Diary Entries

W9 (10 hours)
Goals:

1. Create the basic structure of a website in html.
2. Make sure that the format of the website is organised and intuitive through css styling
3. Identify what kinds of interactions I would like to have in my website (Researching for Js functions/events)
4. Identifying a topic, finding a study and getting it approved

What I have accomplished: (10 hours)
HTML:

- Created the basic structure of the website using a suitable template I found online. (https://github.com/russellsamora/scrollama)
- Input the content that I have curated for my data story, I also read up on different data story guides to help inform me of my next steps. (https://medium.com/kontinentalist/so-you-want-to-code-a-data-story-42c73443fa3/ https://pudding.cool/process/introducing-scrollama/)
- Link to correct CSS and JS files.

Errors faced:

1. The template that I used had all the code collated into one html file
   → I had to separate the css code and js code and link them to external .js and .css files correctly
   → I had to troubleshoot what unnecessary codes to remove, how to properly separate the codes and use the right script src and link href for css as some of them used external online sources.

CSS:

- Cleaned up and adjusted the dimensions of the various classes/sections present in the html.
- Selected and styled main elements such as header, footer, and main content area
- Changed colours, fonts, and design elements to enhance overall look of website

Errors faced:

1. The format of the website was too small, it was unable to fit the content that I wanted to present properly.
2. The fonts being used were linked to an external online source that I was quite confused about
   → So I changed the overall fonts used and removed the links to the external sources & used presets that were already there instead.

JS:

- Using the template, shift from utilising d3 to DOM selectors.
- Understanding how the various functions worked: scrollers/resize functions.
- Added eventlistener to certain functions to ensure proper dimensions are sent to the template library for it to work properly.

Errors faced:

1. I was unfamiliar with d3 selectors and felt more comfortable using DOM selectors, so I researched ways to be able to shift from one to the other.
2. I wanted to have my visualisations transition from one to another as the user scrolls through the webpage, but I am restricted by a certain function/css (I have yet to find) that allows me to do so.

W10 (5 hours)

Goals:

1. Finding the right charts for my data and inputting them.
2. Find out how to use the scrollama library to allow for a smooth transition between chart to chart when scrolling.

What I have accomplished: (5 hours)

HTML:

- I read up on the resources given in the mod to choose appropriate charts to use.
- Inputting the charts I created from (https://flourish.studio/)

CSS:

- Making sure that the charts that I input fit with the area that I had catered for them in the webpage by editing the dimensions of the different tags in html.

JS:

- Trying to transition from chart to chart as the user scrolls through the webpage using scrollama's functions. There is a fixed <figure> tag that is pinned to the

Overall errors faced:

- I was not able to make any changes to my code that will help input different charts into the <figure> tag in the html and have them show one by one as I scroll through my webpage. I tried these steps:
    a. Giving a separate <p> for each chart and giving each chart a unique class → updating the handleStepEnter() function in my JS code to dynamically update the <figure> tag with the appropriate chart based on the current step (progress of scrolling) by using a switch statement that checks the value of response.index and updates the <p> element with the corresponding class name. → Add CSS style for the specific class that I used with the JS functions to make the corresponding <p> visible.
    b. Giving a separate for each chart within the <figure> tag → updating the handleStepEnter() function in my JS code to update the content of the <figure> tag with an appendChild function → Edit CSS by editing the <figure> tag's position.
    c. In a last ditch effort, I tried to swap around the positions to make the charts stick and the text be relative to scrolling but it did not look presentable and clear for the user to read.
- Conclusion: I was not able to solve my problem with scrollama and I will be looking into consulting Prof/TA to advise me on how to move on with my project.
    a. Github page along with the respective html, css and js files: https://github.com/joshnerdg-nm2207/joshnerdg-nm2207.github.io
    b. Updated Fiddle:https://jsfiddle.net/45mqw9gy/1/

W11 (5 hours)

Goals:

- Finalising the data for my data story; Since I was using a study, I needed to find ways to tell a story through the huge amount of data present.
- Structuring my website and making sure that the interface caters to the type and length of data that I will be inputting in the following week.

