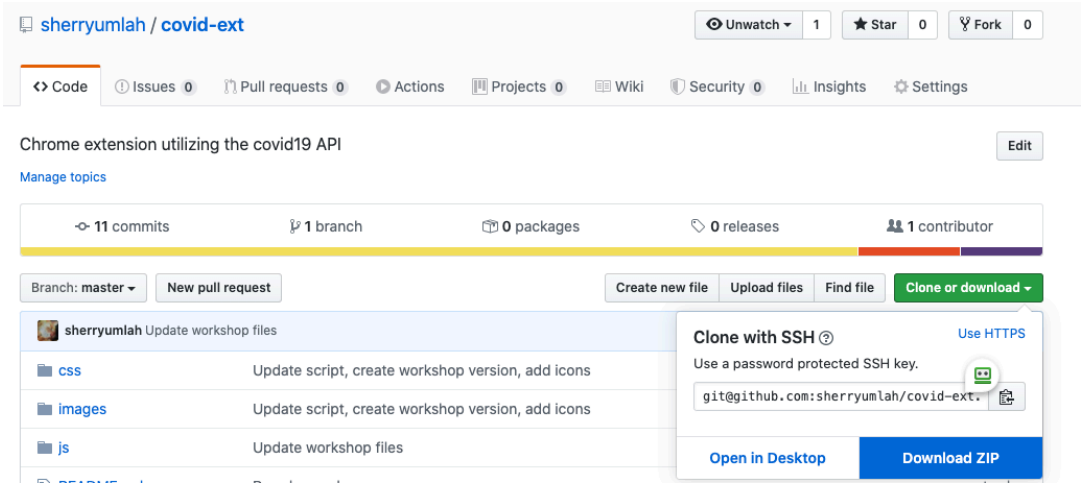
Covid-19 Chrome Extension Workshop Steps

Prerequisites:

- Chrome Browser installed
- IDE/Code editor of your choice installed
- Basic understanding of HTML, CSS, and JavaScript
- Download or clone the repo at: https://github.com/sherryumlah/covid-ext

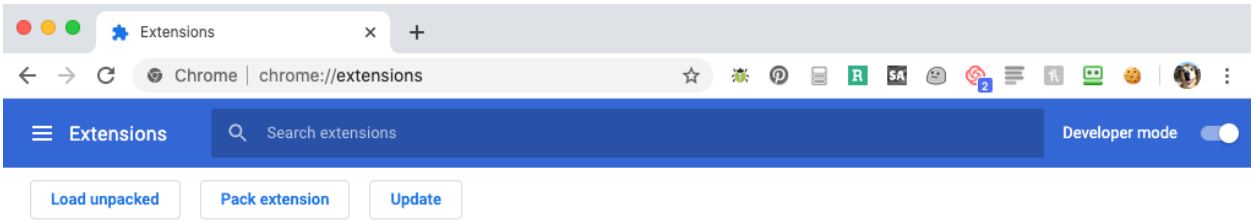


Pre-Workshop Set-up Instructions:

Install the Extension:

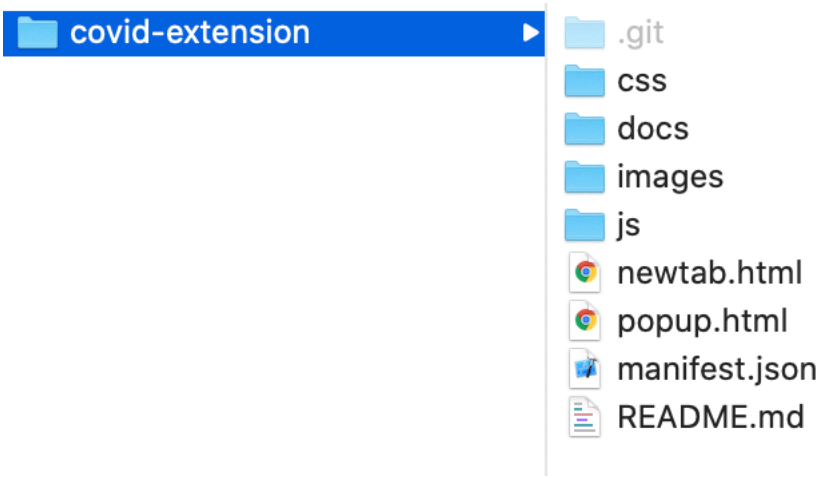
1. Download or clone the repo at: https://github.com/sherryumlah/covid-ext and familiarize yourself with the file structure and code.

