МойОфис

Комплект Средств Разработки (SDK)

Руководство программиста

АВТОНОМНЫЙ МОДУЛЬ РЕДАКТИРОВАНИЯ

2.3

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)»

АВТОНОМНЫЙ МОДУЛЬ РЕДАКТИРОВАНИЯ

РУКОВОДСТВО ПРОГРАММИСТА

2.3

На 18 листах

Москва

2023

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем.

Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе используются следующие сокращения (см. Таблицу 1):

Таблица 1 - Сокращения и расшифровки

Сокращение	Расшифровка
АМР	Автономный Модуль Редактирования.
ОС	Операционная система.
ПО	Программное обеспечение.
ПО МойОфис	Программное обеспечение «МойОфис Комплект Средств Разработки (SDK). Автономный Модуль Редактирования».
API	Application Programming Interface (программный интерфейс приложения).
SDK	Software Development Kit (комплект для разработки программного обеспечения).

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение

Автономный Модуль Редактирования предназначен для встраивания в прикладные системы сторонних производителей в качестве компонента для просмотра и редактирования текстовых или табличных документов. В состав АМР входит специальная версия веб-приложений редакторов текста и таблиц МойОфис, предназначенная для исполнения в среде веб-браузера в монопольном режиме.

1.2 Возможности

Автономный Модуль Редактирования используется для встраивания в веб-приложения сторонних производителей, в качестве компонента для просмотра и редактирования текстовых или табличных документов.

Пользователю АМР доступны следующие возможности:

1. Обработка электронных текстовых и табличных документах в следующих форматах:
   - чтение и запись текстовых и табличных документов в формате OOXML, расширения файлов DOCX, XLSX, DOTX, XLTX;
   - чтение и запись текстовых и табличных документов в формате ODF, расширения файлов ODT, ODS, OTT, OTS;
   - чтение и запись текстовых и табличных документов в формате ODF, расширения файлов XODT, XODS, XOTT, XOTS;
   - чтение и запись текстовых документов в формате TXT и табличных документов в формате CSV;
   - чтение файлов в форматах PDF.
2. Редактирование содержимого документов, включая текстовые и табличные данные, диаграммы, изображения и др.
3. Поиск и замена фрагмента текста в документе.
4. Запуск макрокоманд.

1.3 Уровень подготовки пользователя

Пользователем ПО МойОфис является веб-разработчик, интегрирующий компоненты просмотра или редактирования в свое приложение.

Требования к квалификации пользователя ПО МойОфис:

1. Уверенное знание современных технологий разработки Single Page Applications: Javascript, Typescript, HTML, CSS.
2. Знание API межоконного взаимодействия и их технических особенностей.
3. Знание систем управления пакетами javascript (yarn, npm).
4. Знание систем сборки веб-приложений (webpack).
5. Навыки настройки веб-сервера.

1.4 Системные требования

Использование ПО МойОфис возможно в браузерах: Chrome, Яндекс.Браузер, Mozilla FireFox, Microsoft Edge (Chromium). Версии браузеров для используемой ОС приведены в документе «МойОфис Комплект Средств Разработки (SDK). Автономный Модуль Редактирования. Системные требования».

Для интеграции ПО МойОфис необходим установленный веб-сервер.

Для запуска демо-приложения необходимо установленное ПО Docker.

Полный перечень требований к программному и аппаратному обеспечению приведен в документе «МойОфис Комплект Средств Разработки (SDK). Автономный Модуль Редактирования. Системные требования».

2 УСТАНОВКА

2.1 Дистрибутив

Дистрибутив ПО МойОфис поставляется в виде архивного файла **MyOffice_AMP_<release_name>_<build number>.zip**, где **<release_name>** – название релиза, а **<build number>** – номер сборки программы.

Архивный файл содержит следующие данные:

- правовые уведомления;
- папка **wte/dist** с ресурсами для развертывания;
- папка **amr-api/dist** с JavaScript файлом amrApi.js – скрипт для взаимодействия внешнего приложения с SDK;
- папка **demo** с демонстрационным приложением.

2.2 Запуск демо приложения в контейнере docker

Для интеграции ПО МойОфис выполните следующие действия:

1. Создайте каталог установки, например, папку **AMP**.
2. Извлеките содержимое архивного файла дистрибутива (см. раздел 2.1) в каталог установки.
3. Перейдите в папку **demo** каталога установки и в командной строке последовательно запустите следующие команды:

```
yarn install
yarn build
docker build -t wte-demo .
docker run -it -p 8000:80 wte-demo
```

2.3 Особенности использования

Текущие ограничения использования:

1. После сбоя документ автоматически не сохраняется и редактор не восстанавливается.
2. Бинарные форматы документов Microsoft (`doc`, `xls`, etc) не поддерживаются.
3. Для повторного открытия документа необходима перезагрузка `iframe`.

