve the two-character code listed by the data-lang custom element. event.preventDefault() will prevent the <a> tag from carrying out its normal link action and refreshing the page. If it refreshes the page, it will cause us to have to reload the API data, which will make our site a much slower experience for users. Finally, we pass the two-character language code to translatePage() function.

Episode 2.7 Publish your site

We will host our project online for free using GitHub Pages. You will need to create a GitHub account.

Is there already a Carpentries episode on this?

Part 3: Develop an Action Plan to Promote Your Project and Encourage Participation and Collaboration

Episode 3.1.

Congratulations on developing a search and discovery system for data stored in a Google Sheet! This is an impressive achievement. In this episode, you will explore how to expand the impact of your project and encourage participation and collaboration. After this episode, you will be able to:

OBJECTIVES

1. Identify and evaluate effective strategies to promote and increase visibility of your project among your diverse and global potential users.
2. Learn how to encourage participation in your project and nurture collaborations that enhance its multilingual functionality, cultural diversity, and global usefulness.
3. Develop an action plan to promote your project and encourage participation and collaboration.

Let's begin by exploring various strategies to increase the visibility of your project. Think about how you bring people in! How can you effectively promote your website to reach potential users who might find it valuable? While not all of these possibilities may be directly applicable to your specific situation, it's valuable to consider a wide range of options. Use the following checklist to help you identify the most effective ways to promote your project.

1. Create a variety of content about your project and encourage others to reuse your data and link to your project. Some examples:

- Write a wikipedia entry.
- Allow reuse of your data by selecting a creative commons license appropriate to your work https://chooser-beta.creativecommons.org/ Also, share your data with similar projects so that you can reuse their content and they can reuse yours.
- Encourage librarians to link to your project in:
    - E-resources/database finder platforms. Example: Library of Congress E-resources Platform
    - Library Guides. You can track those created using LibGuides, a content management and information sharing system, in this website: https://community.libguides.com/

    - Encourage other library projects or initiatives to include a metadata record for your project. Example: project LACLI in the Handbook of Latin American Studies and Resources for College Libraries (this database requires a subscription).

2. Disseminate information about your project via listservs of professional organizations (both library and subject related). In the case of our project LACLI, we promote it in organizations such as the Seminar on the Acquisition of Latin American Library Materials (SALALM), sections of the Latin American Studies Association (LASA), the ACRL's European Studies Section (WESS), and the National Association to Promote Library and Information Services to Latinos and the Spanish-Speaking (REFORMA). In addition to national and international organizations, consider promoting it in regional and local organizations as well.

3. Build an effective social media strategy.

- Create social media accounts/profiles in those social media services you consider more effective for your intended audience. At the beginning of your social media strategy, you might need to explore a wider net of social media platforms and narrow them down based on their analytics. For example, **Linkedin** might be a good option to target professionals across various industries and career stages interested in networking. **X** (formerly known as Twitter) can be proven effective targeting journalists, media professionals and political activists. Also consider age groups. For example, **Instagram** is particularly popular among younger users in the 18-29 range; **TikTok** is heavily used by younger Millennials. Consider the potential of others such as **Reddit** that is known for its diverse communities and discussions. **Reddit** is popular among tech-savvy users in their 20s.
- Follow organizations and people related to the subject of your project. Make sure you tag them when you create content related to something of their interest or something that has been created by them. This is a low-maintenance way to start building connections with them by helping promote their own content or share with them related content connected to their interests. For example, in LACLI we create weekly posts featuring sources from our database. If we promote a source created by an organization, we tag it along with tags to potential individual users interested in it. This also expands the visibility of your posts to anyone that follows the account you are tagging.
- Be strategic about the context, nature of the content and when you post it. For example, in LACLI we have featured content related to historical commemorations or celebrations. During the "back to school" period, we have promoted free OER resources. Make sure you use hashtags to increase the visibility of your posts. Also, use at least one hashtag that can bring all the posts related to a specific theme or focus together. For example, in LACLI we use the hashtag #LACLIresources to bring all the resources that we feature together. In terms of content, explore various tones depending on your audience but keep them short and to the point. In the case of LACLI, we focus on the value and uniqueness of each featured resource. When crafting your posts, carefully consider two key language aspects:
    1. Choice of language: Decide whether to write in English, Spanish, or other languages based on your target audience.