2. After saving the github repository files to your local drive, type **chrome://extensions** in your browser's address bar and press the return key to navigate to Chrome's extensions page.



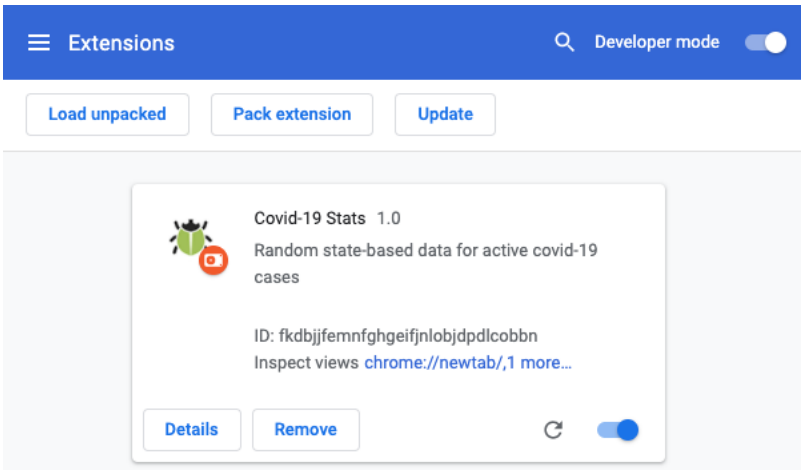
3. Ensure that the **Developer mode** toggle in the upper-right corner is toggled ON.

4. Click the **Load unpacked** button and select the extension's folder that you saved to your drive in step 1. Your main folder for the extension package might be named slightly differently than the image below. In the image below, the extension folder is named "covid-extension" and it contains all of the files that make up this Chrome extension:



5. With the extension folder selected, click the **Select** button.

The extension should now be installed!



Each time you open a new tab in Chrome, this extension will display a photo of a random state with accompanying Covid-19 statistics for the state.

You've completed the pre-workshop set-up! The workshop instructions below will be completed together on Zoom.

Workshop Instructions:

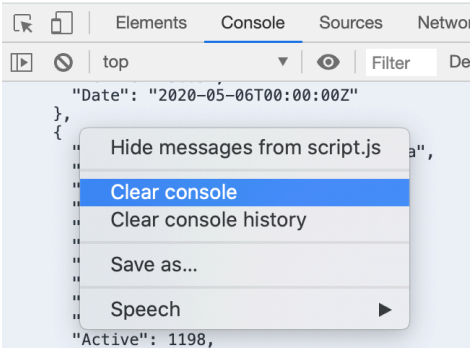
1. Rename script.js to script-final.js

2. Rename script-wip.js to script.js

3. Open a new tab.

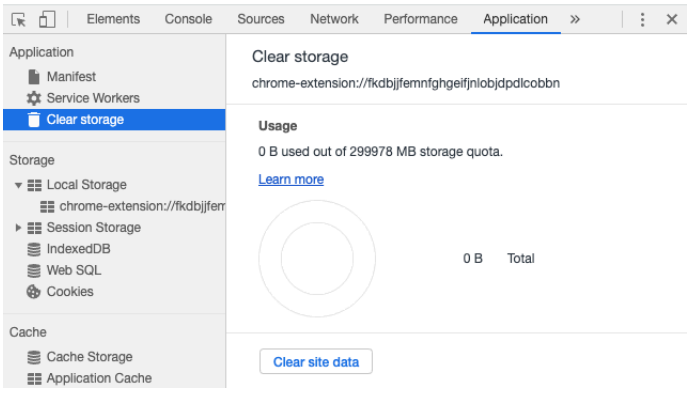
4. Open the **JavaScript Console.**
   This is accessible by right-clicking on the page and choosing **Inspect** and clicking on the **Console** tab or via the **View** menu:
   View > Developer > JavaScript Console.