Для правильной и быстрой работы приложения нужно реализовать следующее:

1. Правильно настроить заголовки для типа файлов WebAssebly (`application/wasm` для `wasm` файлов).
2. Подключить правильные заголовки кеширования на уровне HTTP (https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Cache-Control, https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching).
3. При перемещении внутри таблиц возможно выполнение встроенных браузерных жестов («назад» и «вперед»). Чтобы избежать срабатывания браузерных жестов навигации, необходимо в элементах `<html/>` и `<body/>` страницы интегратора добавить стиль `overscroll-behavior-x: none`.
4. Необходимо предоставить доступ iframe к Clipboard API: `<iframe src="editor_url" allow="clipboard-read; clipboard-write"></iframe>`.
5. Необходимо дождаться события 'load' у `iframe`, в котором мы запускаем приложение.
6. Подключить статическое сжатие контента (рекомендуется `Brotli`) https://www.smashingmagazine.com/2021/01/front-end-performance-assets-optimizations/#assets-optimizations). Запрос клиента должен содержать в заголовке `accept-encoding:gzip, br`, сервер должен поддерживать сжатие.

2.4 Проверка работоспособности с использованием демо приложениия

Для проверки работоспособности ПО МойОфис выполните следующие действия:

1. Перейдите в папку **demo** каталога установки ПО МойОфис и в командной строке последовательно запустите следующие команды:

```
yarn install
yarn start:dist
```

2. Запустите браузер из числа поддерживаемых, например, Mozilla FireFox.

3. В браузере перейдите по адресу:

```
http://localhost:8000
```

4. На экране откроется окно демонстрационного примера использования ПО МойОфис.

ПО МойОфис считается работоспособным, если:

1. Содержимое окна демонстрационного примера использования ПО МойОфис на экране аналогично приведенному на Рисунке 1.

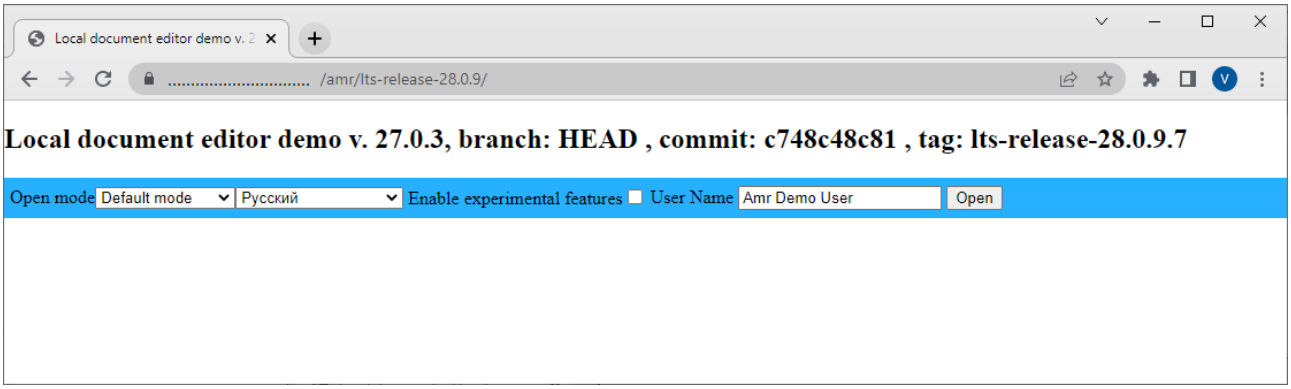

Рисунок 1 – Окно демонстрационного примера использования ПО МойОфис

2. При нажатии кнопки **Open** (см. Рисунок 1) и после подтверждения выбора текстового документа, произошла успешная загрузка документа в окно редактирования (например, см. Рисунок 2).

