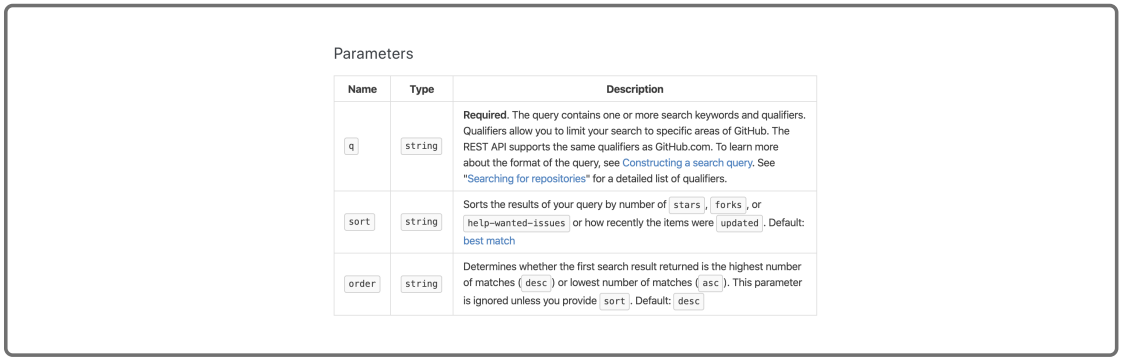
6.5.3 Create a Function to Call the GitHub Search API

Alright, let's build a function to call the search API. However, before we do so, let's make sure to create a new Git branch called `feature/languages`.

And before we spend too much time on the smaller details, we should try getting the API endpoint working. To do so, let's head over to the **GitHub API docs section on searching for repos (https://developer.github.com/v3/search/#search-repositories)** .

Note that the search method only returns up to 100 repositories. This works for our purposes, because we only need the user to view 30 or so at a time.

Direct your attention to the parameters section, as shown in the following image:



Parameters

Name	Type	Description
`q`	`string`	**Required.** The query contains one or more search keywords and qualifiers. Qualifiers allow you to limit your search to specific areas of GitHub. The REST API supports the same qualifiers as GitHub.com. To learn more about the format of the query, see Constructing a search query. See "Searching for repositories" for a detailed list of qualifiers.
`sort`	`string`	Sorts the results of your query by number of `stars`, `forks`, or `help-wanted-issues` or how recently the items were `updated`. Default: best match
`order`	`string`	Determines whether the first search result returned is the highest number of matches (`desc`) or lowest number of matches (`asc`). This parameter is ignored unless you provide `sort`. Default: `desc`

Remember, we recently learned how to use parameters! This time, we can use them to specify that we want the search to not only include results that match the query but also sort them.

