

User Manual

3-Heights™ PDF Web Viewer API

Version 4.12.26.6



Contents

1	Introduction	3
1.1	Description	3
1.2	Features	3
1.3	Supported Input Formats	3
1.4	Compliance	3
1.5	Interfaces	4
1.6	Supported Browsers	4
1.7	How to Best Read this Manual	4
2	Installation and Deployment	5
2.1	Changing default folder structure	5
2.2	Setup IIS	5
2.2.1	web.config	6
2.3	Setup Apache	6
3	License Management	7
3.1	License Installation and Management	7
3.2	Troubleshooting	7
3.2.1	License key cannot be set	7
3.2.2	The current license does not permit running the product in this environment (e.g. domain)	7
4	Programming Interfaces	8
4.1	JavaScript	8
5	User's Guide	9
5.1	Using the Input Mode	9
5.2	Selected Items	9
5.2.1	Selected Items Enumeration	9
5.2.2	Annotation Item Enumeration	9
5.2.3	Freertext Annotation Item Enumeration	10
5.2.4	Selected Text Enumeration	10
5.3	Touch Handling on Mobile Devices	10
5.3.1	Touch handling on the document Enumeration	10
5.3.2	Touch handling on a annotation Enumeration	10
5.3.3	Touch handling on a popup annotation Enumeration	11
5.3.4	Touch handling while creating an annotation Enumeration	11
5.4	Rich Text for Free Text Annotations	11
5.4.1	Supported CSS2 Styles	12
5.5	Localisation of the PDF Web Viewer API	13
5.6	Stamp Annotations	14
5.7	Limitations of the 3-Heights™ PDF Web Viewer API	14
5.7.1	Color Management	14
5.7.2	System Fonts	14

6	Interface Reference	16
6.1	Enumerations	16
6.1.1	PdfPageLayoutMode Enumeration	16
6.1.2	PdfFitMode Enumeration	16
6.1.3	InputMode Enumeration	16
6.1.4	PdfItemType Enumeration	17
6.2	PDF Viewer Application Program Interface	18
6.2.1	Properties	18
	PageCount	18
	Rotation	18
	BorderSize	18
	FitMode	18
	PageLayoutMode	19
	PageNo	19
	Zoom	19
	IgnoringPreferences	19
	ProductVersion	19
6.2.2	Methods	19
	create	19
	open	20
	openFDF	20
	openBlob	20
	close	20
	saveFile	21
	showThumbnails	21
	isOpen	21
	addImageStampTemplate	21
	setInputMode	22
	disableContextMenu	22
	isContextMenuDisabled	22
	search	22
	configureSearcher	22
	setLicenseKey	23
	setAnnotationDefault	23
	hasChanges	23
6.2.3	Events	24
	eventBus.addListener	24

1 Introduction

1.1 Description

The 3-Heights™ PDF Web Viewer API is a compact, high-performance, high-quality PDF viewer. It offers a multitude of navigational and display options for displaying documents. This viewer is mainly characterized by its increased performance and its uniform and simple interface.

The PDF Web Viewer API has been optimized for displaying PDF/A files. For best viewing experience, it is recommended to convert PDF files using the 3-Heights™ PDF to PDF/A Converter before viewing.

1.2 Features

- Navigate manually (user action) or programmatically through a document
- Select between different fit modes: actual size, fit to width, fit to height
- Rotate and display the page
- Show thumbnails and use them for navigation
- Change language of the GUI (currently English and German supported)
- Create, edit and delete annotations
 - Text annotation
 - Ink annotation
 - Stamp annotation (Draft, Approved, etc.)
 - Add custom stamps (based on JPEG, PNG or TIFF files)
 - Freetext annotation
 - Highlight annotations (highlight, strike out, underline, squiggly)
- Touch handling for mobile devices
- Enter password to decrypt PDF documents
- Read document from file or memory or from a blob
- Supports the Forms Data Format (FDF) file format

1.3 Supported Input Formats

- PDF 1.x (PDF 1.0, ..., PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3
- FDF

1.4 Compliance

Standards:

- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)
- ISO 19005-1 (PDF/A-1)
- ISO 19005-2 (PDF/A-2)
- ISO 19005-3 (PDF/A-3)

1.5 Interfaces

The API is exposed via a JavaScript file.

1.6 Supported Browsers

- Chrome 63+
- Firefox 55+
- Edge 41+
- Safari 11.0.3+

Chrome, Firefox and Microsoft Edge support WebAssembly a new compact, size- and load-time-efficient format.

1.7 How to Best Read this Manual

If you are reading this manual for the first time, i.e. would like to evaluate the software, the following steps are suggested.

1. Read the chapter [Introduction](#) to verify this product meets your requirements.
2. Identify what interface your programming language uses.
3. Read and follow the instructions in the chapter [Installation and Deployment](#)
4. In the chapter [Programming Interfaces](#) find your programming language. For a start it is generally best to refer to these samples rather than writing code from scratch.
5. (Optional) Read the chapter [User's Guide](#) for general information about the API. Read the [Interface Reference](#) for specific information about the functions of the API.

