/\*!

 \* Copyright (c) Ascensio System SIA 2024. All rights reserved

 \*

 \* http://www.onlyoffice.com

 \*

 \* Version: 8.0.1 (build:31)

 \*/

;(function(DocsAPI, window, document, undefined) {

 /\*

 # Full #

 config = {

 type: 'desktop or mobile or embedded',

 width: '100% by default',

 height: '100% by default',

 documentType: 'word' | 'cell' | 'slide' | 'pdf' ,// deprecate 'text' | 'spreadsheet' | 'presentation',

 token: <string> encrypted signature

 document: {

 title: 'document title',

 url: 'document url'

 fileType: 'document file type',

 options: <advanced options>,

 key: 'key',

 vkey: 'vkey',

 referenceData: 'data for external paste',

 info: {

 owner: 'owner name',

 folder: 'path to document',

 uploaded: '<uploaded date>',

 sharingSettings: [

 {

 user: 'user name',

 permissions: '<permissions>',

 isLink: false

 },

 ...

 ],

 favorite: '<file is favorite>' // true/false/undefined (undefined - don't show fav. button)

 },

 permissions: {

 edit: <can edit>, // default = true

 download: <can download>, // default = true

 reader: <can view in readable mode>,

 review: <can review>, // default = edit

 print: <can print>, // default = true

 comment: <can comment in view mode> // default = edit,

 modifyFilter: <can add, remove and save filter in the spreadsheet> // default = true

 modifyContentControl: <can modify content controls in documenteditor> // default = true

 fillForms: <can edit forms in view mode> // default = edit || review,

 copy: <can copy data> // default = true,

 editCommentAuthorOnly: <can edit your own comments only> // default = false

 deleteCommentAuthorOnly: <can delete your own comments only> // default = false,

 reviewGroups: ["Group1", ""] // current user can accept/reject review changes made by users from Group1 and users without a group. [] - use groups, but can't change any group's changes

 commentGroups: { // {} - use groups, but can't view/edit/delete any group's comments

 view: ["Group1", ""] // current user can view comments made by users from Group1 and users without a group.

 edit: ["Group1", ""] // current user can edit comments made by users from Group1 and users without a group.

 remove: ["Group1", ""] // current user can remove comments made by users from Group1 and users without a group.

 },

 userInfoGroups: ["Group1", ""], // show tooltips/cursors/info in header only for users in userInfoGroups groups. [""] - means users without group, [] - don't show any users, null/undefined/"" - show all users

 protect: <can protect document> // default = true. show/hide protect tab or protect buttons

 }

 },

 editorConfig: {

 actionLink: { // open file and scroll to data, used with onMakeActionLink or the onRequestSendNotify event

 action: {

 type: "bookmark", // or type="comment"

 data: <bookmark name> // or comment id

 }

 },

 mode: 'view or edit',

 lang: <language code>,

 location: <location>,

 canCoAuthoring: <can coauthoring documents>,

 canBackToFolder: <can return to folder> - deprecated. use "customization.goback" parameter,

 createUrl: 'create document url',

 sharingSettingsUrl: 'document sharing settings url',

 fileChoiceUrl: 'source url', // for mail merge or image from storage

 callbackUrl: <url for connection between sdk and portal>,

 mergeFolderUrl: 'folder for saving merged file', // must be deprecated, use saveAsUrl instead

 saveAsUrl: 'folder for saving files'

 licenseUrl: <url for license>,

 customerId: <customer id>,

 region: <regional settings> // can be 'en-us' or lang code

 user: {

 id: 'user id',

 name: 'user name',

 group: 'group name' // for customization.reviewPermissions or permissions.reviewGroups or permissions.commentGroups. Can be multiple groups separated by commas (,) : 'Group1' or 'Group1,Group2'

 image: 'image url'

 },

 recent: [

 {

 title: 'document title',

 url: 'document url',

 folder: 'path to document',

 },

 ...

 ],

 templates: [

 {

 title: 'template name', // name - is deprecated

 image: 'template icon url',

 url: 'http://...'

 },

 ...

 ],

 customization: {

 logo: {

 image: url,

 imageDark: url, // logo for dark theme

 imageEmbedded: url, // deprecated, use image instead

 url: http://...

 },

 customer: {

 name: 'SuperPuper',

 address: 'New-York, 125f-25',

 mail: 'support@gmail.com',

 www: 'www.superpuper.com',

 phone: '1234567890',

 info: 'Some info',

 logo: '',

 logoDark: '', // logo for dark theme

 },

 about: true,

 feedback: {

 visible: false,

 url: http://...

