6.4.1 Introduction

You've made great progress so far on the Git it Done app! You've flexed your JavaScript abilities and received the anticipated responses for the GitHub API calls for both the repos and the issues endpoints. However, in order to test the API call, you initially hardcoded it, as shown in this image:

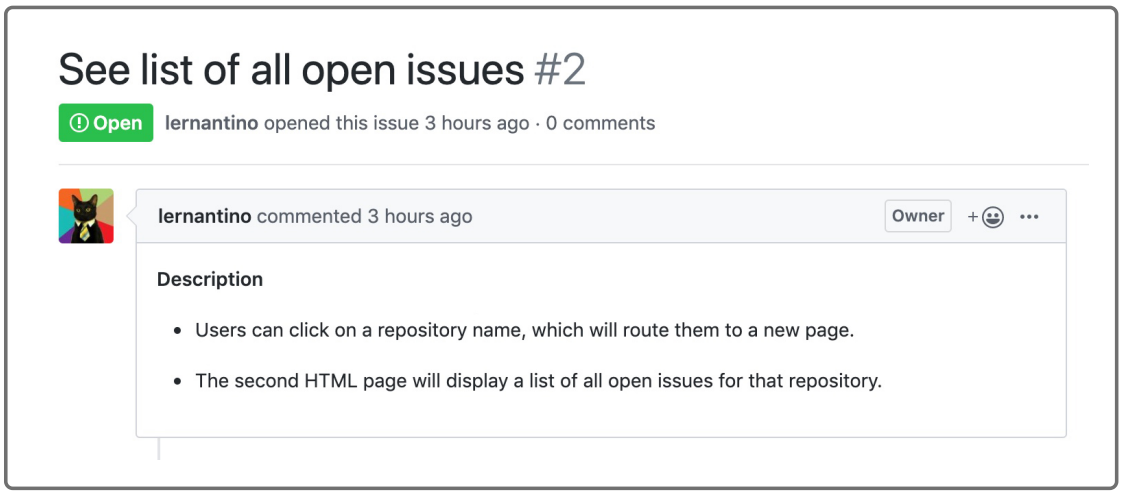
As the following image shows, the homepage can populate a list of repos from the user:

But the problem is getting from the repository search results to the new issues page. If you click on an individual repo from the list on the homepage, nothing happens. You can't navigate to the `single-repo.html` page to view the corresponding issues for a selected repo.

In this lesson, we'll resolve that problem by calling the GitHub API issues endpoint dynamically. Once the user has selected a repo from the list on the homepage, they'll be automatically directed to the `single-repo.html` page. There, they'll see a list of issues that correspond to the repo name selected, if issues for that repo exist.

Your friend Amiko is pleased with your progress so far; the most difficult parts of this project are behind you! Now you just need to connect the pages.

To determine what we need to accomplish in this lesson, let's look at the corresponding GitHub issue, shown in the following image:



As the issue describes, if a user clicks on a repo name from the list, a new page should load, populated with open issues related to the selected repo. But how do we transfer the repo name from the homepage, `index.html`, to the issues page, `single-repo.html`?

ON THE JOB

Whenever we build something mechanical from component parts—whether that be a computer, fan, or car—once we've determined that the parts work, we can wire them together so that the pieces function together. Similarly, once we've verified that the API calls work by rendering the response in the browser, we can connect these parts to operate as a cohesive application. Web developers at all stages of their careers often have to connect different parts of an application to supply critical data when and where it's needed.

In this lesson, you'll learn how to do the following:

- Pass information from one page to another using query parameters
- Obtain data from a URL using browser-provided location objects
- Make the API call dynamic by using the query parameter to alter the request

In tackling this problem, you'll implement a strategy that will serve you throughout your career. It’s common in web development to allow users to select individual items from a high-level list to display more details about those items, just as you're doing with the Git it Done search results. E-commerce sites and restaurant-listing sites exemplify this practice.