useQuery Hook

In this lesson, we create a hook to handle queries.

WE'LL COVER THE FOLLOWING

- useQuery
- Next

useQuery

A query in our app is to filter ToDo items. This hook returns two functions, `setQuery` and `getQuery`

`setQuery` is a function to update `query`, while `getQuery` is a function to read `query`.

```
// ./hooks/useQuery.js

import { useCallback } from 'react';
import { useTrackedState, useSetDraft } from '../store';
export const useQuery = () => {
  const state = useTrackedState();
  const getQuery = () => state.query;
  const setDraft = useSetDraft();
  const setQuery = useCallback(
    query => {
      setDraft(draft => {
        draft.query = query;
      });
    },
    [setDraft],
  );
  return { getQuery, setQuery };
};
```

There is an important note here: The reason we use `getQuery` is because we

used React Tracked. If we just used the bare React context, we could simply do the following:

```
return {
  query: state.query,
  setQuery,
};
```

If we do this with React Tracked, `query` is always marked as “used” when using this hook because it accesses the `query` property of the state inside the hook.

Another recommended pattern is that `useQuery` only returns `setQuery` and is named `useSetQuery`. Another pattern could be that you use `state.query` directly in components.

Next

In the next lessons, we learn how to create components with the hooks we created.