6.3.6 Add an Extra Message

So we've got the `single-repo.html` page working like we want, but there's a little bug. Right now, we can't view more than 30 issues at a time. If a repo has more than 30 issues, the user can't tell that some aren't being displayed.

You can test this out by calling `getRepoIssues()` with `"facebook/react"`, `"expressjs/express"`, `"angular/angular"`, or pretty much any popular repo out there.

If you want to find out why only 30 issues show up, you can refer to the GitHub documentation on pagination. Notice that GitHub limits requests to 30 items at a time.

PAUSE

Why would GitHub limit the number of results like that?

Transmitting the contents of the entire GitHub database through every request made to their API would be costly. Someone has to pay for all those downloads!

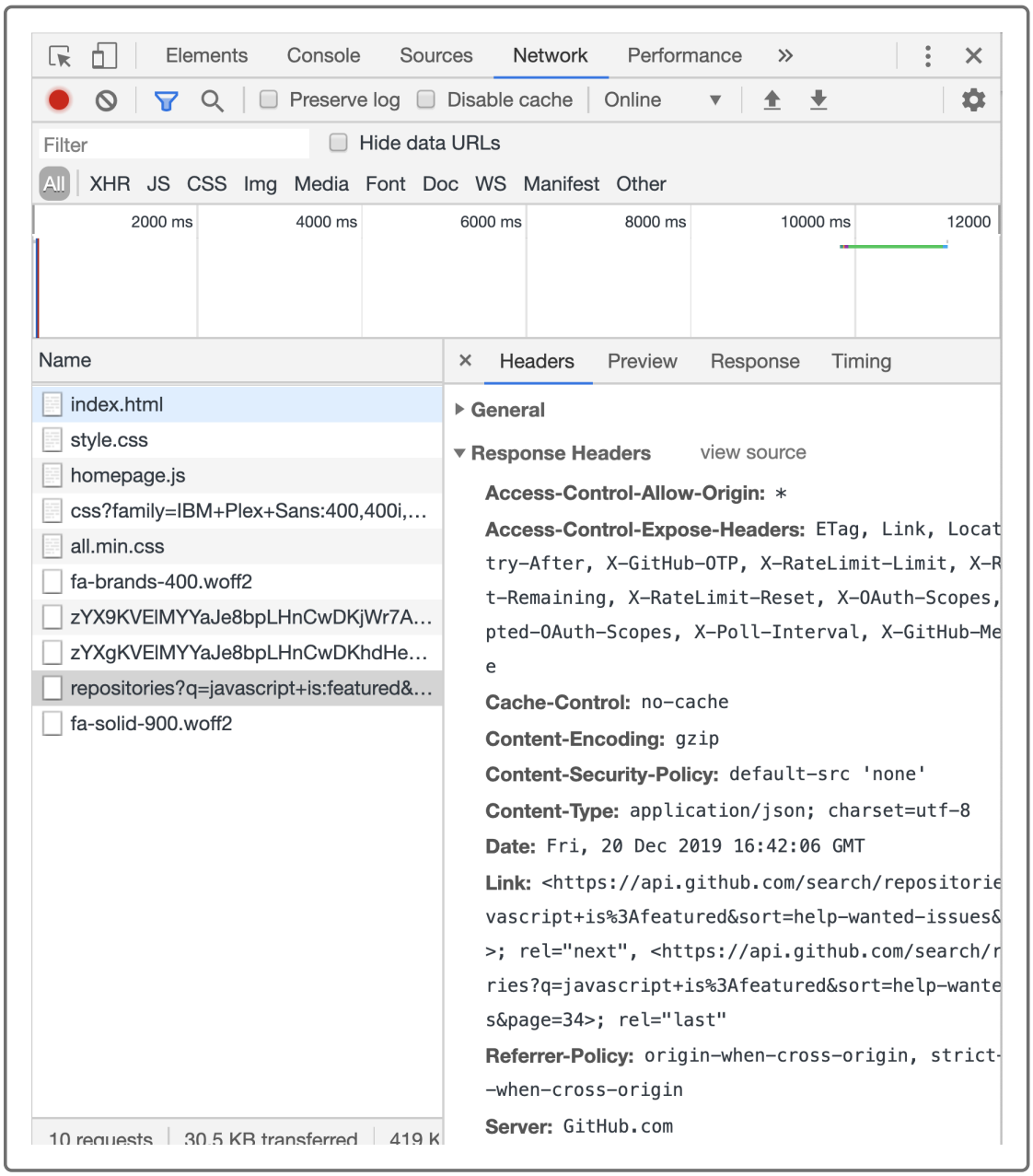
Hide Answer

The documentation explains that GitHub will actually supply a link header with more information if pagination exists for the data we're requesting. Okay, so what are headers?

HTTP headers allow the client and the server to pass additional information with an HTTP request or response. This often includes information like whether or not to cache (locally store) data and, if so, for how long. This data goes in the headers because it's often small in file size and not directly pertinent to the content in the body of a request or response.

For example, the very common `Content-Type` header specifies the format of the requested resource. The GitHub API responses always include a `Content-Type` value of `application/json` to signify that the response is formatted as JSON.

You can view the headers on a request or response in DevTools by clicking on the request in the Network tab, then selecting Headers on the righthand side, as shown in the following image:



DEEP DIVE

DEEP DIVE

Headers can also include information about authorization, cookies, and other security-related data. For more information on headers, visit the [MDN web docs on HTTP]

headers (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers).

For this app, we won't worry about trying to show all of the paginated results. But we do want to make sure we let users know if there are more issues that can't be viewed. If that's the case, we'll direct users to GitHub, where they can see the full list.

Here's how we'll get the message across. In `getRepoIssues()`, add a condition that checks for the `Link` header. If it exists, console log "repo has more than 30 issues", as shown in the following code:

```
if (response.ok) {
  response.json().then(function(data) {
    displayIssues(data);

    // check if api has paginated issues
    if (response.headers.get("Link")) {
      console.log("repo has more than 30 issues");
    }
  });
}
```

Test the app with a repository that's bound to have more than 30 issues, like `"facebook/react"`. Now we'll change this console log message into something a little more user-friendly.

First, you'll need to create a new container in the `single-repo.html` file. At the bottom of the `<main>` element, add the following:

```
<div id="limit-warning">

</div>
```

Next, at the top of `single.js`, declare a DOM reference to this container as such:

```
var limitWarningEl = document.querySelector("#limit-warning");
```

Create a new `displayWarning()` function with a `repo` parameter. Then, in `displayWarning()`, update the `textContent` of `limitWarningEl` with the beginning part of the message, as follows:

```
var displayWarning = function(repo) {
  // add text to warning container
  limitWarningEl.textContent = "To see more than 30 issues, visit ";
};
```

Then in `displayWarning()`, append a link element with an `href` attribute that points to `https://github.com/<repo>/issues`. Don't forget to append your link to the warning container! The code should look like this:

```
  var linkEl = document.createElement("a");
  linkEl.textContent = "See More Issues on GitHub.com";
  linkEl.setAttribute("href", "https://github.com/" + repo + "/issues"
  linkEl.setAttribute("target", "_blank");

  // append to warning container
  limitWarningEl.appendChild(linkEl);
```

Finally, in `getRepoIssues()`, replace the console log in the `if (response.headers.get("Link"))` statement with a call to `displayWarning(repo)`.

Just as we did in the last section, test this new function out with different popular repos. If all looks good, commit your progress!