RFC 158 APPROVED: Code Intel UI

Editor: garo@sourcegraph.com
Status: Approved
Requested reviewers (please review by EOD 2020-05-15): Eric, Felix, Rob, Maria
Approvals: Thorsten, Nick, Joe, Eric

1. Background

This is a collection of a variety of code intel related UI improvements, whose background will be explained next to the problem. Most are fairly straightforward and the improvement itself doesn't require comment, but I'm collecting everything here to make it easier to figure out who will do what when.

There isn't really a process for getting a new UI facing code intel feature from design to implementation, which is part of the reason these improvements aren't going to make it into milestone 3.16. It's also unclear who should own investigating a bug in code intel UI. I understand Rob is working on addressing this from the design side, but want to note that this problem should be solved as part of success for this RFC (which might end up involving no work).

The process stuff is pretty orthogonal to all the concrete UI proposals in this document, but I'm including it here because solving the two problems is interrelated (answers to the process questions will determine who takes what actions proposed by this RFC, and trying to figure out who takes what actions proposed by this RFC will help us figure out process). Let me know if you'd rather I cut those sections into their own RFC.

2. Problems (in order of importance)

2.1 We don't encourage the user to add LSIF indexes to their repositories

The actual product could be the primary funnel for getting new users of precise code intelligence, and we aren't leveraging this opportunity. The info icon on search based code intel results is the only place we could do that, and it doesn't have a CTA.

2.2 Process around code intel UI is unclear

- Right now the code intel team only has 2 members, neither of which know much about our web code or web dev in general.

- Code intel bugs often surface in the UI and get sent to our team, which isn't always equipped to answer them.
- Our team also isn't equipped to make significant changes to the UI, which can cause what could be a single-person effort to get bogged down in cross-team communication
- Design process to be addressed by Rob separately

2.3 Users don't know what kind of code intelligence they're using

Currently the only UI providing this information is an info icon on hovers and reference results powered by search based code intel, which requires you to jump to another page to understand what's going on.

This information is also not currently complete: it's possible to have an imprecise definition and a precise hover text, yet there would be no icon indicating any imprecise data (as we don't have a way to display a badge for definitions).

2.4 Users are forced to leave the page unnecessarily

This most often happens in the browser extension, but in general information which could be presented directly to the user is often hidden behind buttons + page jumps. Specifically, I'm thinking about:

- Find references in the browser extension take you to dot-com instead of displaying the info in-line
- Go-to-def button doesn't differentiate between when it will go to a new page or not

2.5 The "back" button on the browser doesn't behave as expected

The pain of ending up on a new page is made even worse by the fact that the back button doesn't behave as I expect. Each explicit interaction on the page creates a new entry in my browser history, but I expect the back button to take me to the previous page, not the previous page state.

3. Proposals

3.1 The code intel team owns the entire stack

Sourcegraph uses mission based teams. Short term, this will require onboarding one or both of the members into our org's web stack, and borrowing design resources from the design team. Long term, we will ideally have expert web + design knowledge on the team, although having that for design will be very long term.

3.2 Hovers and file pages provide/overhaul UI to provide indexing status, differentiate code intel source, and encourage users to add/fix indexing

Several proposals here are grouped together as their design and implementation are heavily interrelated. I made a (very) rough draft of all these designs on Figma so you have a visual companion to this text (click the text bubble to see comments I left on the design).

3.2.1 Encourage users to add/fix indexing

Whenever LSIF indexing is easy and robust, we should be encouraging users to take advantage. I see this happening in the following places:

- Subtly, on files powered by search based code intel with experimental LSIF support
- Subtly, on files powered by precise code intel
    - Users should know why the cool thing is happening, and how to make it happen everywhere
- Subtly, on files powered by search based code intel when we don't know whether the user can do anything about it
    - This scope is users on dot-com that are not logged in with their GitHub credentials
- Aggressively, on files powered by search based code intel when the user could do something about it
    - This scope is all users of private instances, and users of dot-com whose GitHub credentials let us determine whether they have the appropriate repo permissions to upload indexes dot-com

This should happen at both the file level (e.g. before interacting with the page, there should be a UI element which accomplishes this task), and during each code intel interaction (hovers and find-refs).

3.2.2 Differentiate between code intel result sources and provide indexing status

- At both the file level and during each code intel interaction, we should indicate the following states of code intel on this file:
    - Loading
    - Basic code intel
        - Because no index uploaded

                - This corresponds to “files powered by search based code intel” in the previous section
            - Because index is processing
                - This should look like a loading icon for all users
            - Because index processing failed
                - This should look like an alert for users that can do something about it, and just info otherwise
        - Stale index
            - Because no index uploaded
                - This either indicates some kind of CI error, or that the user has stopped automatically uploading indexes.
            - Because index is processing
                - This should look like a loading icon for all users
            - Because index processing failed
                - This should look like an alert for users that can do something about it, and just info otherwise
        - Precise code intel
            - This corresponds to “files powered by precise code intel” in the previous section
- In the search results pane, precise and search based results should live in separate sections, rather than being bunched together with indicators
    - Search based results should be hidden by default and loaded on demand to increase responsiveness
- Make sure we address #10688 re flickering hovers when no intel is available

3.3 Provide as much information as possible to users without them leaving the page

- Go-to-def button needs to be fixed so it accurately displays when it will cause a page jump
    - On Sourcegraph, I’d like to know when I’ll be jumping to a new file vs staying in the same file
    - On the code host extensions, I’d like to know when I’ll be staying on the same page, jumping to a new page on the code host, or jumping to sourcegraph
- The code host extensions should be able to display find-references inline
- The site and code host extensions should highlight references to the hovered symbol in the currently viewed code file, to provide as much information about references as possible prior to any button-presses

3.4 Navigating backwards on Sourcegraph should only move back between “jump” actions

- A new browser history entry should be created only when navigating to a new file or taking a go-to action
    - Specifically, we should not create a new one when clicking on symbols, or on a find-refs action (but should when a reference is actually followed)
- When navigating backwards within the same page (which should only happen because we made a jump-to-def/ref within the same file), it is visually obvious where that jump came from and that a jump has occurred.
- Jumping to the same symbol that is being jumped from should not create a history entry

4. Definition of success

- Each relevant team’s handbook page is updated to reflect new process decisions made + existing undocumented ones
- All proposed features are implemented.
- Potential metrics for encouraging users to add LSIF indexing
    - This is already indirectly tracked by the LSIF WAU KR
    - Number of users navigating to LSIF docs from the new UI elements
    - We could track which users navigate to the LSIF docs, and then when we receive uploads cross-reference that log to determine success. This is probably too noisy and too much work to be worth it

Feedback

In a chat with Ryan about Code Intel results, he came up with the idea to elicit feedback from the user about the quality of the result. My (Rob) quick take on his suggestion was something like the following:

Rate our code intel quality: 👍 or 👎 .
After pressing an answer the question is replaced by a flash message like: "Thank you, we will review the issue.".