**Step 1** move the array of objects (tweets) into data.js

Do the imports/exports and add type=”module” to the html in the js script section

**Step 2**. In the useless chatbox example we added a textarea with id….we took the id called chatInput and called the .value….chatInput.value…this way we can get any text being typed….we also cleared out the text to an empty string after the button is clicked…..chatInput.value = ‘’

Also inside the css we set the resize value to none in the textarea property…this way users cannot physically resize the textarea box.

**Step 3**. Lets take the data from the data.js file and pass it via a for of loop and into an html string. We’ll also call the getFeedHtml() and do a console.log(feedHtml) to make sure that we are receiving the data in a string…NOTE that the feedHtml has a **+=** ….this is b/c we have multiple tweets to pass over and over

We begin by setting a let variable of feedHtml to an empty string, then we created our for loop, next we took our feedHtml variable and set template strings `` and inside we passed the html to be rendred with the variables replaced and the array set to .length for the replies variable as we want the length of the replies…(some may have 0 tweets other may have hundreds….so we need .length)

function getFeedHtml() {

  //Job of fuction to iterate through data.js and create HTML from each tweet with the boilerplate code which we will replace using template literals

  let feedHtml = "";

  for (let feed of tweetsData) {

\*/

    feedHtml += `

<div class="tweet">

    <div class="tweet-inner">

        <img src="${feed.profilePic}" class="profile-pic">

        <div>

            <p class="handle">${feed.handle}</p>

            <p class="tweet-text">${feed.tweetText}</p>

            <div class="tweet-details">

                <span class="tweet-detail">

                    ${feed.replies.length}

                </span>

                <span class="tweet-detail">

                    ${feed.likes}

                </span>

                <span class="tweet-detail">

                    ${feed.retweets}

                </span>

            </div>

        </div>

    </div>

</div>

`;

  } //END for of function

//   console.log(feedHtml); we did a test with console now we will just return feedHtml

return feedHtml

} //end getFeedHtml function

DIFFERENCE WHEN USING += AND = FOR THE FOR OF LOOP

Remember in our cat emotions app we used the for of loop when rendering a single cat image inside the modal and we used the equal sign alone b/c we are only passing in one pic

function renderCat() {

  const catObject = getSingleCatObject();

  memeModalInner.innerHTML = `<img

        class="cat-img"

        src="./images/${catObject.image}"

        alt=${catObject.image}

        >`;

  memeModal.style.display = "flex";

replace for of loop to the forEach loop

tweetsData.forEach(function (feed) {

**NOTE**  IF YOU ARE TYPING OUT THE forEach make sure you capitalize the **E**  in forEach…….you forgot to capitalize and it kept giving you an error that drove you crazy ....... tweetsData.foreach it was displaying not a function and it’s hard to tell that **foreach** needed be **forEach**

**Now we are going to render out the tweets to the html**

1. We need to create a new function for separation of concerns. We’ll call it render()
2. Inside this new function we just want to parse or render the tweets that we will call from the function from above getFeedHtml() and we will render it out to the id=”feed” in the HTML.
3. function render() {
4. document.getElementById("feed").innerHTML = getFeedHtml();
5. }

//call render()

render()

ADDING ICONS WITH FONTAWESOME

<https://cdnjs.com/libraries/font-awesome>

search page of fontawesome

<https://fontawesome.com/search>

DATA ATTRIBUTES

Let’s say we have an image with icons contained all inside a div img-container

And we want to access the share or heart icons to add a heart or share by clicking those icons. We need to actually use data-attributes in javascript. The data attributes is used by adding the keyword “data-“ ……so the word data with a hyphen…the next word that follows after the hypen is basically it’s identifier…so in JS when logging the console or actually selecting that property you would use:

e.target.dataset.share

If you look below at the html…share is followed after the hypen so .share is used to identify the share icon. Note that this is followed under the event or e property.

And after the identifier ie: share, heart it’s followed by =”” and inside the quotation marks you put the id of the image which here is “image-1”

  <div class="img-container">

        <img src="dino2.jpeg" alt="Man in front of dinosaur" id="image-1" />

        <div class="social-icons-container">

          <i class="fa-solid fa-heart"></i>

          <i class="fa-solid fa-share" data-share="image-1"></i>

        </div>

      </div>

Final result:

document.addEventListener("click", function (e) {

  if (e.target.dataset.share) {

    console.log(e.target.dataset.share);

  }

    else if (e.target.dataset.heart) {

    console.log(e.target.dataset.heart);

  }

We set up our data attribute in the twimba app to identify each user by their uuid. We did three ways for the reply, like, and retweet

 <div>

            <p class="handle">${feed.handle}</p>

            <p class="tweet-text">${feed.tweetText}</p>

            <div class="tweet-details">

                <span class="tweet-detail">

                <i class="fa-regular fa-comment-dots" data-reply="${feed.uuid}"></i>

                    ${feed.replies.length}

                </span>

                <span class="tweet-detail">

                <i class="fa-solid fa-heart" data-like="${feed.uuid}"></i>

                    ${feed.likes}

                </span>

                <span class="tweet-detail">

                <i class="fa-solid fa-retweet" data-retweet="${feed.uuid}"></i>

                    ${feed.retweets}

                </span>

            </div>

If you notice that after the attribute the uuid has been linked with feed.uuid……remember that the feed is from the function that is accessing the data.js file with tweetsData

 tweetsData.forEach(function (feed) {

    feedHtml += `

<div class="tweet">

Now we will setup an event listener to actually get the users tweets.

First we will access the document object and listen out for “click” also we need to use the event (e) inside the function…We then will access the event.target.dataset.like……

document.addEventListener("click", function (e) {

  console.log(e.target.dataset.like);

the goal is to access likes and since our data attribute is linked to our uuid from the above notes

  <span class="tweet-detail">

                <i class="fa-solid fa-heart" data-like="${feed.uuid}"></i>

                    ${feed.likes}

                </span>

We can use the dataset attribute of dataset.like…….this will identify the uuid as we have targeted it already…note if we click the like or heart icon we will return the uuid for that user…if we click anywhere else on the page it will return undefined…

document.addEventListener("click", function (e) {

  console.log("like", e.target.dataset.like);

  console.log("retweet", e.target.dataset.retweet);

here we just added the retweet attribute so if we click the retweet icon, the uuid for retweet will display and the first part for the “like” will be undefined and same goes if you click the “like” the uuid will display and the “retweet” will be undefined.