5. Clear the console by right-clicking on the console pane and selecting **Clear Console.**



6. Navigate to the **Application** section. There should be no key/value pairs visible in **Storage > Local Storage**.

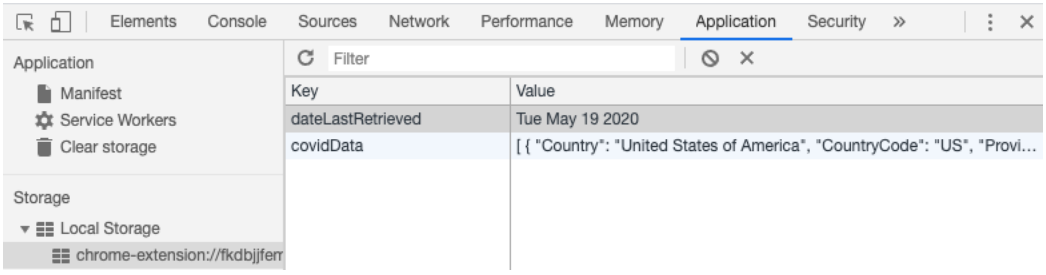
   Note: If you have any values stored here from previously running the extension, clear them now by clicking on **Clear storage** under the **Application** section and then clicking the **Clear site data** button.



7. Refresh the page and examine the console. We can see from the log that we're returning today's date. We're also looking at our local storage to find out when we last retrieved data. In this case the **lastRetrievedDate** key has a value of null because we cleared local storage and haven't fetched new data from the API yet. As a result, the **covidData** key also has a value of null. When we fetch from the API, we update these two keys by storing today's date in the **lastRetrievedDate** key and store our response results in the **covidData** key. Examining the Local Storage we see no key/value pairs because we haven't fetched any data yet.

8. Uncomment lines **128-130** and **lines 60-89**. **(Leave line 77 commented!)** In this block of code (lines 128-130), we're comparing the **lastRetrievedDate** in local storage with today's date and checking for the existence of a value for the **covidData** key in our local storage. If the dates don't match or we're missing a value for **covidData**, then we call the **getCovidData** function (lines 60-89), passing it today's date, which fetches data from the API. We take the response and store it as the value for our covidData key in local storage, then we call the **setLastRetrievedDate** function which stores today's date as the value for the **lastRetrievedDate** key in local storage (lines 44-47).

9. Save the file. Clear the console. Refresh the page in the browser. Examining the console we can see that we fetched the data from the API and saved the response and today's date to local storage. These key/value pairs now appear in the **Application** section under **Local Storage**. Example:



10. Now that the data exists locally, the program will fetch the data from local storage for the duration of the date. Uncomment lines **52-55** to make the **getLocalCovidData** function available to us. This function is called when we've already fetched API data for the day and it pulls the value for the **covidData** key from local storage and returns it.

11. Save the file and refresh the page in the browser. Examine the console and note that the program acts differently. The log suggests that today is still the same date – we've already fetched data for today – and it logs the data that is stored locally: an array of objects that contains covid data for specific states.

12. Now let's populate our page with this data. Uncomment line **77**, lines **101-120** and **131-135** to make the **displayData** function available to us so we can call it if we've already retrieved data from the API for the day.

    The **displayData** function takes a parameter - the **covidData** key value from local storage - and lets us parse it.

    Then, the program generates a random number and uses it to select an object from the array of stored API results.

