6.1.6 Update Request Logic with Dynamic Variables

Testing a server-side API with hardcoded values (like the Octocat username) can help us verify that the API will work at all before we get carried away with too much other logic. Now that we know we can get the data from GitHub, we can update the `getUserRepos()` function to request any user's repositories.

CONNECT THE DOTS

You can liken an API endpoint to a function in the sense that the same base URL can return different data depending on the argument(s) given. Consider the following examples:

```
getUserRepos("octocat"); // https://api.github.com/
getUserRepos("microsoft"); // https://api.github.co
```

Or more generically:

```
getUserRepos(user); // https://api.github.com/users
```

In `homepage.js`, restructure the `getUserRepos()` function as follows:

```
var getUserRepos = function(user) {
  // format the github api url
  var apiUrl = "https://api.github.com/users/" + user + "/repos";

  // make a request to the url
  fetch(apiUrl).then(function(response) {
    response.json().then(function(data) {
      console.log(data);
    });
  });
};
```

Note how we've added a parameter to the `getUserRepos()` function and inserted the parameter into the GitHub API URL. We then use the newly formatted URL in the subsequent `fetch()` request.

Test out the new logic by calling `getUserRepos()` with a few different usernames, like `getUserRepos("microsoft")` or `getUserRepos("facebook")`. You can even try your own username! Check the console each time to see if the returned array changes.

If everything looks good, go ahead and save your progress with Git. There's still more work to be done before you can mark this user repos feature complete, and in the next lesson you'll continue where you left off.