Autopopulation

In this lesson we learn how to dynamically load more tweets.

WE'LL COVER THE FOLLOWING

Scroll action listener

Let's add an event listener to the scroll action, and when it reaches near the bottom of the page (say, 100px above), we'll use that as a trigger to load. Here's where the states will be used. If the state is in pending, we know not to trigger another request.

```
function onScroll(event) {
  const scrolledTo = window.innerHeight + window.pageYOffset;
  const scrollLimit = document.body.offsetHeight;
  const scrollThreshold = 30;

  if (scrollLimit - scrolledTo <= scrollThreshold) {
    // TODO update
  }
}


window.addEventListener('scroll', onScroll);
```

We can't quite yet do the update, since we don't have the lastTweetId.

In the loop for processing API responses, we'll add this line.

```
lastTweetId = tweetResponse.id;
```

Processing for new tweets now needs to be additive, so we switch

```
document.body.innerHTML = tweetsHTML;
```

to

```
document.body.innerHTML += tweetsHTML;
```

And if the component is pending, we do nothing. This gives us an onScroll function that looks like this:

```
function onScroll(event) {
  if (isComponentPending()) {
    return;
  }
  const scrolledTo = window.innerHeight + window.pageYOffset;
  const scrollLimit = document.body.offsetHeight;
  const scrollThreshold = 30;

  if (scrollLimit - scrolledTo <= scrollThreshold) {
    const params = {
      pageSize: DEFAULT_PAGE_SIZE,
      sortOrder: DEFAULT_SORT_ORDER,
      lastTweetId
    }
    api.get(HOST + 'tweets', params, onNewTweets);
    setPending();
  }
}
```

Test it out!

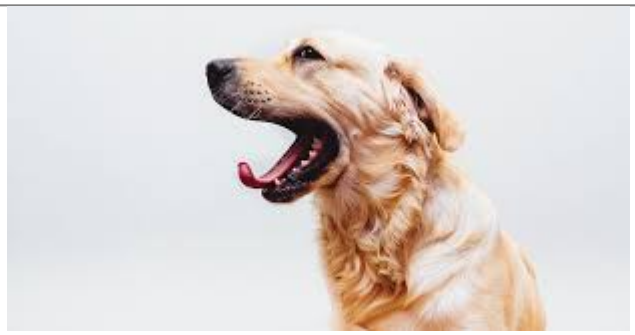
Output

JavaScript

HTML

CSS (SCSS)

Here I am... Loading...



If you were patient enough to make it to the bottom (or just astute enough to

have thought of it!), you'd notice that there's sort of a bug. When there are no more elements to serve, the client keeps trying to fetch incessantly.

By the way, this is a good way to accidentally DDOS yourself. If you're not familiar, it just means that the server gets so many requests that it gets overloaded and can't do its job anymore. If you have many clients encountering this bug, they'll be making a disproportionate amount of requests compared to other clients.

Now, I don't know how Twitter handles this because their data is pretty much near-infinite, but we can take a stab at it. I think to give users the best experience, we should set a timer when we get a response with no data, and while this timer is going, we cease all attempts at populating more data. That way, users get to enjoy the content at the bottom, and if they're persistently waiting for more content, they'll eventually get it – in theory.

Q Which change do you think works best?

COMPLETED 0%

1 of 1

Output

JavaScript

HTML

CSS (SCSS)

Show Hint

I’ll leave the implementation to you if you’d like to give it a shot. In case you want the solution, please click the **Show Solution** button above.