 },

 goback: {

 url: 'http://...',

 text: 'Go to London',

 blank: true,

 requestClose: false // if true - goback send onRequestClose event instead opening url

 },

 reviewPermissions: {

 "Group1": ["Group2"], // users from Group1 can accept/reject review changes made by users from Group2

 "Group2": ["Group1", "Group2"] // users from Group2 can accept/reject review changes made by users from Group1 and Group2

 "Group3": [""] // users from Group3 can accept/reject review changes made by users without a group

 },

 anonymous: { // set name for anonymous user

 request: bool (default: true), // enable set name

 label: string (default: "Guest") // postfix for user name

 },

 review: {

 hideReviewDisplay: false, // hide button Review mode

 hoverMode: false, // true - show review balloons on mouse move, not on click on text

 showReviewChanges: false,

 reviewDisplay: 'original', // original for viewer, markup for editor

 trackChanges: undefined // true/false - open editor with track changes mode on/off,

 },

 layout: { // hide elements, but don't disable feature

 toolbar: {

 file: { // menu file

 close: false / true, // close menu button

 settings: false / true, // advanced settings

 info: false / true // document info

 save: false/true // save button

 } / false / true,

 home: {

 mailmerge: false/true // mail merge button

 },

 layout: false / true, // layout tab

 references: false / true, // de references tab

 collaboration: false / true // collaboration tab

 draw: false / true // draw tab

 protect: false / true, // protect tab

 plugins: false / true // plugins tab

 view: {

 navigation: false/true // navigation button in de

 } / false / true, // view tab

 save: false/true // save button on toolbar in

 } / false / true, // use instead of customization.toolbar,

 header: {

 users: false/true // users list button

 save: false/true // save button

 },

 leftMenu: {

 navigation: false/true,

 spellcheck: false/true // spellcheck button in sse,

 mode: false/true // init value for left panel, true - is visible, false - is hidden, used for option "Left panel" on the View Tab

 } / false / true, // use instead of customization.leftMenu

 rightMenu: {

 mode: false/true // init value for right panel, true - is visible, false - is hidden, used for option "Right panel" on the View Tab

 } / false/true, // use instead of customization.rightMenu

 statusBar: {

 textLang: false/true // text language button in de/pe

 docLang: false/true // document language button in de/pe

 actionStatus: false/true // status of operation

 } / false / true, // use instead of customization.statusBar

 },

 features: { // disable feature

 spellcheck: {

 mode: false/true // init value in de/pe

 change: false/true // hide/show feature in de/pe/sse

 } / false / true // if false/true - use as init value in de/pe. use instead of customization.spellcheck parameter

 },

 font: {

 name: "Arial",

 size: "11px";

 },

 chat: true,

 comments: true,

 zoom: 100,

 compactToolbar: false,

 leftMenu: true, // must be deprecated. use layout.leftMenu instead

 rightMenu: true, // must be deprecated. use layout.rightMenu instead

 hideRightMenu: false, // hide or show right panel on first loading

 toolbar: true, // must be deprecated. use layout.toolbar instead

 statusBar: true, // must be deprecated. use layout.statusBar instead

 autosave: true,

 forcesave: false,

 commentAuthorOnly: false, // must be deprecated. use permissions.editCommentAuthorOnly and permissions.deleteCommentAuthorOnly instead

 showReviewChanges: false, // must be deprecated. use customization.review.showReviewChanges instead

 help: true,

 compactHeader: false,

 toolbarNoTabs: false,

 toolbarHideFileName: false,

 reviewDisplay: 'original', // must be deprecated. use customization.review.reviewDisplay instead

 spellcheck: true, // must be deprecated. use customization.features.spellcheck instead

 compatibleFeatures: false,

 unit: 'cm' // cm, pt, inch,

 mentionShare : true // customize tooltip for mention,

 macros: true // can run macros in document

 plugins: true // can run plugins in document

 macrosMode: 'warn' // warn about automatic macros, 'enable', 'disable', 'warn',

 trackChanges: undefined // true/false - open editor with track changes mode on/off, // must be deprecated. use customization.review.trackChanges instead

 hideRulers: false // hide or show rulers on first loading (presentation or document editor)

 hideNotes: false // hide or show notes panel on first loading (presentation editor)

 uiTheme: 'theme-dark' // set interface theme: id or default-dark/default-light

 integrationMode: "embed" // turn off scroll to frame

 mobileForceView: true/false (default: true) // turn on/off the 'reader' mode on launch. for mobile document editor only

 },

 coEditing: {

 mode: 'fast', // <coauthoring mode>, 'fast' or 'strict'. if 'fast' and 'customization.autosave'=false -> set 'customization.autosave'=true. 'fast' - default for editor

 // for viewer: 'strict' is default, offline viewer; 'fast' - live viewer, show changes from other users

 change: true, // can change co-authoring mode. true - default for editor, false - default for viewer

 },

 plugins: {

 autostart: ['asc.{FFE1F462-1EA2-4391-990D-4CC84940B754}'],

 pluginsData: [

 "helloworld/config.json",

 "chess/config.json",

 "speech/config.json",

 "clipart/config.json",

 ]

 },

 wopi: { // only for wopi

 FileNameMaxLength: 250 // max filename length for rename, 250 by default

 }

 },

 events: {

 'onAppReady': <application ready callback>,

 'onDocumentStateChange': <document state changed callback>

 'onDocumentReady': <document ready callback>

 'onRequestEditRights': <request rights for switching from view to edit>,

 'onRequestHistory': <request version history>,// must call refreshHistory method

 'onRequestHistoryData': <request version data>,// must call setHistoryData method

