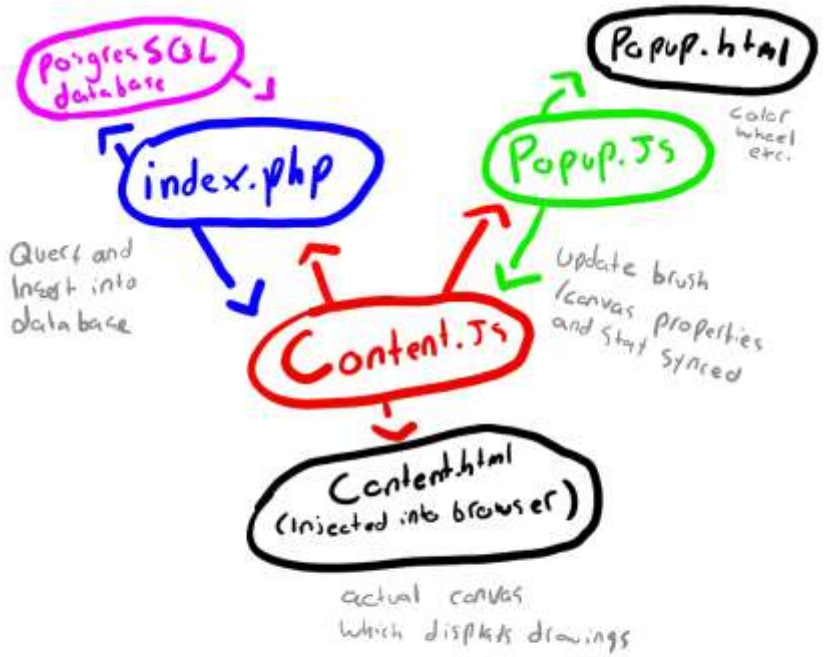
How the code works:
When you turn *Chrome Tagger* on, immediately chrome begins injecting custom html and scripts into every page you visit. The html takes the form of a fixed canvas element, scripts immediately ask a Heroku server via a GET request for any strokes associated with the tab's current hostname. The server processes the response, querying a posgreSQL database, returning the serialized result in the body of the response. The client then deserializes the data so that each stroke becomes its own object with a color and an arbitrary amount of vertices (each with a position and width). On every animation frame the canvas draws any deserialized strokes offsetting them based off the current scroll.

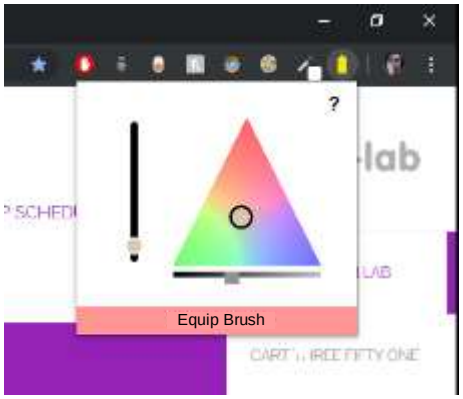
On the release of a mouse/stylus/finger press, on the completion of a stroke that stroke is serialized and POST-ed to the server to be stored appropriately in the posgresSQLdatabase. A custom serialization method was preferred over working with built in JSON serialization solutions because it was faster and I was able to more easily address bugs as I was making the thing myself.

When the extension's content is first injected into the client side webpage, it also messages the popup.js, asking it what colour and width it should set its brush to (if it exists). Whenever the user opens the popup.html, the popup.js asks the content.js what colour and width the brush is and matches those properties. Whenever the client changes these values on the popup.html, the popup.js sends a message to the content.js so it changes its brush/canvas properties appropriately.

In summary, in the project I used php and posgresSQL for the backend, and pure JS in front end.