        2. Writing style: Adapt your tone and vocabulary to suit your specific readers, whether they're academics, high school teachers, or other groups.
- Promote your project by using hashtags such as #DataScience or #TidyTuesday so others can contribute content to your project or reuse it to create other digital objects such as visualizations.
- Be aware of the limitations of social media. When you rely solely on social media platforms, you're not necessarily casting the widest possible net for your audience. While some individuals engage deeply with content and actively participate in discussions, many others interact differently. Furthermore, numerous individuals confine their social media use strictly to personal matters, avoiding professional or project-related content entirely. A significant portion of users tend to passively consume content. This passive consumption can limit the meaningful interactions and collaborations you might be hoping to achieve for your project.

4. Use Analytics tools to figure out aspects such as:
- Best day and time to post your content
- Visitor metrics
- Type of content with more views and reactions

Google Analytics is the tool we use to monitor how people interact with the LACLI website. This tool provides various metrics, but we find the information about users' geographical locations especially valuable. This data helps us tailor our outreach strategies more effectively. Also take advantage of the built-in analytics tools offered by various social media platforms.

You can also use AI tools to undertake additional types of analysis. For example, if you want to compare your project with similar projects, you could use this prompt: Compare [name of your project] to other similar online repositories specialized in [specify your subject(s)] Always ask follow-up questions when using AI tools. This helps you get more accurate and relevant answers.

5. Apply for awards and grants that could bring both recognition and financial support.

For this type of content, you'll find several specialized databases that can be particularly helpful:
- **Subscription-based** (it might include some partial access under some circumstances):
    - Foundation Directory Online (FDO)
    - Pivot-RP
- **Free access**:
    - Grants.gov

Consider applying to awards and grants by organizations that support digital projects such as:
- **International organizations** such as the Global Sustainability Coalition for Open Science Services (Scoss), a network of organizations committed to helping secure OA and OS infrastructure. It provides funding for open infrastructure https://scoss.org/need-funding-for-open-infra/

- **National organizations**. The ones below are US-based. Think of any available in the country where you develop your project.
  - National Endowment for the Humanities (NEH). The NEH Office of Digital Humanities offers several grant programs specifically for digital projects. These grants support various stages of digital projects, from early development to implementation and evaluation.
  - Institute of Museum and Library Services (IMLS)
  - Mellon Foundation
  - American Council of Learned Societies (ACLS). Digital Extension Grants
  - Council on Library and Information Resources (CLIR)
- **Professional organizations**. For example, in the case of LACLI, it was awarded the award for best public project by LASA's Archives, Libraries, and Digital Scholarship section and the SALALM Award for Institutional Collaborative Initiatives. Beyond prestige, these awards bring visibility to your project in specific communities interested in your content. It can also encourage others to explore collaborations. In the case of the SALALM, this award recognizes international collaborations. In addition to professional organizations at the national level, please also consider those at the international level (e.g. the Alliance of Digital Humanities Organizations (ADHO), regional and local level.

Beyond organization-specific awards, there are independent recognition programs for digital humanities and scholarship projects, such as the Digital Humanities Awards. While these may not offer financial support, they can significantly boost your project's visibility and recognition in the field.

Now, let's proceed to our second objective: we will explore strategies **to increase user engagement** in your project and **foster partnerships** that improve your web-based tool's ability to support multiple languages and its overall global usefulness.

