Предлагаю коллегам решение на основе Наблюдателя пересечения (Intersection Observer). Иван сделал недавно урок по 2-м наблюдателям: MutationObserver и ResizeObserver. Правда он откровенное слабый и без примеров.

Наблюдатель пересечения может отследить момент пересечения целевым элементом (за которым наблюдает) другого элемента. Если в настройках не указать другой элемент – будет появление в окне браузера. Таким образом реализуется бесконечный скролл, ленивая загрузка и т.д. Ссылка на видос с примерами внизу.

А вот код.

Создаем пропс компонента:

observer = new IntersectionObserver(([entry], observer) => { - объект наблюдателя

if (entry.isIntersecting) {

this.observer.unobserve(entry.target);

this.getCharList(this.state.offset);

}

}, {}); - на вход получает настройки, здесь мы их не делаем – получится при появлении внизу экрана

в параметрах стрелочной функции по передаются записи наблюдателя (но здесь она одна, поэтому просто пишем [entry] (см. видео по ссылке ниже) и сам наблюдатель (здесь не используется). саму запись срабатывания можете законсолить и посмотреть структуру.

далее проверяем, что объект пересек (появлися внизу) и тогда удаляем наблюдатель с элемента, а потом вызываем следующие 9 персонажей.

теперь к хуку окончания обновления компонента прикрутим запуск наблюдателя:

componentDidUpdate(prevProps) { - нам понадобится предыдущий пропс для условия

const lastItem = document.querySelector('li.char\_\_item:last-of-type'); - привязываемся к последней карточке героя в списке

далее комплексное условие:

if (lastItem && - элемент есть на странице

!this.state.newItemLoading && - мы не стадии загрузки

this.props.charId === prevProps.charId && - это не клик по персонажу (для панели Инфо)

!this.state.charEnded) { - персонажи не кончились

this.observer.observe(lastItem); - навешиваем наблюдатель на последний элемент

}

}

Да, здесь для условия клика я в пропс списка передал сверху тот же charId, что и передается после клика в панели Инфо и сравниваю старый пропс с новым.

Если не вешать условия по клику и окончанию персонажей и по подгрузке, тогда получим либо кучу зацикленных запросов (это режим подгрузки) либо множество запусков наблюдателя (хотя при этом работает, но может вызвать утечку памяти и неэтично это )

Во всей этой истории кнопка становится бесполезной…

Вот видео с 4 кейсами на русском по Intersection Observer: <https://www.youtube.com/watch?v=ZYqBZmU-tA0&list=PLwh3ORn7W8cbyAa2knA2d1BHH3sZ8K-gK&index=5>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

второй ответ:

Мой кейс (он же первый реально практический проект) был такой.

Нужно было обработать две таблицы Excel и результаты записать в новый файл.

Для работы с таблицами есть несколько библиотек, я использовал **ExcelJS**.

Для работы с таблицей сначала создается соответствующий переменная-объект этой библиотеки **new Workbook()**. Он получает структуру, но данных листов таблицы в нем еще нет.

Затем я с помощью **FileReader()** считываю данные из файла таблицы в переменную как буфер (потому что из под браузера считать как файл не получится).

Так вот, несмотря на выполнение метода onload данные из файла в мою переменную-объект записываются не сразу, а код обработки уже пытается прочесть данные, которые еще не записались из буфера в переменную-объект.

Я пытался делать **async/await**, потом ставить **.then()** - ничего не помогало – дальнейшие вычисления шли с ошибкой попытки прочитать пустой массив данных листов таблицы. Да еще как там этот **FileReader()** срабатывает под капотом, я не знаю…

Костыль я сделал с помощью таймаута: через задержку запускалась обработка, но если поставишь немного, то большая таблица не успеет считаться, а поставишь много (3 сек) пользователь будет ждать…

Так и пришла идея контролировать длину массива с листами таблицы в переменной. Как только она стала отличной от нуля – запускаем обработку.

Идею, естественно, не придумал, а нашел. Повторю, геттеры и сеттеры я не хотел пользовать, хотелось что попроще.

Код такой:

**function doStuff() {**

**if (workbook.\_worksheets.length === начальная длина ) {**

**setTimeout(doStuff, 50);**

**return;**

**}**

// код обработки таблицы

**}**

**doStuff();**

Надеюсь, ход моих мыслей понятен )