 'onRequestRestore': <try to restore selected version>,

 'onRequestHistoryClose': <request closing history>,

 'onError': <error callback>,

 'onWarning': <warning callback>,

 'onInfo': <document open callback>,// send view or edit mode

 'onOutdatedVersion': <outdated version callback>,// send when previous version is opened

 'onDownloadAs': <download as callback>,// send url of downloaded file as a response for downloadAs method

 'onRequestSaveAs': <try to save copy of the document>,

 'onCollaborativeChanges': <co-editing changes callback>,// send when other user co-edit document

 'onRequestRename': <try to rename document>,

 'onMetaChange': // send when meta information changed

 'onRequestClose': <request close editor>,

 'onMakeActionLink': <request link to document with bookmark, comment...>,// must call setActionLink method

 'onRequestUsers': <request users list for mentions>,// must call setUsers method

 'onRequestSendNotify': //send when user is mentioned in a comment,

 'onRequestInsertImage': <try to insert image>,// must call insertImage method

 'onRequestCompareFile': <request file to compare>,// must call setRevisedFile method. must be deprecated

 'onRequestSharingSettings': <request sharing settings>,// must call setSharingSettings method

 'onRequestCreateNew': <try to create document>,

 'onRequestReferenceData': <try to refresh external data>,

 'onRequestOpen': <try to open external link>,

 'onRequestSelectDocument': <try to open document>, // used for compare and combine documents. must call setRequestedDocument method. use instead of onRequestCompareFile/setRevisedFile

 'onRequestSelectSpreadsheet': <try to open spreadsheet>, // used for mailmerge id de. must call setRequestedSpreadsheet method. use instead of onRequestMailMergeRecipients/setMailMergeRecipients

 'onRequestReferenceSource': <try to change source for external link>, // used for external links in sse. must call setReferenceSource method

 }

 }

 # Embedded #

 config = {

 type: 'embedded',

 width: '100% by default',

 height: '100% by default',

 documentType: 'word' | 'cell' | 'slide',// deprecate 'text' | 'spreadsheet' | 'presentation',

 document: {

 title: 'document title',

 url: 'document url',

 fileType: 'document file type',

 key: 'key',

 vkey: 'vkey'

 },

 editorConfig: {

 licenseUrl: <url for license>,

 customerId: <customer id>,

 autostart: 'document', // action for app's autostart. for presentations default value is 'player'

 embedded: {

 embedUrl: 'url',

 fullscreenUrl: 'url',

 saveUrl: 'url',

 shareUrl: 'url',

 toolbarDocked: 'top or bottom'

 }

 },

 events: {

 'onAppReady': <application ready callback>,

 'onBack': <back to folder callback>,

 'onError': <error callback>,

 'onDocumentReady': <document ready callback>,

 'onWarning': <warning callback>

 }

 }

 \*/

 // TODO: allow several instances on one page simultaneously

 DocsAPI.DocEditor = function(placeholderId, config) {

 var \_self = this,

 \_config = config || {};

 extend(\_config, DocsAPI.DocEditor.defaultConfig);

 \_config.editorConfig.canUseHistory = \_config.events && !!\_config.events.onRequestHistory;

 \_config.editorConfig.canHistoryClose = \_config.events && !!\_config.events.onRequestHistoryClose;

 \_config.editorConfig.canHistoryRestore = \_config.events && !!\_config.events.onRequestRestore;

 \_config.editorConfig.canSendEmailAddresses = \_config.events && !!\_config.events.onRequestEmailAddresses;

 \_config.editorConfig.canRequestEditRights = \_config.events && !!\_config.events.onRequestEditRights;

 \_config.editorConfig.canRequestClose = \_config.events && !!\_config.events.onRequestClose;

 \_config.editorConfig.canRename = \_config.events && !!\_config.events.onRequestRename;

 \_config.editorConfig.canMakeActionLink = \_config.events && !!\_config.events.onMakeActionLink;

 \_config.editorConfig.canRequestUsers = \_config.events && !!\_config.events.onRequestUsers;

 \_config.editorConfig.canRequestSendNotify = \_config.events && !!\_config.events.onRequestSendNotify;

 \_config.editorConfig.mergeFolderUrl = \_config.editorConfig.mergeFolderUrl || \_config.editorConfig.saveAsUrl;

 \_config.editorConfig.canRequestSaveAs = \_config.events && !!\_config.events.onRequestSaveAs;

 \_config.editorConfig.canRequestInsertImage = \_config.events && !!\_config.events.onRequestInsertImage;

 \_config.editorConfig.canRequestMailMergeRecipients = \_config.events && !!\_config.events.onRequestMailMergeRecipients;

 \_config.editorConfig.canRequestCompareFile = \_config.events && !!\_config.events.onRequestCompareFile;

 \_config.editorConfig.canRequestSharingSettings = \_config.events && !!\_config.events.onRequestSharingSettings;

 \_config.editorConfig.canRequestCreateNew = \_config.events && !!\_config.events.onRequestCreateNew;