IMPORTANT

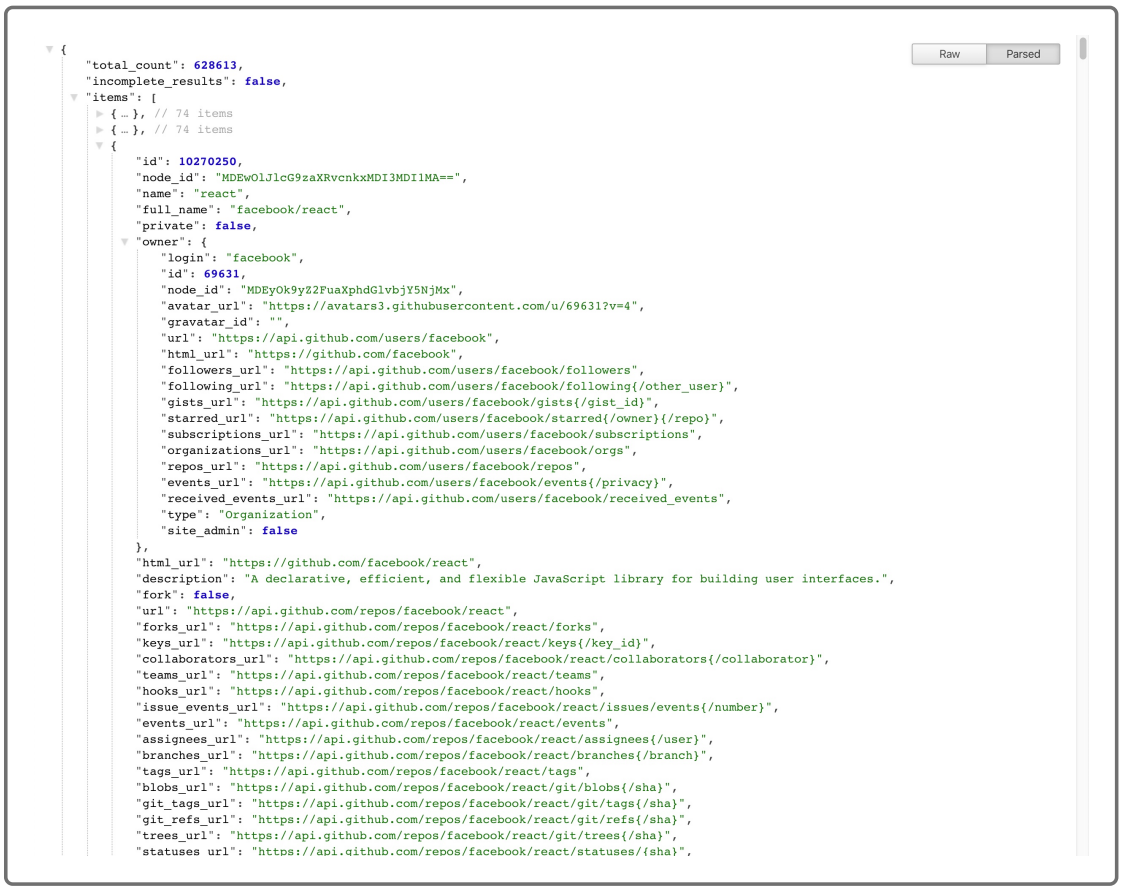
> We should sort results when we make queries involving large collections of data. In this case, if we don't, the API could return unmaintained, unpopular repositories. We want to avoid these kinds of repositories, which can be more challenging to contribute to.

Now that we're using multiple parameters, we need to specify where the first parameter ends and the second one begins. We can do this by including the `&` symbol between each parameter in the URL.

With that in mind, enter the following URL in your browser:

```
https://api.github.com/search/repositories?q=javascript+is:featured&sc
```

The result should look something like the following image:



Now try that same endpoint without the `sort` parameter.

Notice that the result still includes repositories with a high number of stars. This happens because the `sort` parameter has a default value of `score` in case we don't provide one. GitHub applies the `score` value to repositories based on their relevance to the search term and their popularity.

The GitHub API allows us to add additional parameters, known as **qualifiers**. For example, adding `+is:featured` tells the API that we only want featured repositories. That's not all, though! We can add as many qualifiers as we can think of, as long as the URL remains under 255 characters.

DEEP DIVE

DEEP DIVE

To learn more, read the **GitHub Search API documentation (https://developer.github.com/v3/search/#limitations-on-query-length)** .

As mentioned in the documentation, the structure for the API call will look something like this:

```
q=SEARCH_KEYWORD_1+SEARCH_KEYWORD_N+QUALIFIER_1+QUALIFIER_N
```

This example provided by GitHub includes two search keywords and two qualifiers, but the query itself can accept a varied number of each. The `_N` syntax indicates the last qualifier. A real example of this would look like the following code:

```
q=javascript+html+css+is:featured
```

In this case, the `N` keyword would be `css` and the `N` qualifier would be `is:featured`, because they are the last of each set of data, respectively.

DEEP DIVE ▲

DEEP DIVE

To learn more, see the **GitHub API docs on searching for repositories (https://help.github.com/en/github/searching-**

for-information-on-github/searching-for-repositories) .

