Examining Google's Autocomplete Functionality

Before we begin building, we'll analyze Google search Engine's Autocomplete Functionality which is the component that we're aiming to build.
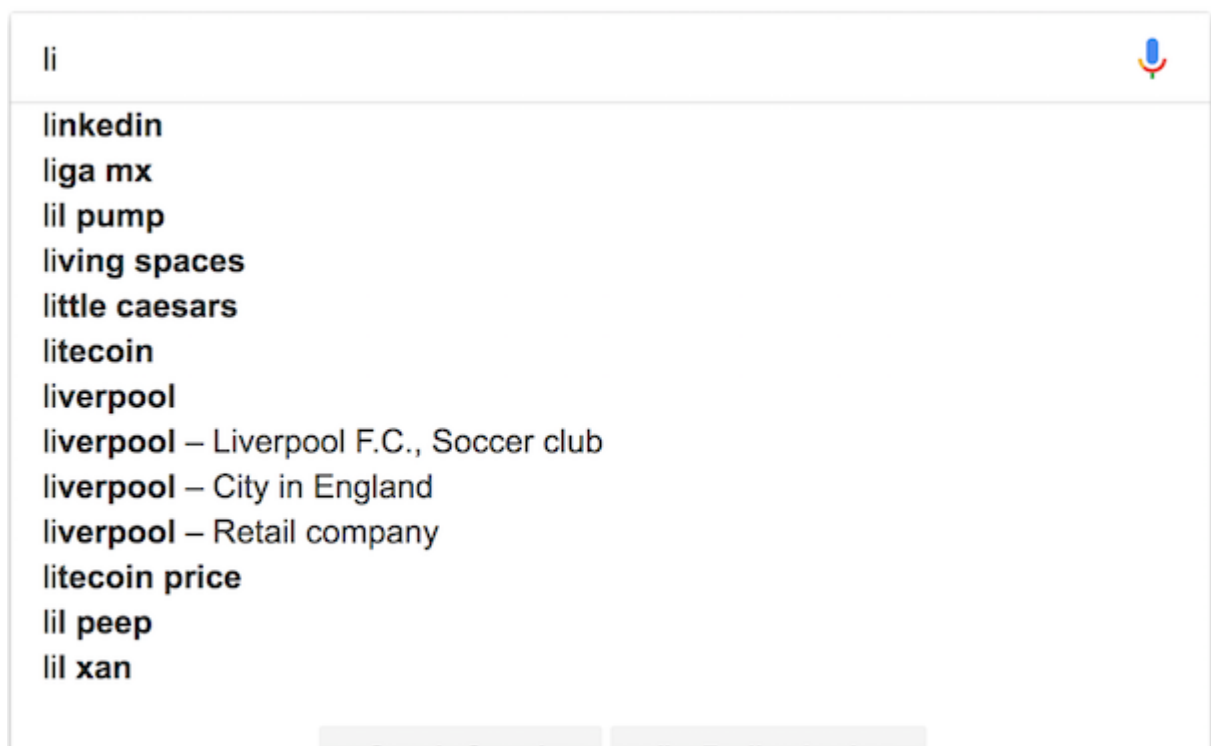
Let's do a little exploring to account for all the edge cases that aren't apparent at first, but might affect our implementation.

Case 1: #

When there exist multiple results of the same string, the detail is added after a hyphen

Case 2:

Results seem to assume for some autocorrect, and therefore won't always be prefixed by what's in the input bar. In such cases, the entire result is bolded.

Case 3:

Results are shown for every character input, including backspace, and only the most recent is shown. Google appears to abort outstanding requests if what's in the input bar is no longer the same. This is performance optimization we can ignore for our component.

Name	Status	Type	Initiator	Size	Time
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=1&gs_id=o&q=e&xhr=t	(canceled)	xhr	rs=ACT90oFp3mVg1xvL9vsR2X4Dvo...	0 B	690 ms
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=2&gs_id=s&q=ed&xhr=t	(canceled)	xhr	rs=ACT90oFp3mVg1xvL9vsR2X4Dvo...	0 B	779 ms
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=3&gs_id=w&q=edu&xhr=t	200	xhr	rs=ACT90oFp3mVg1xvL9vsR2X4Dvo...	258 B	2.02 s

This, by the way, is the Chrome devtools. Most popular browsers include some form of this tool to help developers look under the hood of the web page and see what the client is doing. If you haven't seen this before, I highly recommend familiarizing yourself with it, as it'll be an immensely useful part of your toolkit. This particular view is of the network tab. I have it opened while typing "e" then "d" then "u," with my connection artificially set to "slow 3G" (if I didn't slow down my connection, the responses come back too quickly – plus it's always good to test components for users with slower internet than the machines you're developing on!). The devtools is showing me that when I

typed "d" before the results for "e" came back, the request for "e" was

considered canceled – and likewise when I typed "u." Only the last request succeeded.

Case 4: #

There doesn't appear to be caching on the client-side for results, which means that if I input something that the server's already given me a response for, the client will still make another request.

Name	Status	Type	
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=1&gs_id=2z&q=e&xhr=t	200	xhr	!
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=2&gs_id=33&q=ed&xhr=t	200	xhr	!
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=3&gs_id=37&q=edu&xhr=t	200	xhr	!
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=2&gs_id=3a&q=ed&xhr=t	200	xhr	!
search?client=psy-ab&hl=en&gs_rn=64&gs_ri=psy-ab&cp=1&gs_id=3d&q=e&xhr=t	200	xhr	!

This view of the network is verifying the above. When I backspace to "ed" and then to "e," requests are still made even though they had already been made just moments ago. That makes things slightly easier for us!

Q   What scenario would we most likely want the client to cache results?

COMPLETED 0%

1 of 1   <   ✓

It's not always easy to decide whether your web app should cache responses from servers. Doing so increases the complexity of the app and may not be worth it. However, if your web app's user experience is negatively affected by slow response times to repetitive requests, the benefits of caching could be worth the extra complexity. This can be true when, for example, the server is nearing load capacity, or the responses are large and take a perceptible amount of time to download. Since Google's search responses are a few hundred bytes, and they probably have one of the highest capacity server architectures in the world, I can see why there's no need for caching.

To explain the other answers of the quiz: sensitivity of data is the reason for *not* caching, since saving things to your web browser's local storage exposes another surface area for attack. These responses can vary depending on time risk being outdated if cached, and the last answer is just a trick answer – it's not possible with regular HTTP.

Case 5: #

The buttons merge with the autocomplete results when they appear but are standalone below the search bar when the results are not present.

Now that we've properly scoped the component, let's start building it in the next lesson.