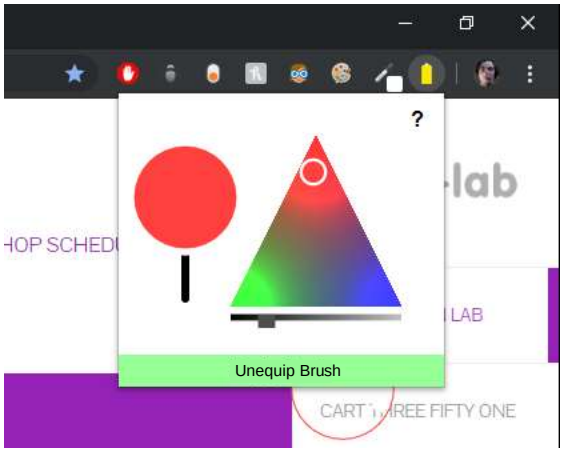


Documentation



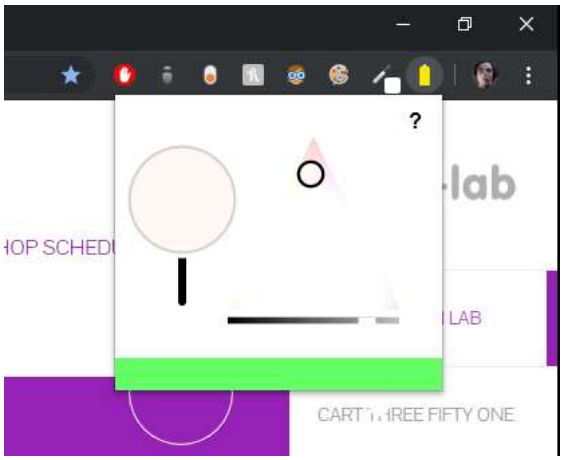
Shows the extension popup.

Users can select a brush color, size, enable and disable the brush, or select the “?” to be taken to a random already-drawn-on website.



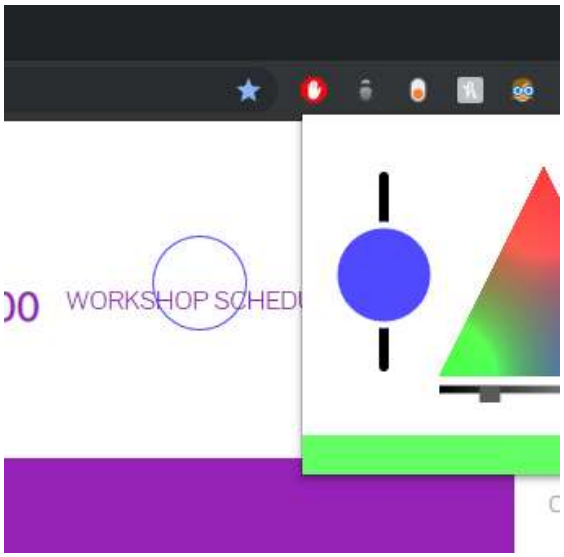
Shows the extension popup.

The brush is equipped, and a large red brush selected.

