NM2207

Session 11 Challenges

The challenges are due to be completed and submitted Saturday night of the same week.

# Overview of what we will do today:

* Function calls
* Buttons
* Instantiating and destroying charts
* Events and chart updates

**Warm up**

Before diving into the main tutorial, let's familiarize ourselves with some essential concepts.

1. **Events**

In JavaScript, events are actions or occurrences that happen in the browser, such as a button click, page load, or mouse movement. Developers can write code to listen for and react to specific events, allowing for dynamic and interactive web applications.

Common event types include:

* click: Triggered when an element is clicked.
* load: Triggered when a page has finished loading.
* mousemove: Triggered when the mouse pointer moves over an element.

To handle events in JavaScript, you can use the addEventListener method. For example:

**Text

Description automatically generated**

This code listens for a click event on the element with the ID exampleButton. When clicked, it logs "Button clicked!" to the console. You can even change the value property of the button after it’s clicked to reflect that something has changed.

**b. Fetch API**

The Fetch API is a modern, built-in JavaScript feature that provides an easy way to fetch resources (e.g., data, files) asynchronously across the network.

To fetch a resource, you can use the fetch() function, which returns a Promise that resolves to the Response object representing the response to the request. For example:

**Text

Description automatically generated**

In this example, the Fetch API is used to fetch JSON data from a URL. The fetched response is converted to a JSON object, and then the data is logged to the console.

1. **Instantiating a new Chart**

Creating a new Chart.js instance involves using the Chart constructor, which takes two arguments:

* The target HTML element (usually a canvas element) where the chart will be rendered.
* An object containing the chart configuration, such as chart type, data, and options.

When you instantiate a new chart, the Chart.js library automatically handles the rendering and updating of the chart based on the provided configuration. For example:

Text

Description automatically generated

In this example, a new Chart.js instance is created for a bar chart with three data points. The chart is rendered in the canvas element with the ID positive-bar-chart. The chartConfig object contains the chart type, data, and options, which are passed to the **Chart constructor.**

1. **Chart.update() function**

The Chart.js library provides the update() function, which allows you to update the data and configuration of a chart after it has been created. This is useful when you want to display new or modified data in an existing chart without recreating the entire chart.

To use the update() function, call it on an existing chart instance. For example:



In this example, a new label and data point are added to an existing chart. The update() function is then called on the chart instance to redraw the chart with the new data.

Finally, on the other hand, if we wanted to, we could “delete” the chart through a button by calling the destroy() function instead and then create a whole new chart by calling positiveChart = new Chart() once again. We are not using the “destroy” command today.

Now that you have a basic understanding of events, the Fetch API, and the Chart.update() functions, you should be better prepared to tackle the main tutorial.

**PART 1**

**Understanding the createCharts function and tracing the function call stack**

**Step 1: Analyzing the createCharts Function**

In VSCode, open main.js and main\_old.js, which is the completed code from Week 9. Begin by studying the createCharts function in the provided code. Take note of how it generates two horizontal bar charts using the Chart.js library. One chart displays the most positive phrases in the Singlish vocabulary, and the other displays the most negative phrases.

Write down your understanding of what the createCharts function is doing, and identify the following:

* How are the arrays for storing labels and data for both positive and negative words initialized?
* How is the data from the dataTable used to populate the arrays with appropriate labels (words) and data (sentiment scores)?
* How are the data objects, negativeDataObj and positiveDataObj, created for the charts?
* How are the negativeChart and positiveChart created using the Chart.js library?

**Step 2: Tracing the Function Call Stack**

Compare the provided code in main.js with main\_old.js, which is the completed code from Week 9. Trace the ideal function call stack by identifying the following:

* Map each group of statements in Week 9’s code to your Week 11 functions
* Now, based on how Week 9’s code did one thing after another, can you figure out what should be the order of function calls for your Week 11 functions? For example:
  + What other functions are called before or after the createCharts function?
  + How do the functions interact with each other?

**Part 3: Linking everything together: write the startLoad function**

Now that you understand how things link together, your task is to write the startLoad function. Use the example code of the Fetch API call from the warm up section to write this function. This function should:

1. Fetch the CSV data from the provided URL.
2. Convert the fetched response to text.
3. Call the processData function to process the data.
4. Call the createCharts function to create the charts.

**PART 2**

**Adding the More Words Button and Implementing the updateData function**

**Step 1: Add the More Words Button**

In this part, you will add a button to the HTML page, which will allow users to update the data displayed in the charts. Use the example code in the warm-up (if you don’t remember how to 😊) to add a button to your page.

**Step2: Write the updateData function**

Next, write the code for the updateData function. This function should:

1. Change the button text to "More Words please!" when clicked.
2. Increase the maxDisplayWords variable when clicked.
3. Call the updateChartData function for both negativeChart and positiveChart

**Text

Description automatically generated**

**Step 3: Link the More Words Button to the updateData function**

Finally, link the More Words button to the updateData function by adding an event listener. Use the example code in the warm-up (if you don’t remember how to 😊) to do so.

**PART 3**

**Understand and complete the updateChartData function**

**Here’s an algorithm which explains what’s happening in the updateChartData function.**

1. Start
2. Receive chart and type as input parameters
3. Based on the type parameter, set the threshold and startingIndex values
4. Initialize a loop, starting from the startingIndex and iterating over the data table

4.1. Check if the current word's sentiment score meets the threshold

4.2. Check if the number of labels in the chart is less than maxDisplayWords

4.3. If both conditions in 4.1 and 4.2 are met **(we will be writing this line of   
 code in Part 3)**:

4.3.1. Add the current word's label to the chart's labels array

4.3.2. Add the current word's sentiment score to the chart's data array

4.3.3. Update the corresponding index (lastNegativeIndex or lastPositiveIndex)

5. End loop

6. Call the update() method on the chart instance. ***(we will be writing this line of   
 code in Part 3)***

7. End

You can use this textual representation to create a visual flowchart using a flowchart creation tool or by drawing it on paper.

**Here’s a detailed explanation, if you prefer things that way:**

* The function takes two parameters: chart and type. The chart parameter is the Chart.js instance that needs to be updated, and the type parameter is a string, either "neg" or "pos", representing whether the chart is for negative or positive words.
* Based on the type parameter, the function sets the threshold and starting index for updating the chart data. If type is "neg", the threshold is set to the negative threshold, and the startingIndex is set to the last negative index. If type is "pos", the threshold is set to the positive threshold, and the startingIndex is set to the last positive index.
* The function then iterates over the data table, starting from the startingIndex. It checks whether the current word's sentiment score meets the threshold and whether the number of labels in the chart is less than the maxDisplayWords.
* ***If the conditions in step 3 are met, the function adds the current word's label and sentiment score to the chart's labels and data arrays.***
* The function updates the corresponding index (either lastNegativeIndex or lastPositiveIndex) to keep track of the last processed word in the data table for the next update.
* **Finally, the function calls the update() method on the chart instance. This redraws the chart with the updated data and labels.**

**Here’s what you have to do:**

1. Using the clues in bold above, fix the updateChartData function that updates the chart data and labels. Use the example snippet in the warm up code, and add them somewhere in the function.
2. Use the xx.update() method to redraw the charts with the updated data. Where would you add this command, and what is the “xx”?

**Reflection:**

1. What was the easiest Part in today’s tutorial? Why do you think so?
2. What’s the hardest Part in today’s tutorial? How can you/we make this more intuitive?
3. Does your final project require you to use events? Why or why not?