 \_config.editorConfig.canRequestReferenceData = \_config.events && !!\_config.events.onRequestReferenceData;

 \_config.editorConfig.canRequestOpen = \_config.events && !!\_config.events.onRequestOpen;

 \_config.editorConfig.canRequestSelectDocument = \_config.events && !!\_config.events.onRequestSelectDocument;

 \_config.editorConfig.canRequestSelectSpreadsheet = \_config.events && !!\_config.events.onRequestSelectSpreadsheet;

 \_config.editorConfig.canRequestReferenceSource = \_config.events && !!\_config.events.onRequestReferenceSource;

 \_config.frameEditorId = placeholderId;

 \_config.parentOrigin = window.location.origin;

 (function(){function b(a){this.frame=a;this.x=window.scrollX;this.y=window.scrollY;this.lockCounter=0;document.addEventListener("scroll",this.onScroll.bind(this),!1);window.addEventListener("blur",this.onBlur.bind(this),!1);window.addEventListener("pointermove",this.onMove.bind(this),!1);window.addEventListener("wheel",this.onMove.bind(this),!1);this.frame.addEventListener("pointerover",this.onOver.bind(this),!1);this.frame.addEventListener("pointerleave",this.onLeave.bind(this),!1)}window.AscEmbed=

window.AscEmbed||{};b.prototype.onScroll=function(){document.activeElement===this.frame||0!==this.lockCounter?window.scrollTo(this.x,this.y):(this.x=window.scrollX,this.y=window.scrollY)};b.prototype.onBlur=function(){document.activeElement===this.frame&&this.lockWithTimeout(500)};b.prototype.onOver=function(){};b.prototype.onLeave=function(){this.lockWithTimeout(100);this.frame.blur()};b.prototype.onMove=function(){document.activeElement===this.frame&&(this.lockWithTimeout(100),this.frame.blur())};

b.prototype.lockWithTimeout=function(a){this.lockCounter++;var c=this;setTimeout(function(){c.lockCounter--},a)};window.AscEmbed.initWorker=function(a){window.AscEmbed.workers=window.AscEmbed.workers||[];a=new b(a);window.AscEmbed.workers.push(a);return a}})();

 var onMouseUp = function (evt) {

 \_processMouse(evt);

 };

 var \_attachMouseEvents = function() {

 if (window.addEventListener) {

 window.addEventListener("mouseup", onMouseUp, false)

 } else if (window.attachEvent) {

 window.attachEvent("onmouseup", onMouseUp);

 }

 };

 var \_detachMouseEvents = function() {

 if (window.removeEventListener) {

 window.removeEventListener("mouseup", onMouseUp, false)

 } else if (window.detachEvent) {

 window.detachEvent("onmouseup", onMouseUp);

 }

 };

 var \_onAppReady = function() {

 if (\_config.type === 'mobile') {

 document.body.onfocus = function(e) {

 setTimeout(function(){

 iframe.contentWindow.focus();

 \_sendCommand({

 command: 'resetFocus',

 data: {}

 })

 }, 10);

 };

 }

 \_attachMouseEvents();

 if (\_config.editorConfig) {

 \_init(\_config.editorConfig);

 }

 if (\_config.document) {

 \_openDocument(\_config.document);

 }

 };

 var \_onMessage = function(msg) {

 if ( msg ) {

 if ( msg.type === "onExternalPluginMessage" ) {

 \_sendCommand(msg);

 } else if ((window.parent !== window) && msg.type === "onExternalPluginMessageCallback") {

 postMessage(window.parent, msg);

 } else

 if ( msg.frameEditorId == placeholderId ) {

 var events = \_config.events || {},

 handler = events[msg.event],

 res;

 if (msg.event === 'onRequestEditRights' && !handler) {

 \_applyEditRights(false, 'handler isn\'t defined');

 } else {

 if (msg.event === 'onAppReady') {

 \_onAppReady();

 }

 if (handler && typeof handler == "function") {

 res = handler.call(\_self, {target: \_self, data: msg.data});

 }

 }

 }

 }

 };

 var \_checkConfigParams = function() {

 if (\_config.document) {

 if (!\_config.document.url || ((typeof \_config.document.fileType !== 'string' || \_config.document.fileType=='') &&

 (typeof \_config.documentType !== 'string' || \_config.documentType==''))) {

 window.alert("One or more required parameter for the config object is not set");

 return false;

 }

 var appMap = {

 'text': 'docx',

 'text-pdf': 'pdf',

 'spreadsheet': 'xlsx',

 'presentation': 'pptx',

 'word': 'docx',

 'cell': 'xlsx',

 'slide': 'pptx',

 'pdf': 'pdf'

 }, app;

 if (\_config.documentType=='text' || \_config.documentType=='spreadsheet' ||\_config.documentType=='presentation')

 console.warn("The \"documentType\" parameter for the config object must take one of the values word/cell/slide/pdf.");

 if (typeof \_config.documentType === 'string' && \_config.documentType != '') {

 app = appMap[\_config.documentType.toLowerCase()];

 if (!app) {

 window.alert("The \"documentType\" parameter for the config object is invalid. Please correct it.");

 return false;

