Задачи

Написать простой пример бесконечной прокрутки (infinite scroll)

Внешний вид результата

Это должна быть лента небольших карточек, каждая из которых содержит данные о конкретном пользователе (имя, фотография, электронная почта) с публичного API https://randomuser.me

Вам нужно собрать ленту, в которую добавляются новые записи, когда пользователь досмотрел её до конца. Рендерить ленту по-частям (т.н. "virtual scroll", используется для снижения потребления ресурсов) не нужно.

Стек

- голый VueJs или ReactJS, естественно, без внешних библиотек.
- абсолютно любые пре/пост-процессоры, UI-киты, CSS-фреймворки, инструменты разметки
- любой сборщик на ваш вкус, но хотелось бы посмотреть как вы конфигурируете vite или webpack
- eslint

На что мы будем смотреть

- На производительность по процессору/памяти
- На экономию трафика и минимизацию количества запросов
- На алгоритмическую сложность. Если вы выполняете полноценную итерацию по большому массиву записей (напр., *.map( ).filter( )*) при каждом его изменении - это очень не очень.
- Предполагается, что вы умеете собирать каркас vue-проекта самостоятельно, так что следы от vue-cli, react-scripts, webpack-cli, npx и прочих радостей жизни дадут очень жирный минус. Конечно, мы их используем в работе, но вам пока нельзя.
- Если цветовая палитра результата выжжет нам глаза - мы вас тоже не возьмём =)

Задокументировать логику в комментариях к коду, используя синтаксис jsdoc

Что писать в комментариях

- Предназначение переменных и методов в вашей реализации. Лучше на английском.
- Сигнатуры методов (типы принимаемых аргументов, типы возвращаемых значений)
- Типы переменных при объявлении. Это могут быть пользовательские типы (в т.ч., Ваши), такие как, например, тип Response описывающий ответ сервера, возвращаемый методом .fetch( )

```
/**
 * @type {Response}
 * @description Raw data which we got from the rest-api,
 * it is able to convert to JSON
 */
const response = await fetch( input: 'https://some-api-url.org/***')
```

В примере выше ссылка на тип позволяет читающему быстро ознакомиться с частью содержимого возвращаемой структуры - это мы отвечаем на вопрос "зачем писать столько комментариев"

```
/** This Fetch API interface represents the response to a request. */
interface Response extends Body {
    readonly headers: Headers;
    readonly ok: boolean;
    readonly redirected: boolean;
    readonly status: number;
    readonly statusText: string;
    readonly trailer: Promise<Headers>;
    readonly type: ResponseType;
    readonly url: string;
    clone(): Response;
}
```

Если при запуске jsdoc в строгом режиме (флаг "--pedantic") по вашим комментариям будет без ошибок собираться документация - это хороший плюс. Можете попросить знакомого junior-разработчика прочесть код - если ему что-то не понятно - лучше лишний раз проверить плотность и информативность комментов перед отправкой.

Положить в репозиторий на github / gitlab / bitbucket

Если репозиторий закрытый - обратиться к нам, скажем, кому дать доступы.