Building the Simple Liker App

- Due No Due Date
- Points 1
- Submitting a website url

(https://github.com/learn-co-curriculum/phase-1-building-simple-liker) (https://github.com/learn-co-curriculum/phase-1-building-simple-liker/issues/new)

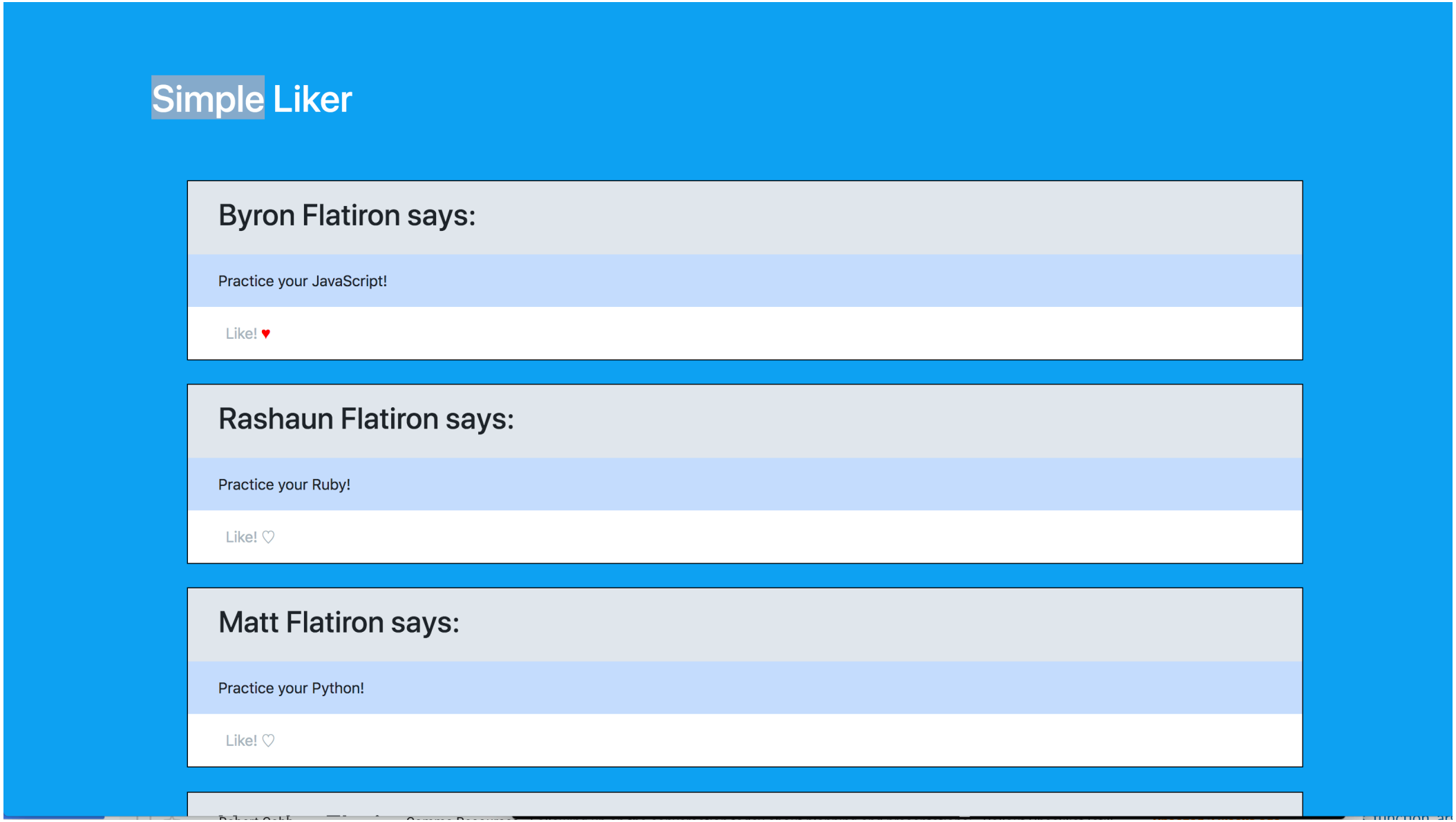
Learning Goals

- Set up an event listener to respond to a user action
- Submit a request to a mocked-up server
- Update the DOM based on the mock server's response

Introduction

Remember when we started this exploration of the "Simple Liker" application? You might not have been sure that you would make it to this point, but you have. Right now you should have the information needed to create a basic web application!

Your goal is to implement the "liking" functionality of "Simple Liker." As a reminder, the final product should look something like this:



The focus of this lab is the JavaScript code. You should only need to make one change to the HTML, and no changes to the CSS file.

You might be tempted to look back at previous code, but don't. Use your knowledge and documentation from the internet (if needed), to build the application.

Instructions

You will be doing your coding in `main.js`. If you take a look at the file, you will see that a function, `mimicServerCall()`, is being provided for you. This function will "mock" the behavior of a backend server. You will invoke `mimicServerCall()` in response to a user action, and the function will randomly return either a "success" or "fail" response. Your code will then need to handle the response appropriately: updating the appearance of the heart if it returns a "successful" response, and displaying an error in the DOM otherwise.

Note that the content of the "successful" response from the server is not important here — we only care that it's successful. This means you will not need to call `.json()` on the response so you only need a single `then()` call.

Here's the specification:

- Add the `.hidden` class to the error modal in the HTML so it does not appear when the page first loads
- When a user clicks on an empty heart:
  - Invoke `mimicServerCall` to simulate making a server request
  - When the "server" returns a failure status:
  - Respond to the error using a `.catch(() => {})` block after your `.then(() => {})` block.
  - Display the error modal by removing the `.hidden` class
  - Display the server error message in the modal
  - Use `setTimeout` to hide the modal after 3 seconds (add the `.hidden` class)
  - When the "server" returns a success status:
  - Change the heart to a full heart
  - Add the `.activated-heart` class to make the heart appear red
- When a user clicks on a full heart:
  - Change the heart back to an empty heart
  - Remove the `.activated-heart` class
- Keep all your styling rules entirely in `style.css`. Do not manipulate any `.style` properties.
- Only manipulate the DOM once the server request responds. Do not make the heart full until you're inside a successful `.then` block.

> Note: The tests will only check for the first part of the specification (adding the `hidden` class). You should verify the rest of the behavior yourself, by checking the page in the browser.

Conclusion

That's it! Congratulations. You're now a real-deal front-end developer! You can use HTML, CSS, and JavaScript to create living, breathing applications. Every web application front-end you see or have seen is built using these three pillars, which you're now skilled with! Give yourself a well-deserved pat on the back!