 } else if (typeof \_config.document.fileType !== 'string' || \_config.document.fileType == '') {

 \_config.document.fileType = app;

 }

 }

 if (typeof \_config.document.fileType === 'string' && \_config.document.fileType != '') {

 \_config.document.fileType = \_config.document.fileType.toLowerCase();

 var type = /^(?:(xls|xlsx|ods|csv|gsheet|xlsm|xlt|xltm|xltx|fods|ots|xlsb|sxc|et|ett)|(pps|ppsx|ppt|pptx|odp|gslides|pot|potm|potx|ppsm|pptm|fodp|otp|sxi|dps|dpt)|(pdf|djvu|xps|oxps)|(doc|docx|odt|gdoc|txt|rtf|mht|htm|html|mhtml|epub|docm|dot|dotm|dotx|fodt|ott|fb2|xml|oform|docxf|sxw|stw|wps|wpt))$/

 .exec(\_config.document.fileType);

 if (!type) {

 window.alert("The \"document.fileType\" parameter for the config object is invalid. Please correct it.");

 return false;

 } else if (typeof \_config.documentType !== 'string' || \_config.documentType == ''){

 if (typeof type[1] === 'string') \_config.documentType = 'cell'; else

 if (typeof type[2] === 'string') \_config.documentType = 'slide'; else

 if (typeof type[3] === 'string') \_config.documentType = 'pdf'; else

 if (typeof type[4] === 'string') \_config.documentType = 'word';

 }

 }

 var type = /^(?:(pdf|djvu|xps|oxps))$/.exec(\_config.document.fileType);

 if (type && typeof type[1] === 'string') {

 \_config.editorConfig.canUseHistory = false;

 }

 if (!\_config.document.title || \_config.document.title=='')

 \_config.document.title = 'Unnamed.' + \_config.document.fileType;

 if (!\_config.document.key) {

 \_config.document.key = 'xxxxxxxxxxxxxxxxxxxx'.replace(/[x]/g, function (c) {var r = Math.random() \* 16 | 0; return r.toString(16);});

 } else if (typeof \_config.document.key !== 'string') {

 window.alert("The \"document.key\" parameter for the config object must be string. Please correct it.");

 return false;

 }

 if (\_config.editorConfig.user && \_config.editorConfig.user.id && (typeof \_config.editorConfig.user.id == 'number')) {

 \_config.editorConfig.user.id = \_config.editorConfig.user.id.toString();

 console.warn("The \"id\" parameter for the editorConfig.user object must be a string.");

 }

 \_config.document.token = \_config.token;

 }

 return true;

 };

 (function() {

 var result = /[\?\&]placement=(\w+)&?/.exec(window.location.search);

 if (!!result && result.length) {

 if (result[1] == 'desktop') {

 \_config.editorConfig.targetApp = result[1];

 // \_config.editorConfig.canBackToFolder = false;

 if (!\_config.editorConfig.customization) \_config.editorConfig.customization = {};

 \_config.editorConfig.customization.about = false;

 \_config.editorConfig.customization.compactHeader = false;

 }

 }

 })();

 var target = document.getElementById(placeholderId),

 iframe;

 if (target && \_checkConfigParams()) {

 iframe = createIframe(\_config);

 if (\_config.editorConfig.customization && \_config.editorConfig.customization.integrationMode==='embed')

 window.AscEmbed && window.AscEmbed.initWorker(iframe);

 if (\_config.document && (\_config.document.isForm!==true && \_config.document.isForm!==false)) {

 iframe.onload = function() {

 \_sendCommand({

 command: 'checkParams',

 data: {

 url: \_config.document.url,

 directUrl: \_config.document.directUrl,

 token: \_config.document.token,

 key: \_config.document.key

 }

 })

 };

 }

 if (iframe.src) {

 var pathArray = iframe.src.split('/');

 this.frameOrigin = pathArray[0] + '//' + pathArray[2];

 }

 target.parentNode && target.parentNode.replaceChild(iframe, target);

 var \_msgDispatcher = new MessageDispatcher(\_onMessage, this);

 }

 /\*

 cmd = {

 command: 'commandName',

 data: <command specific data>

 }

 \*/

 var \_destroyEditor = function(cmd) {

 var target = document.createElement("div");

 target.setAttribute('id', placeholderId);

 if (iframe) {

 \_msgDispatcher && \_msgDispatcher.unbindEvents();

 \_detachMouseEvents();

 iframe.parentNode && iframe.parentNode.replaceChild(target, iframe);

 }

 };

 var \_sendCommand = function(cmd) {

 if (iframe && iframe.contentWindow)

 postMessage(iframe.contentWindow, cmd);

 };

 var \_init = function(editorConfig) {

 \_sendCommand({

 command: 'init',

 data: {

 config: editorConfig

 }

 });

 };

 var \_openDocument = function(doc) {

 \_sendCommand({

 command: 'openDocument',

 data: {

 doc: doc

 }

 });

 };

 var \_showMessage = function(title, msg) {

 msg = msg || title;

