Необходимо разработать просмотрщик JSON, который бы выводил контент в виде дерева аналогично тому, как показываются структуры данных в Devtools

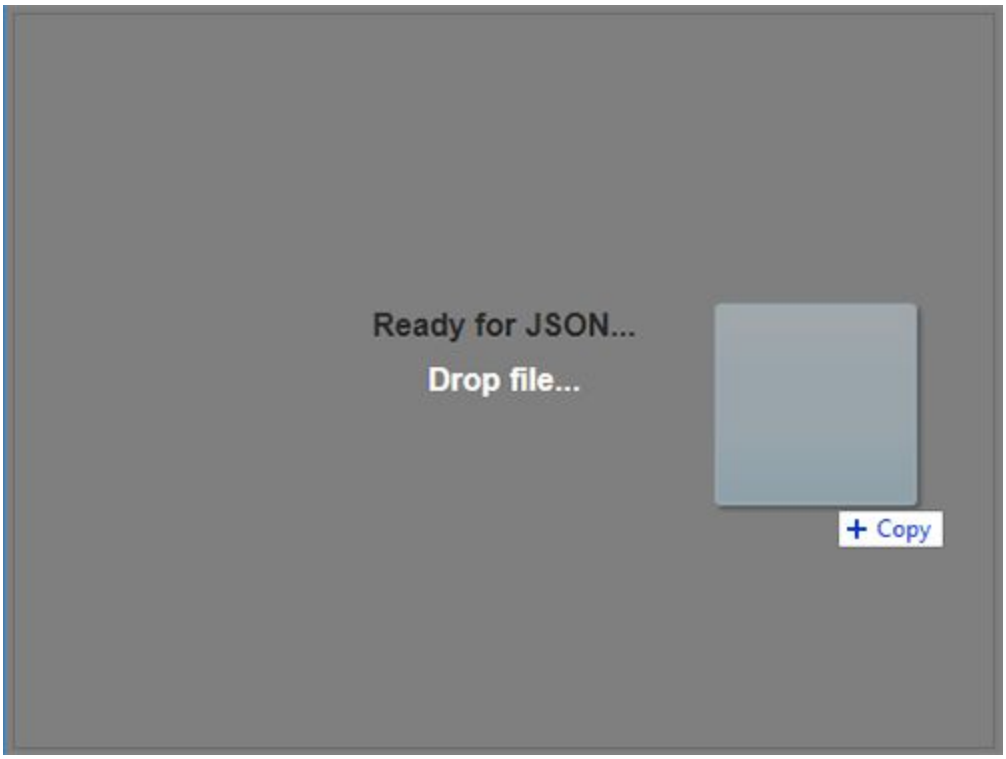
1. Базовая функциональность (5/10 баллов)

Дерево должно выглядеть следующим образом

Узлы-коллекции (array|object) могут отображаться в двух состояниях - свернутом и развернутом (корневой узел развернут всегда). Состояния переключаются по клику на треугольной стрелке слева от узла. При разворачивании раскрывается только сам узел. При сворачивании узла также сворачиваются все его развернутые подузлы.
В свернутом состоянии выводится ограниченный дайджест коллекции (не более трех элементов, далее - **...**), также перед элементами массива не указываются индексы, поскольку в этом случае они очевидны.
В развернутом состоянии индексы массивов показываются всегда, аналогично ключам объектов. Для развернутых узлов их дайджесты не выводятся, вместо них обозначения {**...**} и [**...**].
Счетчик выводится перед каждой коллекцией размером больше 1 элемента.
Атомарные узлы подсвечиваются разными цветами в зависимости от типа. Строки также обрамляются кавычками. Ключи объектов выводятся без кавычек, курсивом.
Поскольку строки могут быть очень длинными, то при их отображении есть несколько ограничений:

- в дайджесте выводится максимум 10 символов, если строка длиннее, то к ним добавляется **…**, чтобы намекнуть пользователю на большее
- размер блока для вывода строки не может превышать 400px * 200px, если контент не помещается, используется скроллинг

Просмотрщик должен быть написан таким образом, чтобы работать в качестве независимого компонента в разных сценариях с разными источниками данных. Однако для целей задания json-контент будет загружаться через перетаскивание соответствующего файла на вкладку приложения



Приложение без данных показывает **Ready for JSON…** или аналогичную жизнеутверждающую надпись на белом фоне, надпись **Drop file…** с затемнением должна появляться, если приложение обнаружит, что пользователь тащит ему свежий файл.
При перетаскивании нового файла на приложение с уже загруженным контентом его состояние сбрасывается, чтобы продемонстрировать новое дерево.
В случае, если файл невозможно прочитать или в нем есть проблемные места, должно выводиться сообщение, где, помимо ошибки, отображается соответствующий фрагмент файла с подсветкой проблемного символа (если такой есть):

2. Фильтрация (+3/10 баллов)