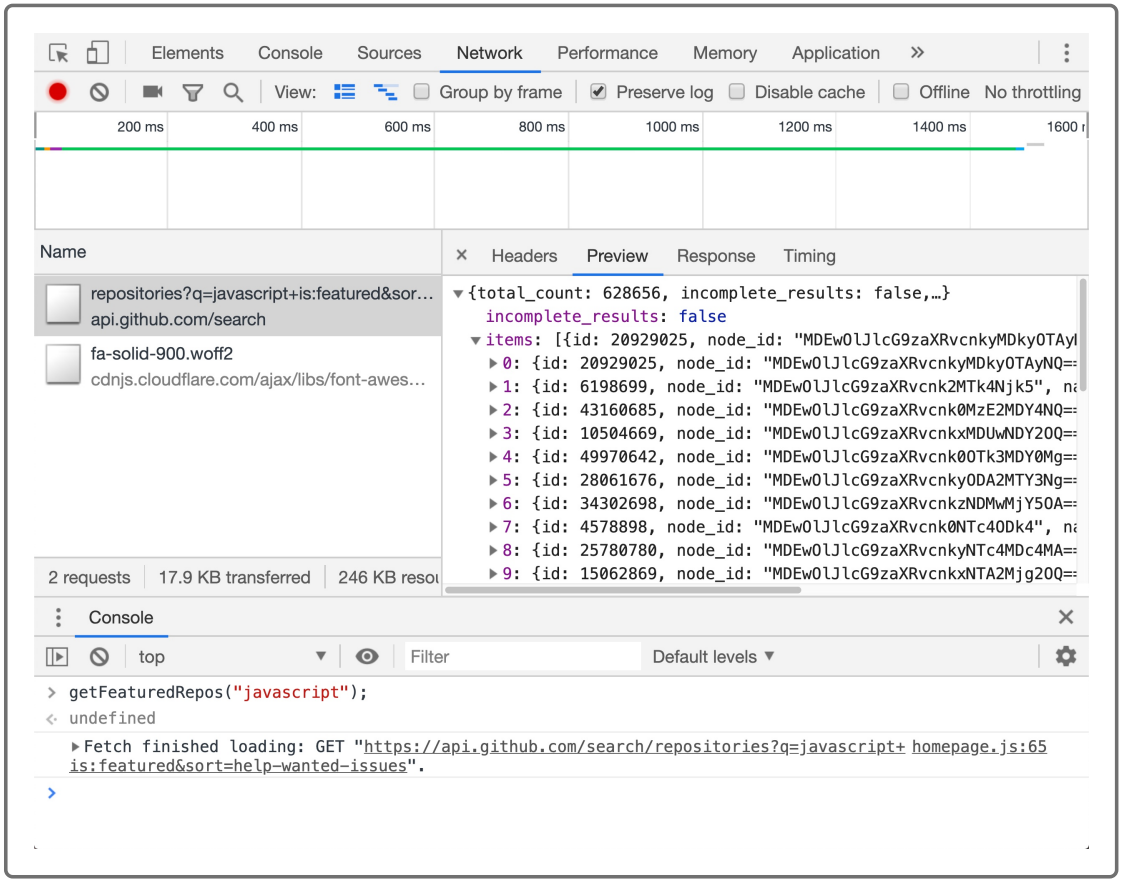
Now, in `homepage.js`, create a new function called `getFeaturedRepos()` that accepts a `language` parameter, creates an API endpoint, and makes an HTTP request to that endpoint using `fetch()`, as shown in the following code:

```
var getFeaturedRepos = function(language) {
  var apiUrl = "https://api.github.com/search/repositories?q=" + langu

  fetch(apiUrl);
};
```

To verify that this endpoint works, open `index.html` in your browser and call `getFeaturedRepos("javascript");` in the console, as shown in the following image:

We didn't console log the response, but we can go to the Network tab and click on the response to view it in its entirety, like in the following image:



Perfect! Now that we've tested the new function, we can move on to displaying the data on the page.