 \_sendCommand({

 command: 'showMessage',

 data: {

 msg: msg

 }

 });

 };

 var \_applyEditRights = function(allowed, message) {

 \_sendCommand({

 command: 'applyEditRights',

 data: {

 allowed: allowed,

 message: message

 }

 });

 };

 var \_processSaveResult = function(result, message) {

 \_sendCommand({

 command: 'processSaveResult',

 data: {

 result: result,

 message: message

 }

 });

 };

 // TODO: remove processRightsChange, use denyEditingRights

 var \_processRightsChange = function(enabled, message) {

 \_sendCommand({

 command: 'processRightsChange',

 data: {

 enabled: enabled,

 message: message

 }

 });

 };

 var \_denyEditingRights = function(message) {

 \_sendCommand({

 command: 'processRightsChange',

 data: {

 enabled: false,

 message: message

 }

 });

 };

 var \_refreshHistory = function(data, message) {

 \_sendCommand({

 command: 'refreshHistory',

 data: {

 data: data,

 message: message

 }

 });

 };

 var \_setHistoryData = function(data, message) {

 \_sendCommand({

 command: 'setHistoryData',

 data: {

 data: data,

 message: message

 }

 });

 };

 var \_setEmailAddresses = function(data) {

 \_sendCommand({

 command: 'setEmailAddresses',

 data: {

 data: data

 }

 });

 };

 var \_setActionLink = function (data) {

 \_sendCommand({

 command: 'setActionLink',

 data: {

 url: data

 }

 });

 };

 var \_processMailMerge = function(enabled, message) {

 \_sendCommand({

 command: 'processMailMerge',

 data: {

 enabled: enabled,

 message: message

 }

 });

 };

 var \_downloadAs = function(data) {

 \_sendCommand({

 command: 'downloadAs',

 data: data

 });

 };

 var \_setUsers = function(data) {

 \_sendCommand({

 command: 'setUsers',

 data: data

 });

 };

 var \_showSharingSettings = function(data) {

 \_sendCommand({

 command: 'showSharingSettings',

 data: data

 });

 };

 var \_setSharingSettings = function(data) {

 \_sendCommand({

 command: 'setSharingSettings',

 data: data

 });

 };

 var \_insertImage = function(data) {

 \_sendCommand({

 command: 'insertImage',

 data: data

 });

 };

 var \_setMailMergeRecipients = function(data) {

 \_sendCommand({

 command: 'setMailMergeRecipients',

 data: data

 });

 };

 var \_setRevisedFile = function(data) {

 \_sendCommand({

 command: 'setRevisedFile',

 data: data

 });

 };

 var \_setRequestedDocument = function(data) {

 \_sendCommand({

 command: 'setRequestedDocument',

 data: data

 });

 };

 var \_setRequestedSpreadsheet = function(data) {

 \_sendCommand({

 command: 'setRequestedSpreadsheet',

 data: data

 });

 };

 var \_setReferenceSource = function(data) {

 \_sendCommand({

 command: 'setReferenceSource',

 data: data

 });

 };

 var \_setFavorite = function(data) {

 \_sendCommand({

 command: 'setFavorite',

 data: data

 });

 };

 var \_requestClose = function(data) {

 \_sendCommand({

 command: 'requestClose',

 data: data

 });

 };

 var \_processMouse = function(evt) {

 var r = iframe.getBoundingClientRect();

 var data = {

 type: evt.type,

 x: evt.x - r.left,

 y: evt.y - r.top,

 event: evt

 };

 \_sendCommand({

 command: 'processMouse',

 data: data

 });

 };

 var \_grabFocus = function(data) {

 setTimeout(function(){

 \_sendCommand({

 command: 'grabFocus',

 data: data

 });

 }, 10);

 };

 var \_blurFocus = function(data) {

 \_sendCommand({

 command: 'blurFocus',

 data: data

 });

 };

 var \_setReferenceData = function(data) {

 \_sendCommand({

 command: 'setReferenceData',

 data: data

 });

 };

 var \_serviceCommand = function(command, data) {

 \_sendCommand({

 command: 'internalCommand',

 data: {

 command: command,

 data: data

 }

 });

 };

 function \_createEmbedWorker() { return AscEmbed.initWorker(iframe); }

 return {

 createEmbedWorker : \_createEmbedWorker,

 showMessage : \_showMessage,

 processSaveResult : \_processSaveResult,

 processRightsChange : \_processRightsChange,

 denyEditingRights : \_denyEditingRights,

 refreshHistory : \_refreshHistory,

 setHistoryData : \_setHistoryData,

 setEmailAddresses : \_setEmailAddresses,

 setActionLink : \_setActionLink,

 processMailMerge : \_processMailMerge,

 downloadAs : \_downloadAs,

 serviceCommand : \_serviceCommand,

 attachMouseEvents : \_attachMouseEvents,

 detachMouseEvents : \_detachMouseEvents,

 destroyEditor : \_destroyEditor,

 setUsers : \_setUsers,

 showSharingSettings : \_showSharingSettings,

 setSharingSettings : \_setSharingSettings,

 insertImage : \_insertImage,

 setMailMergeRecipients: \_setMailMergeRecipients,

 setRevisedFile : \_setRevisedFile,

 setFavorite : \_setFavorite,

 requestClose : \_requestClose,

 grabFocus : \_grabFocus,

 blurFocus : \_blurFocus,

 setReferenceData : \_setReferenceData,

 setRequestedDocument: \_setRequestedDocument,

 setRequestedSpreadsheet: \_setRequestedSpreadsheet,

 setReferenceSource: \_setReferenceSource

 }

 };

