6.4.5 Handle Errors

In this step, we'll focus on error handling in the `single.js` file. To identify where potential errors could arise, consider the key parameters each function needs to operate correctly. For instance, suppose the `getRepoName()` function couldn't retrieve the correct query parameter or perhaps didn't get one at all. What should we do in this situation?

If a query parameter is unavailable, we should redirect the user back to the `index.html` page to try entering a username again.

Let's start by going into the `single.js` file, in the `getRepoName()` function. Add a conditional statement to check for valid values before passing them into their respective function calls. If the `repoName` is valid, feed the value to the `fetch` call; otherwise, redirect back to the `index.html` page.

For now, let's move the `getRepoIssues()` and the `repoName.textContent` expressions into a conditional statement that checks if the `repoName` exists, as follows:

```
if(repoName) {
  repoNameEl.textContent = repoName;
  getRepoIssues(repoName);
}
```

The preceding conditional statement will only display the repo name and make the `fetch` call if the value for `repoName` exists. For the second part of this conditional, we'd like to redirect back to the homepage. To do this, let's refer to the **MDN web docs on the browser-based Location Web API (https://developer.mozilla.org/en-US/docs/Web/API/Location)** .

Here we'll see many available properties and methods, but we'll pay particular attention to the `replace()` method. This method replaces the current resource or page with the one at the provided URL.

Let's try this out in the browser. Open the console and type the following statement into the prompt:

```
document.location.replace("https://google.com/search?q=javascript");
```

Note how this command redirected the page to the Google JavaScript search.

PAUSE

Which relative path would redirect us to `index.html`?

It would be `./index.html`.

Hide Answer

Add the second part to the conditional statement that will occur when the `repoName` doesn't exist. In this case, we want to redirect the user back to the homepage to try again.

```
else {
  document.location.replace("./index.html");
}
```

Excellent work! Now the `getRepoName()` function should look like this (with comments):

```
var getRepoName = function() {
  // grab repo name from url query string
  var queryString = document.location.search;
  var repoName = queryString.split("=")[1];

  if (repoName) {
    // display repo name on the page
    repoNameEl.textContent = repoName;

    getRepoIssues(repoName);
  } else {
    // if no repo was given, redirect to the homepage
    document.location.replace("./index.html");
  }
};
```

Time to check if this error-handling feature works! To simulate not receiving a repository name, open the `single-repo.html` file directly in the browser. Remember, you can do this from VS Code with the "Open in Default Browser" option. Because opening the file directly does not include a query string, the browser will quickly redirect to `index.html`.

By checking to see if the query parameter is available, we can preempt a potential error before it ever reaches the API call. We could also use this technique within the API call itself when we check the validity of the response received from the issues endpoint. Currently, if the response is invalid, the user receives an alert saying there was a problem with the request. Instead, let's redirect the user back to the homepage.

Open the `single.js` file and go to the `getRepoIssues()` function. In the `fetch()` function, we see a conditional statement that's checking if the response from the issues endpoint from the GitHub API is valid. Let's change the `else` portion of this statement to redirect the user back to the homepage rather than send the user an `alert()`. Change that portion of the code now, and remove the `console.log()` as well.

The `fetch()` function should now look like the following code:

```
// make a get request to url
fetch(apiUrl).then(function(response) {
  // request was successful
  if (response.ok) {
    response.json().then(function(data) {
      displayIssues(data);

      // check if api has paginated issues
      if (response.headers.get("Link")) {
        displayWarning(repo);
      }
    });
  } else {
    // if not successful, redirect to homepage
    document.location.replace("./index.html");
  }
});
```

To check if this redirect is working, update the URL in your browser's address bar from `index.html` to something like: `single-repo.html?repo=nope`. There is no `"nope"` repository on GitHub, which will cause a bad response when fetched, and thus trigger the redirect.