



R-ALAB 308A.4.1:

Working with External Data

Version 1.0, 10/18/23

[Click here to open in a separate window.](#)

Introduction

This **graded** assignment challenges you to create an interactive application using data from an external Application Programming Interface (API). In order to accomplish this task, you will employ the tools and techniques you have learned thus far, including [fetch](#), [Axios](#), and [async/await](#) syntax.

Objectives

- Request data from an external API using [fetch](#) and [Axios](#).
- Create an interactive, dynamic webpage that serves content from an external API.
- Use [async/await](#) and Promises to create synchronous logic in asynchronous actions.
- Using [fetch](#) or [Axios](#), [POST](#) data to and [DELETE](#) data from an external API.

Submission

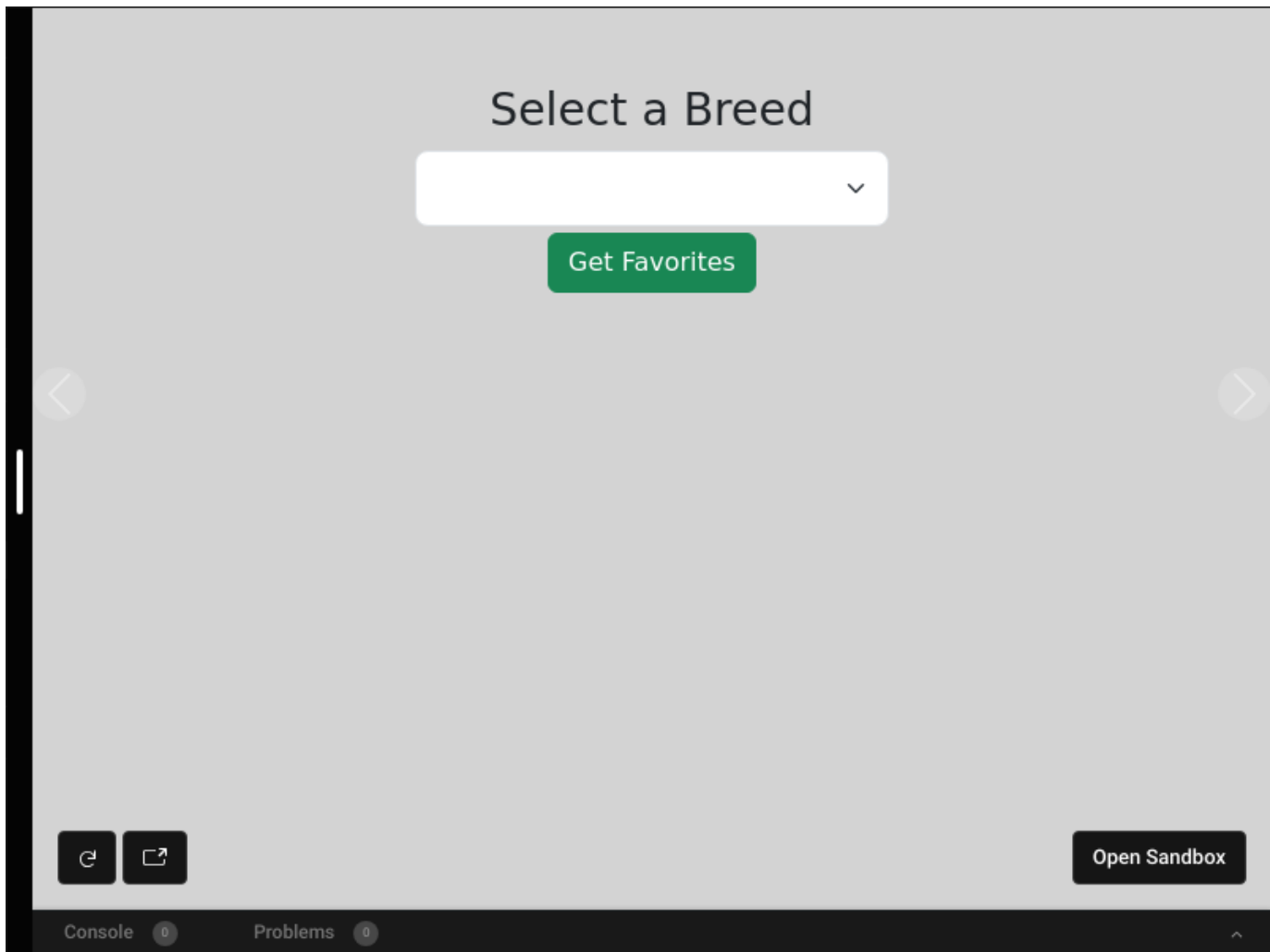
Submit your completed lab using the **Start Assignment** button on the assignment page in Canvas.

Your submission should include:

- A GitHub link to your completed project repository.

Instructions

The CodeSandbox below contains pre-configured contents, including an HTML layout, CSS styling, and instructions embedded in the form of comments. Take a few minutes to familiarize yourself with the contents of the sandbox.



Download the sandbox and initialize a local git repository. Link this repository to GitHub, and use your local development environment to complete the tasks that follow.

You will be primarily working within the [index.js](#) file.

Commit frequently! Every time something works, you should commit it. Remember, you can always go back to a previous commit if something breaks.

Part 1: The API

For the purposes of this assignment, you will be working with [The Cat API](#), which is a free public API.

Navigate to the API's website using the link above, and then create an account as follows:

PRICING ABOUT PRICING DOCUMENTATION SHOWCASE

Signup for an API key

It's completely free, email is only be used to send you an API Key & your stats, and you can use the API as much as you like. No Spam Ever.