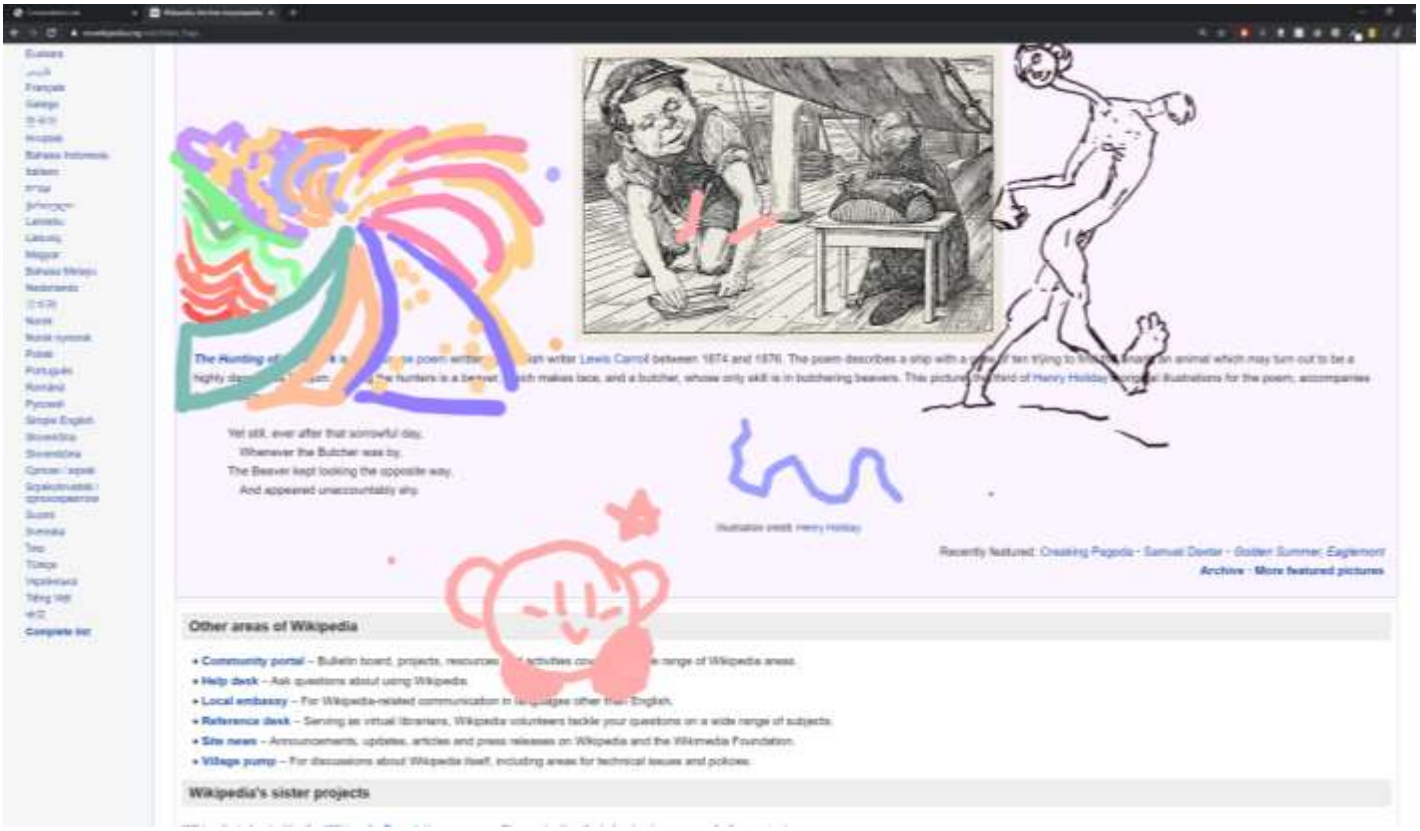


Shows the extension popup.

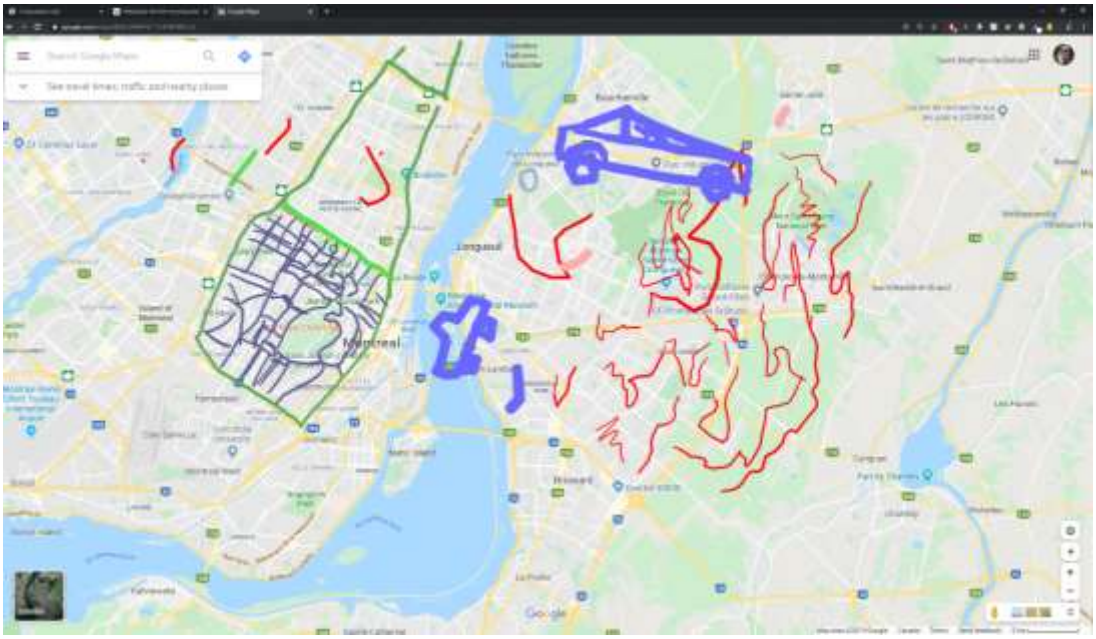
The brush size indicator automatically dons an outline to stand out from the white background when needed.



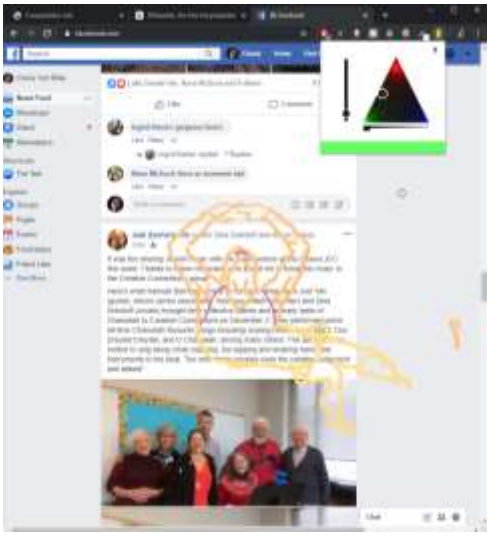
Shows the extension popup and the brush indicator.



Drawings on *en.wikipedia.org*



Drawings on *google.com*



Demonstrating that the app works with infinite scrolling via drawings on *facebook.com*

Demonstrating that the pressure sensitivity works while using a tablet.