Discoverable

- index
- Fix sneaky 404s
- How search works
- How to measure search engine optimization with Lighthouse

index

layout: collection override:tags: [] date: 2018-11-05 pathName: discoverable

Fix sneaky 404s

Single Page Apps can show different content without loading a new page. To do so, they use click handlers on links and the History API. The History API allows to manipulate the browser session history. This way we can update the URL when showing a different page (usually called a "view" in Single Page Apps). It also makes sure the browser's back button still works as expected.

Take a look at the Single Page App in this codelab. It shows either cat or dog images and provides links to toggle between the two animals. It seems to work fine!

Uncovering the sneaky 404

Unfortunately there is a subtle bug in the app. Let's take a look!

{% Instruction 'preview' %}

- Click the **Doggos** link. Notice how the URL changed. {% Instruction 'reload-app' %}

You get a page with "`Cannot GET /doggos`" on it—a sneaky 404. It is "sneaky", because the web app seems to work fine as long as you only click on links within it. It breaks when using the URLs in a new browser window or when refreshing the page. The issue is that the server does not know how to respond to a request for these URLs. The JavaScript code in our web app is using the History API to navigate between them, but the server does not know what to do with them. Whenever the server does not know what to do for a requested URL, it responds with the HTTP status code `404`. With this code the server says it hasn't found anything for the requested URL.

Search engines will not index the URLs in this case, because a user would click on a search result and find the error message, but not the content they were looking for, such as the dog pictures.

Fixing the server

This project uses an express.js server written in JavaScript. Let's fix the server, so it responds with index.html and the single page app will take care of the rest.

{% Instruction 'remix' %}

- Select the `server.js` file.

This file contains the server code. It sets up an express.js server and sends the content of index.html. The route setup in line 15 only serves the web app when requests go to the URL `/`. The server should also serve the other URLs we have created. Let's change this to serve all URLS, so it also works with additional URLs in the future.

To do so, we can change the code starting at line 15 to this:

```
app.get('/*', function(request, response) {
  response.sendFile(__dirname + '/views/index.html');
});
```

The `/*` matches any URL and the server now responds with the web app in `index.html` for any given URL.

{% Instruction 'preview' %}

Refreshing and opening the links in a new incognito window should now work as expected.

How search works

Search engines are the digital version of a librarian. They use a comprehensive index to help find the right information for a query. Understanding the basics of search prepares you to make your content **discoverable** for users.

How crawlers browse the web

Crawling is like reading through all the books in the library. Before search engines can bring any search results, they need to have as much information from the web as possible. To do this, search engines use a crawler—a program that travels from site to site and acts like a browser.

If a book or document is missing or damaged, the crawler can't read it. Crawlers try to fetch each URL to determine the state of the document. If a document returns an error status code, crawlers cannot use any of its content, and might retry the URL at a later time. This ensures only publicly accessible documents get into the index.

If crawlers discover a redirection status code (like 301 or 302), they follow the redirection to a new URL and continue there. Once they get a successful response, meaning they've found a document accessible to users, they check if it's allowed to be crawled and then download the content.

This check includes the HTML and all content mentioned in the HTML, such as images, videos, or JavaScript. Crawlers also extract the links from HTML documents so that the crawler can visit the linked URLs as well. Following links is how crawlers find new pages on the web.

Crawlers don't actively click links or buttons, but instead send URLs to a queue to crawl them later. When accessing a new URL, no cookies, service workers or local storage (like IndexedDB) are available.

Building an index

After retrieving a document, the crawler hands the content to the search engine to add it to the index. The search engine now renders and analyzes the content to understand it. Rendering means displaying the page as a browser would (with some limitations).

Search engines look at keywords, the title, links, headings, text, and many other things. These are called **signals** which describe the content and context of the page. Signals allow search engines to answer any given query with the best possible page.

Search engines might find the same content at different URLs. For example a recipe for "apple pie" might live under `/recipes/apple-pie` and under `/recipes/1234`. To avoid indexing and showing the recipe twice, search engines determine what the main URL should be and discard the alternative URLs showing the same content.

Serving the most useful results

Search engines do more work than just matching the query to keywords in the index. To give useful results, they might consider context, alternative wording, location of the user, and more. For example, "silicon valley" might refer to the geographic region or the TV show. But if the query is "silicon valley cast", results on the region aren't very helpful.

Some queries can be indirect, like "the song from pulp fiction", and search engines need to interpret that and show results for the music in the film. When a user searches for something, search engines determine the most useful results and then show them to the user. Ranking, or ordering, the pages happens based on the query. The order can often change over time if better information becomes available.

Next steps: how to optimize for search engines

Now that you understand the basics of how search engines work, you may see the value in optimizing for search engines. This is called SEO, or 'Search Engine Optimization.' By making sure search engines can find and automatically understand your content, you are improving the visibility of your site for relevant searches. This can result in more interested users coming to your site. Audit your site with Lighthouse and check the SEO results to see how well search engines can make users aware of your content.

How to measure search engine optimization with Lighthouse

The Lighthouse search engine optimization (SEO) audit scans your page, tests for things that matter to search engines, and gives you a score so you can see specific areas for improvement. SEO matters because it's how you get more relevant users viewing your content. If a search engine has trouble seeing your page, you're possibly missing out on traffic sources.

Audit your page with Lighthouse

Run Lighthouse on a page that's representative of the content that you want search engines to see:

{% Instruction 'audit-seo', 'ol' %}

Lighthouse generates a report for your page so you can see areas where you can improve SEO for your site.

To learn how to fix issues flagged by Lighthouse, see the SEO audits collection.

Next steps

It's worth noting that the audit doesn't cover everything you could do to increase visibility in search engines. You'll have to check the guidelines for individual search engines because they may have different requirements.

Here are some guides for specific search engines:

- Bing
- Google Search
- Yandex

There are also non-technical aspects that can improve visibility of your content in search engines, such as writing in the style of your target audience. Bottom line: make great content for the people you want to attract.

To learn more, check out Google's I/O talks:

- Build a successful web presence with Google Search
- Deliver search-friendly JavaScript-powered websites