 DocsAPI.DocEditor.defaultConfig = {

 type: 'desktop',

 width: '100%',

 height: '100%',

 editorConfig: {

 lang: 'en',

 canCoAuthoring: true,

 customization: {

 about: true,

 feedback: false

 }

 }

 };

 DocsAPI.DocEditor.version = function() {

 return '8.0.1';

 };

 MessageDispatcher = function(fn, scope) {

 var \_fn = fn,

 \_scope = scope || window,

 eventFn = function(msg) {

 \_onMessage(msg);

 };

 var \_bindEvents = function() {

 if (window.addEventListener) {

 window.addEventListener("message", eventFn, false)

 }

 else if (window.attachEvent) {

 window.attachEvent("onmessage", eventFn);

 }

 };

 var \_unbindEvents = function() {

 if (window.removeEventListener) {

 window.removeEventListener("message", eventFn, false)

 }

 else if (window.detachEvent) {

 window.detachEvent("onmessage", eventFn);

 }

 };

 var \_onMessage = function(msg) {

 // TODO: check message origin

 if (msg && window.JSON && \_scope.frameOrigin==msg.origin ) {

 try {

 var msg = window.JSON.parse(msg.data);

 if (\_fn) {

 \_fn.call(\_scope, msg);

 }

 } catch(e) {}

 }

 };

 \_bindEvents.call(this);

 return {

 unbindEvents: \_unbindEvents

 }

 };

 function getBasePath(config) {

 if(config.iframeOrigin){

 return config.iframeOrigin

 }else{

 var scripts = document.getElementsByTagName('script'),

 match;

 for (var i = scripts.length - 1; i >= 0; i--) {

 match = scripts[i].src.match(/(.\*)api\/documents\/api.js/i);

 if (match) {

 return match[1];

 }

 }

 return "";

 }

 }

 function getExtensionPath() {

 if ("undefined" == typeof(extensionParams) || null == extensionParams["url"])

 return null;

 return extensionParams["url"] + "apps/";

 }

 function getTestPath() {

 var scripts = document.getElementsByTagName('script'),

 match;

 for (var i = scripts.length - 1; i >= 0; i--) {

 match = scripts[i].src.match(/(.\*)apps\/api\/documents\/api.js/i);

 if (match) {

 return match[1] + "test/";

 }

 }

 return "";

 }

 function getAppPath(config) {

 var extensionPath = getExtensionPath(),

 path = extensionPath ? extensionPath : (config.type=="test" ? getTestPath() : getBasePath(config)),

 appMap = {

 'text': 'documenteditor',

 'text-pdf': 'documenteditor',

 'spreadsheet': 'spreadsheeteditor',

 'presentation': 'presentationeditor',

 'word': 'documenteditor',

 'cell': 'spreadsheeteditor',

 'slide': 'presentationeditor',

 'pdf': 'pdfeditor'

 },

 appType = 'word';

 if (typeof config.documentType === 'string') {

 appType = config.documentType.toLowerCase();

 if (config.type !== 'mobile' && config.type !== 'embedded' && !!config.document && typeof config.document.fileType === 'string') {

 var type = /^(?:(pdf|djvu|xps|oxps))$/.exec(config.document.fileType);

 if (type && typeof type[1] === 'string')

 appType = 'pdf';

 }

 } else

 if (!!config.document && typeof config.document.fileType === 'string') {

 var type = /^(?:(xls|xlsx|ods|csv|xlst|xlsy|gsheet|xlsm|xlt|xltm|xltx|fods|ots|xlsb)|(pps|ppsx|ppt|pptx|odp|pptt|ppty|gslides|pot|potm|potx|ppsm|pptm|fodp|otp)|(pdf|djvu|xps|oxps))$/

 .exec(config.document.fileType);

 if (type) {

 if (typeof type[1] === 'string') appType = 'cell'; else

 if (typeof type[2] === 'string') appType = 'slide'; else

 if (typeof type[3] === 'string' && config.type !== 'mobile' && config.type !== 'embedded') appType = 'pdf';

 }

 }

 if (appType === 'pdf' && (config.type === 'mobile' || config.type === 'embedded')) {

 appType = 'word';

 }

 path += appMap[appType] + "/";

 const path\_type = config.type === "mobile"

 ? "mobile" : (config.type === "embedded")

 ? "embed" : "main";

 path += path\_type;

 var index = "/index.html";

 if (config.editorConfig && path\_type!=="forms") {

 var customization = config.editorConfig.customization;

 if ( typeof(customization) == 'object' && ( customization.toolbarNoTabs ||

 (config.editorConfig.targetApp!=='desktop') && (customization.loaderName || customization.loaderLogo))) {

 index = "/index\_loader.html";