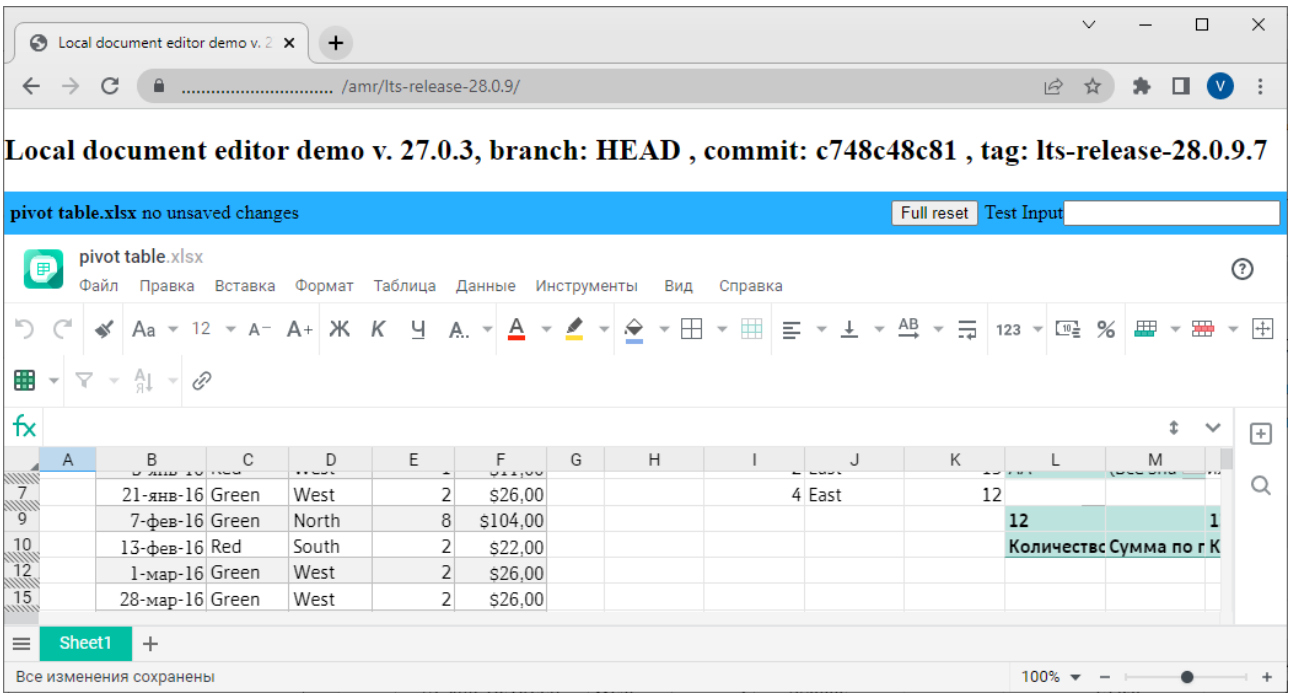

Рисунок 2 – Пример окна редактирования табличного документа

3 API МЕТОДЫ И УВЕДОМЛЕНИЯ

3.1 API методы

3.1.1 Метод openDocument

Метод `openDocument` предназначен для открытия документа в редакторе.

```
openDocument = ({
    content: ArrayBuffer,
    filename: string,
    lang: string,
    mode: 'review' | 'readonly' | 'default',
    userName: string,
    usePerfHelper?: boolean,
    workerInitTimeout?: number,
    coreInitTimeout?: number,
}) => null | ErrorDescription
```

Параметры:

– `content` – содержимое документа (массив байт данных файла);

– `filename` – название документа, отображается в заголовке редактора;

– `lang` – язык интерфейса редактора;

- `'ru-RU'` – русский;
- `'en-US'` – английский;
- `'es-PA'` – испанский;
- `'fr-FR'` – французский;
- `'tt-RU'` – татарский;
- `'ba-RU'` – башкирский;
- `'pt-BR'` – португальский;
- `'de-DE'` – немецкий;
- `'it-IT'` – итальянский;
- `'be-BY'` – белорусский;
- `'kk-KZ'` – казахский;
- `'ky-KG'` – киргизский;
- `'hy-AM'` – армянский;

– `mode` – параметр, позволяющий выбрать один из следующих режимов редактора:

```
enum Modes {
    // только чтение, без разрешения на редактирование документа
    readonly = "readonly",
    // редактирование документа в режиме рецензирования,
    // без возможности отключить отслеживание изменений
    review = "review",
    // редактирование документа с полными правами
    default = "default"
}
```

- `userName` – имя пользователя (строка, значение по умолчанию - 'Local User'), которое будет отображено в комментариях и отслеживаемых изменениях;
- `usePerfHelper` – флаг для включения инструмента PerformanceHelper, использующегося для измерения метрик; необязательный параметр;
- `workerInitTimeout` – таймаут загрузки потока ядра, необязательный параметр;
- `coreInitTimeout` – таймаут инициализации ядра, необязательный параметр.

Метод возвращает `null` если инициализация была успешно выполнена. В случае ошибки метод возвращает структуру, содержащую сообщение об ошибке:

```
type ErrorDescription = {
    errorInfo: string; // сообщение об ошибке
}
```

3.1.2 Метод saveDocument

Метод `saveDocument` предназначен для сохранения редактируемого документа в выбранном формате.

После сохранения документ не закрывается и не блокируется, поэтому пользователь может продолжать работу с документом.

```
saveDocument = (format: string) => Uint8Array
```

Возвращаемый тип: `Uint8Array`.

Метод возвращает содержимое сохраненного документа. После сохранения документ не будет заблокирован или закрыт. Пользователь сможет продолжить редактирование документа.