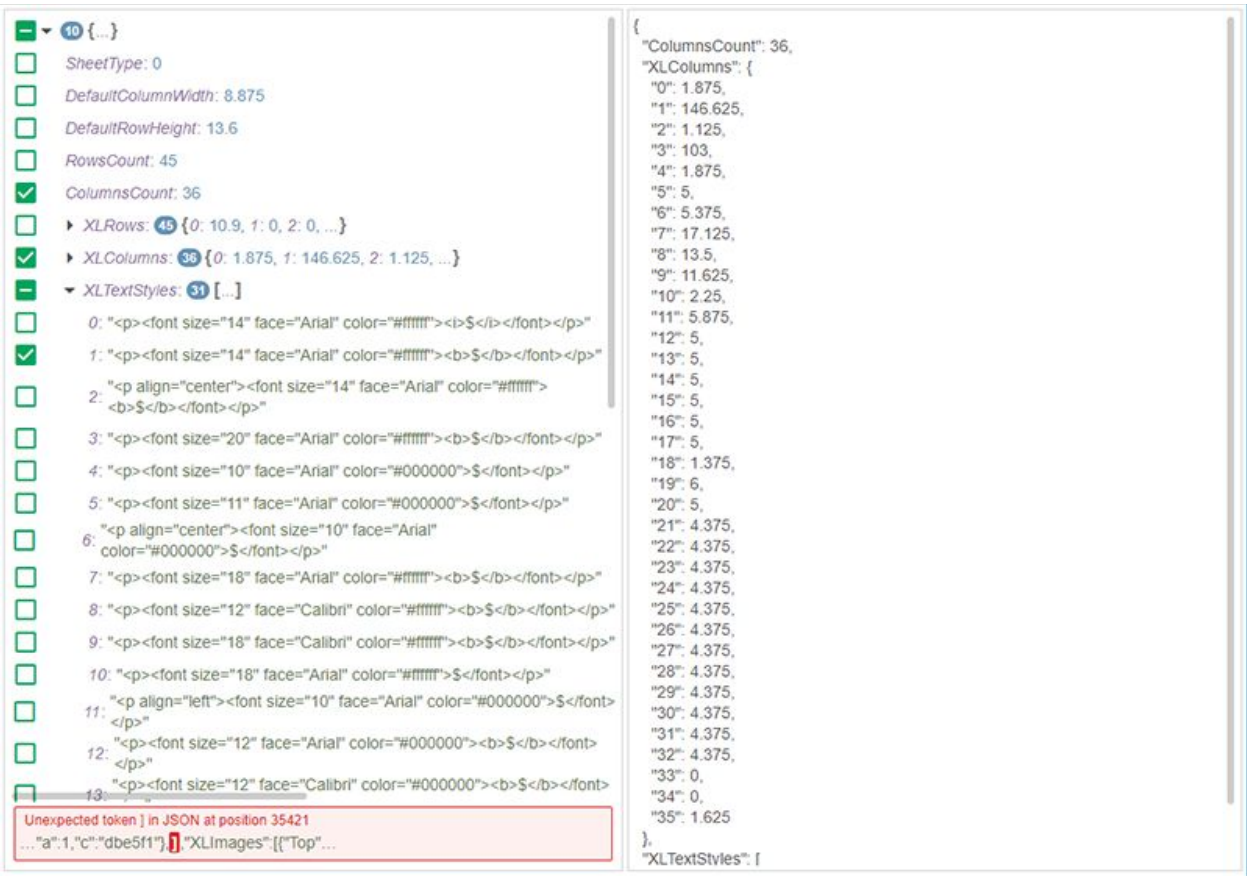
Реализовать возможность выделения узлов дерева и автоматического построения производного JSON, содержащего только выделенные узлы

Как можно увидеть на иллюстрации, узел-коллекция может быть выделен полностью (если выделены полностью все его подузлы) либо частично (если некоторые подузлы не выделены или выделены частично). Клик по частично выделенному узлу снимает выделение со всех его подузлов, их подузлов и так далее. Клик по узлу-коллекции без выделения выделяет все его подузлы, их подузлы и так далее.
При генерации нового JSON-файла в него не включаются пустые узлы, не содержащие хотя бы одного одного атомарного элемента (например, как узел *h* на иллюстрации), даже если они выделены пользователем.
Отступ для каждого следующего уровня сгенерированного JSON - два пробела.
Пользователь должен иметь возможность выделить и скопировать его.

3. Отказоустойчивость (+2/10 баллов)

Реализовать возможность частичного чтения файла в случае ошибки. Пользователь должен иметь возможность видеть те узлы, которые можно прочитать до проблемной позиции. Сообщение об ошибке должно выводиться под деревом.

Полностью выполненное задание будет выглядеть так



Дополнительные условия и примечания:

1. Задание должно быть выполнено на чистом React, без использования библиотек управления состоянием (Redux, MobX, RxJS и т.п.) и готовых компонентов. К счастью, они тут не особо нужны.
2. Можно и нужно использовать create-react-app, react-custom-scrollbars, react-dropzone, clarinet или их аналоги
3. Можно использовать любой метод задания стилей
4. Кое-какие константы для стилей:

6 {...}

▼ font: 3 {...}

▶ basic: 3 {fontFamily: "Roboto, sa...", fontSize: 14, lineHeight: "18px"}

▶ bold: 4 {fontFamily: "Roboto, sa...", fontSize: 16, lineHeight: "18px", ...}

▶ smallBold: 4 {fontFamily: "Roboto, sa...", fontSize: 11, lineHeight: "8px", ...}

rowHeight: 28

levelIndent: 20

syntaxPad: 4

▼ textBlock: 2 {...}

width: 400

maxHeight: 200

▼ color: 7 {...}

basic: "#505050"

▼ counter: 2 {...}

back: "#6591b4"

content: "white"

search: "#00a55c"

expansion: "#808080"

selection: "#00a55c"

▼ json: 6 {...}

syntax: "#808080"

key: "#8c6e9c"

string: "#5c7450"

boolean: "#cc7832"

null: "#cc7832"

number: "#6591b4"

▼ error: 3 {...}

back: "#fff0f0"

message: "#dd2222"

excerpt: "#5c7450"