6. Organize an edit-a-thon, also known as an editathon. It is a collaborative event where people come together to create, update or improve content on a specific project. For example, there is a long tradition of edit-a-thons to contribute content to Wikipedia. These events usually have a specific focus, such as adding content related to a certain subject area, addressing content gaps, or focusing on underrepresented topics, cultures, or perspectives. Edit-a-thons are typically organized for a set period and include a training component, making them suitable for both experienced editors and newcomers. While traditionally held as in-person events, edit-a-thons can also be conducted virtually, allowing for wider participation. Community building is an important aspect of these events as they often serve as networking venues for people with similar interests and help build a sense of community among participants. This can translate into future collaborative partnerships. Edit-a-thons have become popular tools for institutions, organizations, and communities to engage people in content creation and improvement while promoting digital literacy and collaborative knowledge sharing. Ultimately, edit-a-thons can be a perfect venue to increase multilingual content and global reach by organizing edit-a-thons that can focus on a specific language, country etc. The underlying concept is that your project's quality and effectiveness will significantly improve through diverse contributions from individuals representing various cultural backgrounds, linguistic groups, and nationalities. This approach will ensure that the content included in your project is respectful of different values and perspectives. We learn together, we contribute together. Nobody holds the whole knowledge on one topic. This diversity of perspectives and experiences enhances the project's overall scope, relevance, and applicability across different contexts. Global collaboration enables projects to become more comprehensive, inclusive, and diverse by incorporating knowledge and perspectives from people worldwide.

7. Participate in conference presentations and create events such as webinars, lectures, etc. When presenting your work, tailor your approach to each venue's unique opportunities. At a conference presentation, your goal might be broad visibility for your project. In contrast, a roundtable could be ideal for in-depth discussions on specific initiatives or for sharing user experiences. By matching your expectations to the event type, you'll maximize the impact of your event and make the most of each opportunity to showcase your work. Events that target a highly specific community can also be very effective. For example, if you want to promote your content among scholars in Latin American film, a good place to start would be the Film Studies section of the Latin American Studies Association (LASA) https://lasaweb.org/en/sections/ This event could also turn into a edit-a-thon.

8. Choose and apply research methods to meet any specific goals you might have for your project. Focus groups is a pertinent qualitative research method that involves gathering a small group of carefully selected participants to engage in an open discussion about a specific topic, project, or idea. Another useful one could be surveys that would allow you to collect data, analyze, and interpret data. Think creatively about surveys. For example, you could design a survey asking what are the most essential resources included in your project and then disseminate the results as a way to promote your content and help potential users to identify the gems included in your collection.

9. Explore potential collaborations to provide a national and international dimension to your project. Beyond the usual networking opportunities (conferences, online platforms, etc.), look for relevant international research grants that require cross-country collaborations. This type of initiative offers an exceptional opportunity to build meaningful connections. Our experience with the LACLI project serves as an example of this strategy's success. We took the initiative to reach out to two prestigious institutions: Colegio de México and Fundação Getulio Vargas. Our efforts paid off when we were awarded a grant aimed at enhancing the multilingual content of our project. The grant, initially seen as a means to an end, transformed into a long-term collaboration. It provided us with a structured environment, understanding each other's working styles, learning the intricacies of effective international cooperation, and identifying areas where our combined expertise (language, cultural knowledge etc.) could bring solutions to any initiative related to our project. This experience laid a solid foundation for our partnership and opened doors to possibilities for future joint ventures.

Don't limit yourself to long-term partnerships. Be open to seeking help for specific challenges as they arise. When faced with a technical problem you can't solve, don't hesitate to reach out to experts or organizations with the right knowledge. This targeted approach can quickly provide you with the solutions you need. For example, LACLI found an efficient solution for multilingual subject access by collaborating with the Hispanic American Periodicals Index (HAPI). Instead of creating our own controlled vocabulary from scratch, LACLI adopted HAPI's existing thesaurus, which includes terms in English, Spanish, and Portuguese. This partnership not only saved time and resources but also ensured consistency in terminology across platforms. HAPI regularly shares updates to their vocabulary, and both organizations maintain open communication to address any term-related issues. While LACLI occasionally added new terms to meet specific needs, this collaboration significantly streamlined our ability to provide multilingual access to content without duplicating efforts.