 } else if (config.editorConfig.mode === 'editdiagram' || config.editorConfig.mode === 'editmerge' || config.editorConfig.mode === 'editole')

 index = "/index\_internal.html";

 }

 path += index;

 return path;

 }

 function getAppParameters(config) {

 var params = "?\_dc=8.0.1-31";

 if (config.editorConfig && config.editorConfig.lang)

 params += "&lang=" + config.editorConfig.lang;

 if (config.editorConfig && config.editorConfig.targetApp!=='desktop') {

 if ( (typeof(config.editorConfig.customization) == 'object') && config.editorConfig.customization.loaderName) {

 if (config.editorConfig.customization.loaderName !== 'none') params += "&customer=" + encodeURIComponent(config.editorConfig.customization.loaderName);

 } else

 params += "&customer=ONLYOFFICE";

 if (typeof(config.editorConfig.customization) == 'object') {

 if ( config.editorConfig.customization.loaderLogo && config.editorConfig.customization.loaderLogo !== '') {

 params += "&logo=" + encodeURIComponent(config.editorConfig.customization.loaderLogo);

 }

 if ( config.editorConfig.customization.logo ) {

 if (config.type=='embedded' && (config.editorConfig.customization.logo.image || config.editorConfig.customization.logo.imageEmbedded))

 params += "&headerlogo=" + encodeURIComponent(config.editorConfig.customization.logo.image || config.editorConfig.customization.logo.imageEmbedded);

 else if (config.type!='embedded' && (config.editorConfig.customization.logo.image || config.editorConfig.customization.logo.imageDark)) {

 config.editorConfig.customization.logo.image && (params += "&headerlogo=" + encodeURIComponent(config.editorConfig.customization.logo.image));

 config.editorConfig.customization.logo.imageDark && (params += "&headerlogodark=" + encodeURIComponent(config.editorConfig.customization.logo.imageDark));

 }

 }

 }

 }

 if (config.editorConfig && (config.editorConfig.mode == 'editdiagram' || config.editorConfig.mode == 'editmerge' || config.editorConfig.mode == 'editole'))

 params += "&internal=true";

 if (config.frameEditorId)

 params += "&frameEditorId=" + config.frameEditorId;

 var type = config.document ? /^(?:(pdf))$/.exec(config.document.fileType) : null;

 if (!(type && typeof type[1] === 'string') && (config.editorConfig && config.editorConfig.mode == 'view' ||

 config.document && config.document.permissions && (config.document.permissions.edit === false && !config.document.permissions.review )))

 params += "&mode=view";

 if (config.document) {

 config.document.isForm = (type && typeof type[1] === 'string') ? config.document.isForm : false;

 (config.document.isForm===true || config.document.isForm===false) && (params += "&isForm=" + config.document.isForm);

 }

 if (config.editorConfig && config.editorConfig.customization && !!config.editorConfig.customization.compactHeader)

 params += "&compact=true";

 if (config.editorConfig && config.editorConfig.customization && (config.editorConfig.customization.toolbar===false))

 params += "&toolbar=false";

 if (config.parentOrigin)

 params += "&parentOrigin=" + config.parentOrigin;

 if (config.editorConfig && config.editorConfig.customization && config.editorConfig.customization.uiTheme )

 params += "&uitheme=" + config.editorConfig.customization.uiTheme;

 if (config.document && config.document.fileType)

 params += "&fileType=" + config.document.fileType;

 return params;

 }

 function createIframe(config) {

 var iframe = document.createElement("iframe");

 iframe.src = getAppPath(config) + getAppParameters(config);

 iframe.width = config.width;

 iframe.height = config.height;

 iframe.align = "top";

 iframe.frameBorder = 0;

 iframe.name = "frameEditor";

 config.title && (typeof config.title === 'string') && (iframe.title = config.title);

 iframe.allowFullscreen = true;

 iframe.setAttribute("allowfullscreen",""); // for IE11

 iframe.setAttribute("onmousewheel",""); // for Safari on Mac

 iframe.setAttribute("allow", "autoplay; camera; microphone; display-capture; clipboard-write;");

 if (config.type == "mobile")

 {

 iframe.style.position = "fixed";

 iframe.style.overflow = "hidden";

 document.body.style.overscrollBehaviorY = "contain";

 }

 return iframe;

 }

 function postMessage(wnd, msg) {

 if (wnd && wnd.postMessage && window.JSON) {

 // TODO: specify explicit origin

 wnd.postMessage(window.JSON.stringify(msg), "\*");

 }

 }

 function extend(dest, src) {

 for (var prop in src) {

 if (src.hasOwnProperty(prop)) {

 if (typeof dest[prop] === 'undefined') {

 dest[prop] = src[prop];

 } else

 if (typeof dest[prop] === 'object' &&

 typeof src[prop] === 'object') {

 extend(dest[prop], src[prop])

 }

 }

 }

 return dest;

 }

})(window.DocsAPI = window.DocsAPI || {}, window, document);