E-mail
☒ youremail@el

App Description

What type of project will you use the API for?

☐ A personal project

☐ A project for school/college/university

☐ A business project

☐ Opt-in to my newsletter about other APIs (I'm building?)

SIGNUP

[Privacy Policy](#) | [Terms & Conditions](#)

Details
About Us
Terms & Conditions
Privacy Policy

Community and Support
Forum
Github
Discord

Our other APIs
The Auth API
The Report API
The Analytics API
The Image API

Discord Join our Discord community to share your API project!

The first step toward working with any API is exploring its documentation. Every API you work with will be different, so never make any assumptions about how an API works.

Once you have familiarized yourself with the fundamentals of the API, continue to the tasks below.

Part 2: Tasks

Now that you know what you are working with, implement the following. These steps are best accomplished in order, but feel free to approach the problem differently if you see a creative and efficient solution:

1. Create an `async` function "`initialLoad`" that does the following:
 - Retrieve a list of breeds from the cat API using `fetch()`.
 - Create new `<options>` for each of these breeds, and append them to `breedSelect`.
 - Each option should have a `value` attribute equal to the `id` of the breed.
 - Each option should display text equal to the `name` of the breed.
 - This function should execute immediately.
2. Create an event handler for `breedSelect` that does the following:
 - Retrieve information on the selected breed from the cat API using `fetch()`.
 - Make sure your request is receiving multiple array items!
 - Check the API documentation if you are only getting a single object.
 - For each object in the response array, create a new element for the carousel.
 - Append each of these new elements to the carousel.
 - Use the other data you have been given to create an informational section within the `infoDump` element.
 - Be creative with how you create DOM elements and HTML.
 - Feel free to edit `index.html` and `styles.css` to suit your needs.
 - Remember that functionality comes first, but user experience and design are also important.

- Each new selection should clear, re-populate, and restart the carousel.
 - Add a call to this function to the end of your [initialLoad](#) function above to create the initial carousel.
3. Create an additional file to handle an alternative approach.
4. Within this additional file, change all of your [fetch\(\)](#) functions to [Axios](#)!
- [Axios](#) has already been imported for you within [index.js](#).
 - If you've done everything correctly up to this point, this should be simple.
 - If it is not simple, take a moment to re-evaluate your original code.
 - **Hint:** [Axios](#) has the ability to set default headers. Use this to your advantage by setting a default header with your API key so that you do not have to send it manually with all of your requests! You can also set a default base URL!
5. Add [Axios](#) interceptors to log the time between request and response to the console.
- **Hint:** you already have access to code that does this!
 - Add a [console.log](#) statement to indicate when requests begin.
 - As an added challenge, try to do this on your own without referencing the lesson material.
6. Create a progress bar to indicate the request is in progress.
- The [progressBar](#) element has already been created for you.
 - You need only to modify its [width](#) style property to align with the request progress.
 - In your request interceptor, set the width of the [progressBar](#) element to 0%.
 - This is to reset the progress with each request.
 - Research the [axios onDownloadProgress](#) config option.
 - Create a function "[updateProgress](#)" that receives a [ProgressEvent](#) object.
 - Pass this function to the [axios onDownloadProgress](#) config option in your event handler.
 - [console.log](#) your [ProgressEvent](#) object within [updateProgress](#), and familiarize yourself with its structure.
 - Update the progress of the request using the properties you are given.
 - Note that we are not downloading a lot of data, so [onDownloadProgress](#) will likely only fire once or twice per request to this API. This is still a concept worth familiarizing yourself with for future projects.
7. As a final element of progress indication, add the following to your [Axios](#) interceptors:
- In your request interceptor, set the body element's [cursor](#) style to "[progress](#)."
 - In your response interceptor, set the body element's [cursor](#) style to "[default](#)."
8. To practice posting data, we will create a system to "favourite" certain images.
- The skeleton of this [favourite\(\)](#) function has already been created for you.
 - This function is used within [Carousel.js](#) to add the event listener as items are created.
 - This is why we use the [export](#) keyword for this function.
 - Post to the cat API's [favourites](#) endpoint with the given [id](#).
 - The API documentation gives examples of this functionality using [fetch\(\)](#); use [Axios](#)!
 - Add additional logic to this function such that if the image is already favoured, you delete that favourite using the API, giving this function "toggle" behavior.
 - You can call this function by clicking on the heart at the top right of any image.
9. Test your [favourite\(\)](#) function by creating a [getFavourites\(\)](#) function.

- Use [Axios](#) to get all of your favourites from the cat API.
- Clear the carousel and display your favourites when the button is clicked.
- You will have to bind this event listener to [getFavouritesBtn](#) yourself.
- **Hint:** you already have all of the logic built for building a carousel. If that is not in its own function, maybe it should be so that you do not have to repeat yourself in this section.

10. Test your site, thoroughly! Debug your code as needed.

- What happens when you try to load the Malayan breed?
 - If this is working, good job! If not, look for the reason why and fix it!
- Test other breeds as well. Not every breed has the same data available, so your code should account for this.

Solution Demonstration

Your instructor will have access to a demo solution that can be explored to give you an idea of what you should be working towards.

Note that your [infoDump](#) section (task 2) can be much more evolved than it is within the example if you have the time to expand upon it.

Your instructor will also have access to the completed solution code that can be used as a review tool after completion of the assignment. Code review is an important part of the development cycle; reviewing your code, the example solution, or a peer's code can help solidify understanding of these concepts, provided there is time to do so.

If you do review the solution's code, do not assume it is perfect (hint: it isn't)! It works, but there are improvements to be made.

Can you identify a few of these potential improvements? We will give you one for free: where is the error handling?!