In the spirit of avoiding redundant work, seek out opportunities to collaborate with similar projects to share metadata. Additionally, think about using a Creative Commons license for your project. Such a license can make it easier and more straightforward for others to use and build upon your work, fostering a culture of open collaboration and shared resources.

10. Design experiential learning opportunities.
Experiential learning offers numerous advantages for students and creators of projects. This hands-on approach to education provides an engaging learning environment that goes beyond traditional classroom instruction. Students develop a wide range of skills that are valuable in both academic and professional settings. By engaging in real-world challenges, students learn to approach problems creatively and think outside the box. If you develop one of these opportunities, make sure you clearly articulate the following in your position announcement: your learning objectives, skills required, benefits, outcomes, and mentorship provided.

ACTIVITY- AUTHENTIC ASSESSMENT

Apply what you have learned so far in this episode! Now it is your turn to come up with your own plan! Develop an action plan that:

- Identifies and evaluates effective strategies to promote and increase visibility of your project among your diverse and global potential users.
- Encourages participation in your project and nurture collaborations that enhance the multilingual functionality, cultural diversity, and global usefulness of your project.

Note: Use the table below to guide your thinking process as you design your action plan.

ACTION PLAN	
Project Title:	
Promoting Your Project	
1. Create a variety of content about your project and encourage others to reuse your data and link to your project	*Some ideas (delete these as you go):* ● Wikipedia entry ● Select a creative commons license appropriate to your work ● Encourage librarians to link to your project ● Encourage other library projects or initiatives to include a metadata record for your project
2. Disseminate information about your project via Listservs of professional organizations	*Some ideas (delete these as you go):* Think of both subject-related and library/archive related organizations.
3. Build an effective social media strategy	*Some ideas (delete these as you go):* ● Create social media accounts/profiles in those social media services you consider more effective for your intended audience. ● Follow organizations and people related to the subject of your project. ● Be strategic about the context, nature of the content and when you post it. ● Promote your project by using hashtags ● Be aware of the limitations of social media.
4. Use analytics and AI tools	*Some ideas (delete these as you go):* Consider aspects such as: ● Best day and time to post your content ● Visitor metrics ● Type of content with more views and reactions

	● Use effective prompts for various types of analysis such as comparing your project with similar projects.
5. Apply for awards and grants	*Some ideas (delete these as you go):* ● Find information about them in specialized databases ● Think of organizations that might support projects like yours (national agencies, professional organizations, etc.)
Encouraging Participation and Collaboration	
6. Organize an edit-a-thon	*Some ideas (delete these as you go):* ● Define its focus: topic, tasks, language, etc. ● Type of event: online, in-person, hybrid? ● How would you organize it so that it could contribute to community building and potential future collaborations?
7. Participate in conference presentations and create events such as webinars, lectures, etc.	*Some ideas (delete these as you go):* ● Conference presentations, roundtables, academic posters, etc. ● Webinars, lectures ● Events for highly targeted groups
8. Choose and apply research methods to meet any specific goals you might have for your project.	*Some ideas (delete these as you go):* ● Focus groups ● Surveys, questionnaires ● Interviews
9. Explore potential collaborations to provide a national and international dimension for your project.	*Some ideas (delete these as you go):* ● Beyond the usual networking opportunities (conferences, online platforms, etc.), look for relevant international research grants that require cross-country collaborations. ● Don't limit yourself to long-term partnerships. Identify potential collaborators that could be helpful for specific challenges as they arise. ● Actively look for chances to share metadata with related projects.

10. Design experiential learning opportunities	*Some ideas (delete these as you go):* ● Apply to grants to support these initiatives ● Articulate the learning objectives, skills required, value, outcomes, and mentorship provided for these opportunities. Ensure mutual benefit for students and your project.