ToRead list. Постановка задачи, примечания к реализации и полезные ссылки

### Постановка задачи

Необходимо реализовать приложение ToRead List, которое включает в себя следующий функционал.

1. Поиск книг по запросу в электронной библиотеке Open Library.
2. Отображение списка найденных книг страницами по 100 книг с пейджингом (книги грузятся порциями/страницами по 100 штук и есть возможность перехода к следующей и предыдущей странице из 100 книг). Скролл списка в левой части приложения (см. прототип) должен быть организован таким образом, чтобы верхняя панель (с инпутом и GO) и нижняя серая панель с информацией по выборке оставались неподвижны и были видны всегда.
3. Просмотр детальной информации по книге.
4. Добавление книги в список на прочтение (который хранится в в Local Storage). При добавлении необходимо проверить, что данная книга уже не добавлена в список.
5. Удаление книги из списка на прочтение.
6. Возможность отметить книгу как прочитанную.
7. Приложение должно быть устойчиво к изменению размеров браузера (рекомендуем использовать flex)
8. задание "со звездочкой" - infinite scroll вместо пагинации (без использования сторонних компонентов)
9. задание "со звездочкой" - Динамически поиск (без нажатия на кнопку Go, обязательно использовать debounce)

Прототип -- <https://www.figma.com/file/CLtbqhaZvsjd4blSB9C2aB/To-Read-List?node-id=2%3A2>

В левой части приложения отображается список книг, найденных по запросу. В центральной части - подробная информация по выбранной книге. В правой части экрана отображается список книг из Local Storage.

**Общая структура приложения должна совпадать со структурой прототипа**. В то же время стили и цвета менять можно и возможно даже нужно, поскольку внешний вид приложения также имеет значение и будет учитываться. Кликабильные ссылки можно заменить на кнопки. Кликабильные ссылки не должны перебрасывать юзера на другую страницу.

По умолчанию на месте списка отображается пустое пространство. Серая панель отображается, однако кликабильных ссылок “Prev results”, “Next results” и текста с информацией по объему выборки (“Found: N”, “Start: M”, “Page size: 100”) нет.

По вводу запроса в инпут вверху левой части приложения и клику на “GO” отправляется GET запрос вида https://openlibrary.org/search.json?q=<input.value>&page=1. Книги грузятся пачками/страницами по 100 штук. По получению результата отображаются все книги из результата (их количество не превосходит 100) и информация по выборке (“Found: <result.numFound>”, “Start: <result.start>”, “Page size: 100”). Если в ответе numFound больше 100, то отображаются кликабильные ссылки “Prev results” (отобразить следующую страницу/пачку объемом 100 книг) и “Next results” (отобразить предыдущую страницу/пачку объемом 100 книг), по клику на ссылку отправляется и обрабатывается запрос вида https://openlibrary.org/search.json?q=<input.value>&page=<n> с соответствующим n.

По клику на любой элемент в списке в левой части приложения выводится детальная информация по книге. При этом необходимо выделять выбранную книгу (на прототипе у выбранной книги изменен фон и название выделено жирным).

По клику на ссылку “Add book to Read List” происходит добавление книги в Local Storage. При открытии приложения в списке “To read list...” должны отображаться книги, уже содержащиеся в “Local Storage”. В заглавии отображается количество книг и количество прочитанных книг. Клик на “Remove from list” удаляет книгу из списка, клик на “Mark as read” должен приводить к тому, что книга отображается как прочитанная (в т.ч. необходимо отметить это в соответствующем объекте в local storage), на прототипе название и автор прочитанной книги отображаются зеленым цветом.

### Примечания

**StackBlitz c каркасом приложения с занятия в субботу 17.04** -- <https://stackblitz.com/edit/toread-list-share?file=js%2Fbooks-ui.js>

**Какими технологиями можно пользоваться?** Принимаются решения как на чистом JS, так и решения использующие современные фреймворки/библиотеки. Можно использовать TypeScript. По части стилей можно использовать препроцессоры, **однако запрещено пользоваться css-библиотеами (например bootstrap’ом) и библиотеками готовых ui-компонент** (кроме случаев, которые будут оговорены на занятии во вторник 20.04).

Использование фреймворков и других технологий будет плюсом, но в то же время соответствие работы требованиям 1-8, качество и оптимальность кода -- это куда более важные факторы. Качественное решение на чистом JS будет оценено безусловно выше, чем решение, использующее фреймворки, но при этом не соответствующее требованиям и/или с низким качеством кода.

**Насколько нужно заморачиваться тем, чтобы приложение приятно выглядело?** На занятии в субботу 17.04 докладчик с целью экономии времени продемонстрировал решение с минимумом стилей. Это совершенно не значит, что при выполнении задания нужно игнорировать внешний вид приложения. Качественный выбор цветовой гаммы, шрифтов, отступов и т.п. будет безусловным плюсом при оценке работы.