What I have accomplished:

HTML:

- Input the IDs, structures and placeholders for the data that I have curated.
- Found out how overflow properties work
- Ensure a responsive: Utilisation of viewports

CSS:

- Made sure that the visual style and layout of website is coherent
    - Flexbox
    - Box model: Padding, positions, displays
    - Margins, height, widths

JS:

- I was still stuck on the same problems for my JS code

Errors faced:

- I was not able to make any changes to my code that will help input different charts into the <figure> tag in the html and have them show one by one as I scroll through my webpage. I tried these steps:
    a. Giving a separate <p> for each chart and giving each chart a unique class → updating the handleStepEnter() function in my JS code to dynamically update the <figure> tag with the appropriate chart based on the current step (progress of scrolling) by using a switch statement that checks the value of response.index and updates the <p> element with the corresponding class name. → Add CSS style for the specific class that I used with the JS functions to make the corresponding <p> visible.
    b. Giving a separate for each chart within the <figure> tag → updating the handleStepEnter() function in my JS code to update the content of the <figure> tag with an appendChild function → Edit CSS by editing the <figure> tag's position.
    c. In a last ditch effort, I tried to swap around the positions to make the charts stick and the text be relative to scrolling but it did not look presentable and clear for the user to read.

W12 (5 hours)

Goals:

- Solving my JS problems and moving on with the project.

What I have accomplished: (5 hours)

HTML:

- Inputting the charts I created from (https://flourish.studio/)

CSS:

- Making sure that the charts that I input fit with the area that I had catered for them in the webpage by editing the dimensions of the different tags in html.

JS:

- Transitioning from chart to chart as the user scrolls through the webpage using scrollama's functions.

Errors faced:

I was not able to make any changes to my code that will help input different charts into the <figure> tag in the html and have them show one by one as I scroll through my webpage. I tried these steps:

- Giving a separate <p> for each chart and giving each chart a unique class → updating the handleStepEnter() function in my JS code to dynamically update the <figure> tag with the appropriate chart based on the current step (progress of scrolling) by using a switch statement that checks the value of response.index and updates the <p> element with the corresponding class name. → Add CSS style for the specific class that I used with the JS functions to make the corresponding <p> visible.
- Giving a separate for each chart within the <figure> tag → updating the handleStepEnter() function in my JS code to update the content of the <figure> tag with an appendChild function → Edit CSS by editing the <figure> tag's position.
- In a last ditch effort, I tried to swap around the positions to make the charts stick and the text be relative to scrolling but it did not look presentable and clear for the user to read.

Solution: (Guided by Jeremiah)

Finding a way to transition between charts to charts → Utilisation of the concept of showing a chart and hiding the rest.

1. Separating the charts into different divs eg.

HTML:

```
<figure id="charts">
                <!--Chart 1: -->
                <div class="flourish-embed flourish-hierarchy" data-src="visualisation/13110848"
data-height="426px"><script src="https://public.flourish.studio/resources/embed.js"></script></div>

                <!--Chart 2: -->
                <div class="flourish-embed flourish-network" data-src="visualisation/13111412"
data-height="426px" ><script src="https://public.flourish.studio/resources/embed.js"></script></div>

                <!--Chart 3: -->
                <div class="flourish-embed flourish-radar" data-src="visualisation/13111312"
overflow="scroll"><script src="https://public.flourish.studio/resources/embed.js"></script></div>
```

1. Hiding all the charts first.

CSS:

```
figure div {
  display: none;
```

```
}
```

2. Creating two functions; 1 to show a chart, the other to hide the rest.

JS:

```
  const charts = figure.querySelectorAll('.flourish-embed');
//.flourish-embed refers to the class of my chart in html
  console.log(charts);
  function hideCharts() {
    for (let i = 0; i < charts.length; i++) {
      charts[i].style.display = "none";
    }
  }

  hideCharts();
//add on respectively to show/transition to more charts
  if (response.index === 0)
    charts[0].style.display = "block";
  else if (response.index === 1)
    charts[1].style.display = "block";
  else if (response.index === 2)
    charts[2].style.display = "block";
}
```

3. Fixing the CSS, so that the charts are able to be shown in full.

```
article {
  position: relative;
  padding: 0 1rem;
  max-width: 55%;
  /* height: 500px; */
}
figure {
  position: -webkit-sticky;
  position: sticky;
  width: 40%;
  top: 22% !important;
  height: 426px !important;
  right: 0;
  margin: 0;
  -webkit-transform: translate3d(0, 0, 0);
  -moz-transform: translate3d(0, 0, 0);
  transform: translate3d(0, 0, 0);
  background-color: #8a8a8a;
  z-index: 0;
}
```