Take Home Assignment

Tags

Current Code

TL;DR

1. Download code from https://drive.google.com/drive/folders/1Ax-8EwtjYk_E0PWoIu6D5rzh4qJ-jGXe?usp=sharing.
2. See loom link here on how to get started: https://www.loom.com/share/77c82877c32941099a81939f59109848

Longer Discussion

1. Run the node.js server by running `npm run start`.
    a. It starts a http server for serving static html files and api endpoint
    b. It also starts a webSocket server, which Chrome plugin connects to
2. Run the chrome plugin in your browser by following these steps: https://bashvlas.com/blog/install-chrome-extension-in-developer-mode
3. Open any URL in Chrome with the plugin installed. Chrome plugin connects to the webSocket server at startup (and if webSocket is down, tries connecting every 5 seconds)
4. Move your mouse in the URL of choice, click some events, fill out some form etc.
    a. Current code only captures the last URL visited. So, clicking around is okay till it does not lead to change in URL
5. Now, open Chrome in incognito mode (or any other browser which does not have this plugin installed), and open `http://localhost:3000/rrweb-events?id=1`. You should be able to see a video, that captured any movements we did on the other Chrome with plugin installed

Context

1. What you don't see in this code is that the node.js server has a UI on top of it . That UI has a button that says start session button and stop session button
   (For the curious, but not relevant to assignment: The node.js code is part of Electron app. Electron allows us to bundle node.js server with a React based UI . VSCode, Slack, Discord, WhatsApp on Desktop are all Electron applications)
2. When a user clicks on start session button, it starts a new session on the server. A new session means a new sessionId being created
3. In the current version of code, Chrome has no notion of sessionId:
   → It simply sends the data and server stores it on the file system.
   → This means that even if the user has not clicked on start session button, Chrome plugin would simply connect to WebSocket and send data.

Problem statement

1. The problem statement defined here are real world problems we are facing, and this assignment is to give you a feel of the kind of problems you would be solving, if/when you join. There are some concepts that would be outside of what you have worked in the past - but that's the idea of this assignment. We have not implemented these solutions ourselves yet (but we have some ideas on how to)
2. That is totally okay if you are not able to solve all of them. The idea is not to see a working solution, but to discuss the approaches you took, what you learnt in trying to solve.

Part 1: Allow multiple URL's to be stored and retrieved

Look at the node.js code. It only allows to show one URL.

1. Can you expand on the code in node.js (this exercise does not require any changes to Chrome plugin) so that every URL is saved in its own file.
2. Make changes to the html page , so that by passing a query parameter like `?id=1` can show you the first URL visited. Passing `?id=2` shows the second URL visited
3. As part of this exercise, assume that the user would always open a new tab with a new URL. You can safely assume that the URL's we recieve on the backend are always unique

---

Part 2. Assign sessionId to the data sent by Chrome Plugin

TLDR; We want to associate the payload data sent by Chrome, to be associated with a sessionId. This sessionId is created on the server (when the user clicks on start session). The chrome plugin should send a sessionId field with every payload.

Longer Discussion

1. The user is creating a sessionId on the server, which means that the payload sent by Chrome does not have any sessionId in it.
2. The first ask is to pass this sessionId to Chrome plugin and the plugin sends data with sessionId in the payload
3. If there is no sessionId that the server has sent (ie. very beginning when user has not clicked on start session button), Chrome plugins should not send data
4. When the user clicks on stop session button, Chrome plugin should also stop sending data

Hints

We don't expect you to have worked in Chrome plugin before, so let us give you some pointers:

1. You might have to get familiarized to Chrome plugin workings: What is a service worker, and what is a content script. How do they talk to one another
2. Read about the advantages of webSocket server over http server, and why we chose webSocket server here. And how that. would be helpful

How can you test it

(You might have to change in the code to test it as well)

1. We have also added a html page `http://localhost:3000` which connects to the webSocket. You can modify that page to pass in sessionId to the node.js server
   Alternatively, you can use a tool like Postman to connect to this WebSocket and send data
2. Passing in a new sessionId simulates the case when a user has clicked on start session button
3. To simulate the use case of user clicking on stop session button, send `NULL` OR `-1` sessionId

---

Part 3: Check why the rrweb video does not work on existing tab, on changing sessionId

This might be a more challenging problem to solve, and would require you to understand some level of details of Chrome Plugin, as well as rrweb

TLDR; There is a bug that would happen, when we allow sessionId to change on an existing webpage.

Longer Discussion

1. Lets say we are on an existing tab in Chrome. Take any URL (e.g. https://www.esa.int/Science_Exploration/Space_Science/Webb/Webb_Hubble_confirm_Universe_s_expansion_rate)
2. Simulate the use case of start session button being clicked, by using the `http://localhost:3000` page approach listed in Part 1 (e.g. pass sessionId `2` here)
3. Simulate the use case of stop session button being clicked
4. Now simulate start session button again being clicked, this time with sessionId `3`
5. Do stop session button simulation now

What you will observe

1. You will see that the sessionId = 3 data does not show up in the rrweb events video part
2. And the ask is to figure out why does it not work. And then make it work

Hints

This assignment does not assume you have a working knowledge of rrweb, and Chrome Plugin, so let us give some pointers:

1. See how rrweb works by reading its documentation. Pay special attention to different types of events it sends in its type field

   → These series of blog post from the author of rrweb here: https://dev.to/yuyz0112/series/9333

   → There's a company called PostHog that also uses rrweb, and write very good blogpost. If official documentation does not help, try to see if those can help
2. In Part 2, you might have figured out how to capture sessionId in service worker, and how content scripts can use it
3. Now, you will have to figure out how service worker can communicate with content scripts, and how a service worker can connect with content script

---

Bonus Points, Part 4: Show sessionId in options page

If you can show the sessionId in Chrome plugin, in the options page. Options page is the one that opens when user clicks on the Chrome plugin icon