David J. Malan
malan@harvard.edu

Search

Design a front-end for Google Search, Google Image Search, and Google Advanced Search.

Background

Recall from lecture that we can create an HTML form using a `<form>` tag and can add `<input>` tags to create input fields for that form. Later in the course, we'll see how to write web applications that can accept form data as input. For this project, we'll write forms that send data to an existing web server: in this case, Google's.

When you perform a Google search, as by typing in a query into Google's homepage and clicking the "Google Search" button, how does that query work? Let's try to find out.

Navigate to google.com, type in a query like "Harvard" into the search field, and click the "Google Search" button.

As you probably expected, you should see Google search results for "Harvard." But now, take a look at the URL. It should begin with `https://www.google.com/search`, the route on Google's website that allows for searching. But following the route is a `?`, followed by some additional information.

Those additional pieces of information in the URL are known as a query string. The query string consists of a sequence of GET parameters, where each parameter has a name and a value. Query strings are generally formatted as

```
field1=value1&field2=value2&field3=value3...
```

where an `=` separates the name of the parameter from its value, and the `&` symbol separates parameters from one another. These parameters are a way for forms to submit information to a web server, by encoding the form data in the URL.

Take a look at the URL for your Google search results page. It seems there are quite a few parameters that Google is using. Look through the URL (it may be helpful to copy/paste it into a text editor), and see if you can find any mention of "Harvard," our query.

If you look through the URL, you should see that one of the GET parameters in the URL is `q=Harvard`. This suggests that the name for the parameter corresponding to a Google search is `q` (likely short for "query").

It turns out that, while the other parameters provide useful data to Google, only the `q` parameter is required to perform a search. You can test this for yourself by visiting `https://www.google.com/search?q=Harvard`, deleting all the other parameters. You should see the same Google results!

Using this information, we can actually re-implement a front end for Google's homepage. Paste the below into an HTML file called `index.html`, and open it in a browser.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Search</title>
    </head>
    <body>
        <form action="https://google.com/search">
            <input type="text" name="q">
            <input type="submit" value="Google Search">
        </form>
    </body>
</html>
```

When you open this page in a browser, you should see a (very simple) HTML form. Type in a search query like "Harvard" and click "Google Search", and you should be taken to Google's search results page!

How did that work? In this case, the `action` attribute on the `form` told the browser that when the form is submitted, the data should be sent to `https://google.com/search`. And by adding an `input` field to the form whose `name` attribute was `q`, whatever the user types into that input field is included as a GET parameter in the URL.

Your task in this project is to expand on this site, creating your own front end for Google Search, as by exploring Google's interface to identify what GET parameters it expects and adding the necessary HTML and CSS to your website.

Getting Started

- Download the distribution code from https://cdn.cs50.net/web/2020/spring/projects/0/search.zip and unzip it.

Specification

Your website must meet the following requirements.

- **Pages**. Your website should have at least three pages: one for Google Search, one for Google Image Search, and one for Google Advanced Search.
  - On the Google Search page, there should be links in the upper-right of the page to go to Image Search or Advanced Search. On each of the other two pages, there should be a link in the upper-right to go back to Google Search.
- **Query Text**. On the Google Search page, the user should be able to type in a query, click "Google Search", and be taken to the Google search results for that page.
  - Like Google's own, your search bar should be centered with rounded corners. The search button should also be centered, and should be beneath the search bar.
- **Query Images**. On the Google Image Search page, the user should be able to type in a query, click a search button, and be taken to the Google Image search results for that page.
- **Query Advanced**. On the Google Advanced Search page, the user should be able to provide input for the following four fields (taken from Google's own advanced search options)
  - Find pages with... "all these words:"
  - Find pages with... "this exact word or phrase:"
  - Find pages with... "any of these words:"
  - Find pages with... "none of these words:"
- **Appearance**. Like Google's own Advanced Search page, the four options should be stacked vertically, and all of the text fields should be left aligned.
  - Consistent with Google's own CSS, the "Advanced Search" button should be blue with white text. When the "Advanced Search" button is clicked, the user should be taken to search results page for their given query.
- **Lucky**. Add an "I'm Feeling Lucky" button to the main Google Search page. Consistent with Google's own behavior, clicking this link should take users directly to the first Google search result for the query, bypassing the normal results page.
- **Aesthetics**. The CSS you write should match Google's own aesthetics as best as possible.

Hints

- To determine what the parameter names should be, you're welcome to experiment with making Google searches, and looking at the resulting URL. It may also be helpful to open the "Network" inspector (accessible in Google Chrome by choosing View -> Developer -> Developer Tools) to view details about requests your browser makes to Google.
  - Any `<input>` element (whether its `type` is `text`, `submit`, `number`, or something else entirely) can have `name` and `value` attributes that will become GET parameters when a form is submitted.
  - You may also find it helpful to look at Google's own HTML to answer these questions. In most browsers, you can control-click or right-click on a page and choose "View Page Source" to view the page's underlying HTML.
- To include an input field in a form that users cannot see or modify, you can use a "hidden" input field.

How to Submit

Hold on! If you have already submitted and received a passing grade on the **prior version of Project 0** (Homepage), please stop here. You already have received an equivalence credit for this project, and you should not submit this assignment, as it will have no impact on your progress in the course and will therefore only slow our graders down!

1. Visit this link, log in with your GitHub account, and click **Authorize cs50**. Then, check the box indicating that you'd like to grant course staff access to your submissions, and click **Join course**.
2. Install Git and, optionally, install `submit50`.

When you submit your project, the contents of your `web50/projects/2020/x/search` branch should match the file structure of the unzipped distribution code as originally received. That is to say, your files should not be nested inside of any other directories of your own creation (`search` or `project0`, for example). Your branch should also not contain any code from any other projects, only this one. Failure to adhere to this file structure will likely result in your submission being rejected.

By way of example, for this project that means that if the grading staff visits `https://github.com/me50/USERNAME/blob/web50/projects/2020/x/search/index.html` (where `USERNAME` is your own GitHub username as provided in the form, below) your submission for `index.html` for this project should be what appears. If it doesn't, reorganize your repository as needed to match this paradigm.

3. If you've installed `submit50`, execute

   ```
   submit50 web50/projects/2020/x/search
   ```

   Otherwise, using Git, push your work to `https://github.com/me50/USERNAME.git`, where `USERNAME` is your GitHub username, on a branch called `web50/projects/2020/x/search`.
4. Record a screencast not to exceed 5 minutes in length, in which you demonstrate your project's functionality. **Your URL bar must remain visible throughout your demonstration of the project.** Be certain that every element of the specification, above, is demonstrated in your video. There's no need to show your code in this video, just your application in action; we'll review your code on GitHub. Upload that video to YouTube (as unlisted or public, but not private) or somewhere else. In your video's description, you must timestamp where your video demonstrates each of the seven (7) elements of the specification. **This is not optional**, videos without timestamps in their description will be automatically rejected.
5. Submit this form.

You can then go to https://cs50.me/cs50w to view your current progress!