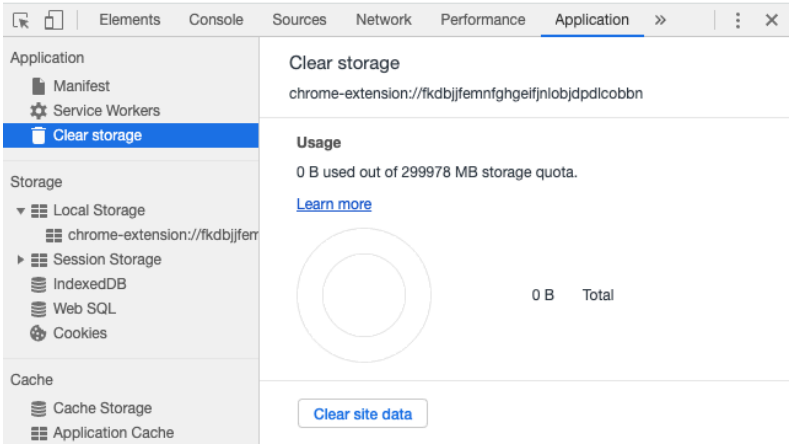
    It then adds commas to the object's Active and Deaths properties so the numbers are more readable.

    It also looks for a file in the local **images** folder of the extension package that matches the name of our object's **Province** property (with spaces stripped) and uses this as the background for the page's body element. *Example: "Rhode Island" will look for and render "images/RhodeIsland.jpg"*

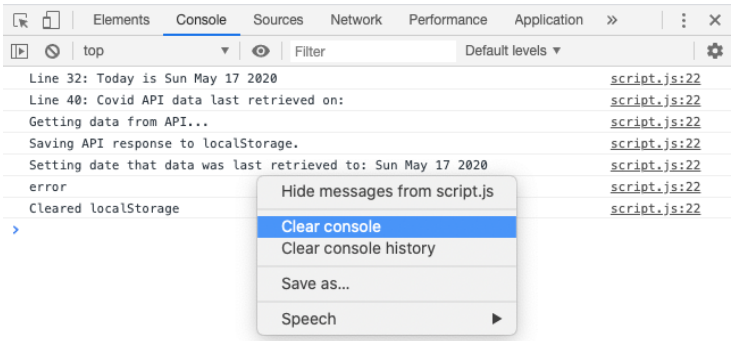
    Lastly, it inserts the object's properties into the html of elements on our **newtab.html** page, including the **Province** (state or location name), **Active** (active number of cases), **Deaths**, and **Date** properties.

    Now that all of the code is uncommented, we can see the extension fully in action!

13. Be sure any previously stored values are cleared by clicking on **Clear storage** under the **Application** section and then clicking the **Clear site data** button.

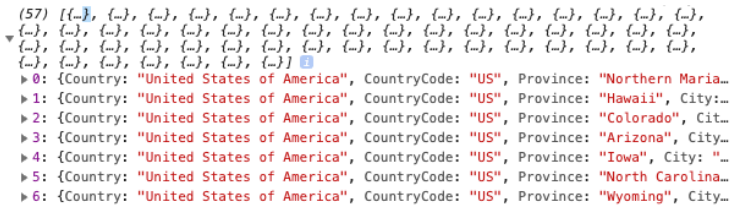


14. Clear the console by right-clicking in the console pane and choosing **Clear console** from the menu.



15. Refresh the page. Examine the console log and note that because we had nothing stored in local storage, we retrieved data from the API, saved the API response to local storage, set our locally stored **dateLastRetrieved** key value to today's date, then generated a random number to select a random object. Once we selected the object, we displayed its properties by updating the DOM elements on our newtab.html file.

You can expand the array of objects in the console to examine them by clicking on the arrow:



You can also expand the randomly selected object to view its properties. Example:

16. Now that we've retrieved and stored data locally, the program acts a little differently.

    For the duration of today, each time a new tab is opened, the program will compare today's date to the **lastRetrievedDate** value in local storage and determine whether it's a new day and time to pull from the API again.

    Refresh the page. Note that the console log reflects that API data was already retrieved for today, so the program skips fetching API data and pulls the data from local storage, instead.

    Tomorrow, the program will fetch new data from the API. Hopefully, tomorrow's Covid-19 stats will look brighter.

    Stay safe and be well.