Параметр:

`format` – перечисление, описывающее поддерживаемый формат сохранения документа:

- `OXML` – Microsoft Office Open XML document format (DOCX, XLSX, DOTX, XLTS);
- `ODF` – Open Document Format (ODT, ODS, OTT, OTS, а также XODT, XODS, XOTT, XOTS);
- `PDF` – Portable Document Format (PDF);
- `PDF/A` – Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

3.1.3 Метод blur

Метод `blur` отключает сохранение фокуса в редакторе.

```
blur = () => Promise<void>
```

Метод возвращает объект `Promise`, при этом нет необходимости дожидаться окончания выполнения.

3.2 API уведомления

3.2.1 Уведомление onChange

```
onChange ({hasChanges: boolean})
```

Уведомляет о наличии изменений в документе. Например, пользователь может отредактировать документ (флаг будет установлен в `true`), затем отменить изменения (флаг будет установлен в `false`), затем внести другие изменения (флаг будет установлен в `true`). После вызова метода `saveDocument` флагу будет присвоено значение `false`.

После вызова метода `saveDocument` флагу будет присвоено значение `false`.

3.2.2 Уведомление onError

```
onError (errorDescription: string)
```

Оповещает о фатальной ошибке при открытии, редактировании или сохранении документа.

В случае возникновения ошибки работа редактора останавливается и на экране отображается диалоговое окно с подробной информацией об ошибке и с кнопкой запроса

перезагрузки. Нажатие кнопки перезагрузки отслеживается с помощью уведомления `onPageReloadRequested` и требует корректной обработки.

В настоящее время отсутствуют автосохранение или автоматическое восстановление. Все несохраненные изменения будут потеряны.

Работа с защищенным паролем документом в настоящее время не поддерживается. При открытии такого документа хост-приложение получит уведомление об ошибке, содержащей строку “**PASSWORD_PROTECTED_DOCUMENT_ERROR**”.

3.2.3 Уведомление onPageReloadRequested

```
onPageReloadRequested()
```

Уведомляет о нажатии пользователем кнопки перезагрузки в окне ошибки при остановке редактора.

4 ПРИМЕР ИСПОЛЬЗОВАНИЯ

Пример использования API методов и уведомлений при разработке приложений.

```
// создание iframe
<iframe id="editor" src={http://some.site.name}>

// инициализация API, подключение скрипта для работы с SDK
import AmrApi from 'AmrApi.js';
const origin = window.location.origin;

const editorIframe = document.getElementById('editor');
const editorApi = new AmrApi(editorIframe, origin);

function onHasChanges(hasChanges) {
    this.hasChanges = hasChanges;
    if (hasChanges) {
        console.log(DOCUMENT_STATE.changed);
    } else {
        console.log(DOCUMENT_STATE.saved);
    }
}
function onError(errorMessage) {
    console.log('Ошибка открытия документа', err);
}
// отслеживание изменений в документе
editorApi.onChange(({ hasChanges }) => {
    onHasChanges(hasChanges);
});
// обработка ошибки
editorApi.onError(onError);
editorApi.onPageReloadRequested(() => window.location.reload());

// открытие документа
const documentData = new UintArray(...);
const docInfo = await editorApi.openDocument({
    content: documentData,
    filename: 'filename',
```

```
            lang: 'ru',
            mode: 'default',
            userName: 'Alex',
        });

        window.console.log('opened', docInfo);
```

По окончании редактирования пользователь может получить измененное содержимое документа:

```
        const documentData = editorAPI.saveDocument('OXML')
        console.log('Document saved', documentData);
```

Если пользователь не сохранил изменения, то можно предотвратить потерю изменений в документе с помощью предупреждения. Для этого можно использовать событие `beforeunload` и проверку флага `hasChanges`. Значение флага `hasChanges` можно получить из обработчика `onChange`. Реализуйте обработчик `onBeforeUnload` в `componenDidUpdate`.

Пример установки состояния флага `hasChanges` и добавления/удаления обработчика для события `beforeunload`:

```
        if (hasChanges) {
          window.addEventListener("beforeunload", onBeforeUnload, {
            capture: true,
          });
        } else {
          window.removeEventListener("beforeunload", onBeforeUnload, {
            capture: true,
          });
        }

        function onBeforeUnload(event) {
            event.preventDefault();
            return (event.returnValue = 'You have unsaved changes. Are you sure
you want to leave this page?');
        }
```


После проверки значения `hasChanges`, запрашиваем пользователя о несохраненных изменениях.

```
      reload() {
          if (this.hasChanges) {
            const result = confirm('You have unsaved changes. Are you sure you
want to leave this page?');
            if (result) {
              window.removeEventListener("beforeunload", this.onBeforeUnload, {
                capture: true,
              });
            } else {
              return;
            }
          }
          // reload iframe
      }
```