2 Installation and Deployment

The 3-Heights™ PDF Web Viewer API comes as a ZIP archive containing various files including runtime binary executable code, files required for the developer, documentation and license terms.

1. Download the ZIP archive of the product from your download account at <https://www.pdf-tools.com>.
2. Unzip the file using a tool like WinZip available from WinZip Computing, Inc. at <http://www.winzip.com>
3. The contents of the zip file go into a folder on a webserver. We provide configuration files for both IIS and apache
4. Check the appropriate option to preserve file paths (folder names).

The unzip process now creates the following subdirectories:

Subdirectory	Description
doc	Contains the <code>htaccess.txt</code> , <code>web.config</code> and the license terms. The <code>htaccess.txt</code> and <code>web.config</code> are needed for the web server configuration
webapp	Contains the the sample files <code>index.html</code> and <code>viewerSampleUI.js</code>
webapp\Css	Contains the CSS files used in the sample
webapp\Fonts	Contains the fonts used in the sample
webapp\Images	Contains the images used in the sample
webapp\pdfwebviewer	Conteains the essential files for the viewer including <code>PdfWebViewer.js</code> which serves as API between the viewer and the sample.

2.1 Changing default folder structure

If you want to change the relative position of the `pdfwebviewer` folder with respect to the `index.html` two changes are needed:

- In `index.html` change `window.PDFTOOLS_WEBVIEWER_BASEURL` so it points to the new root of the `pdfwebviewer` folder.
- In addition the source path for the scripts `PdfWebViewer.js` has to be adjusted

```
<script type="text/javascript">window.PDFTOOLS_WEBVIEWER_BASEURL="new/root/"</script>  
<script type="text/javascript" src="new/root/pdfwebviewer/PdfWebViewer.js"></script>
```

2.2 Setup IIS

In order to run the viewer on an IIS web server copy the contents of the `webapp` folder into the desired directory on your web server. In addition to the `webapp` folder also copy the `doc\web.config` file to the target directory. This is needed to provide the proper caching options for the web viewer and how to treat certain file types such as `asm` and `wasm`.

2.2.1 web.config

web.config contains the MIME mapping for the different types used in the web viewer. Apart from the web.config there is also %SYSTEMROOT%\System32\inetsrv\Config\applicationHost.config and depending on the IIS version this file already contains some of the mimeMap entries. In this case remove the duplicate entries from the web.config.

2.3 Setup Apache

Similarly to the IIS installation the Apache config file htaccess.txt has to be copied from doc to the target folder.

3 License Management

3.1 License Installation and Management

The license key is passed to the application at run-time via the `setLicenseKey` method. The license keys are mapped to a domain name. If the domain name does not match the domain name set in the license key the verification will fail. As the `setLicenseKey` method returns a promise another key (if available) can be tried.

3.2 Troubleshooting

3.2.1 License key cannot be set

The license key cannot be set in the viewer application. The error message is: "Invalid license format."

Possible causes:

- There might be a typo in the key - or the formatting of the key is incorrect.

Solution

Make sure the key is of the form '1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX'. Check the license page on <https://www.pdf-tools.com/pdf20/en/mypdftools/> for the correct key.

3.2.2 The current license does not permit running the product in this environment (e.g. domain)

Possible causes:

- The license key is not meant to be used with the current domain.

Solution

Make sure that the domain on which the viewer is running matches the domain to which the key is registered.

4 Programming Interfaces

4.1 JavaScript

After extracting the PDF Web Viewer API you find a sample implementation of the viewer in the folder `webapp`. The sample for the web viewer is implemented in the `index.html` and `viewerSampleUI.js` where the `index.html` contains the HTML structure of the sample and `viewerSampleUI.js` the JavaScript code. It shows how to use the interface provided by `webapp\pdfwebviewer\PdfWebViewer.js`.

In order to use the viewer you need to run it on a web server. For Apache and IIS configuration files are provided in the download kit.

To run the sample, ensure to set the license key provided on the download page and enter it in the sample. To do this open the `viewerSampleUI.js` and search for [setLicenseKey](#) and replace the string with your key.

5 User's Guide

5.1 Using the Input Mode

With the [setInputMode](#) method there is complete access all the functions to create annotations. It can be used to customise the implementation of the viewer and still be able to achieve the same functionality as the provided sample.

It can be used to define custom menu bars which will allow the creating of any supported annotation and do not have to depend on the context menu (which in fact can be disabled if desired - [disableContextMenu](#)).

After setting an input mode the mouse or touch gestures behave according to the input mode selected. So, for example, if the input mode was set to `PLACE_STICKY` the next touch or mouse click will place a sticky note at the place where the touch/click occurred. See [InputMode](#) for the respective mouse and touch behaviours.

5.2 Selected Items

When subscribing to the event [selectedItems](#) the viewer will fire events when annotations or text is being selected. Depending on the item type, there will be different properties available.

5.2.1 Selected Items Enumeration

General properties of all items	
<code>Id</code>	Each item gets a unique ID. There is no guarantee that an item will have the same ID when a document is closed and then reopened. Selected text items will have an ID of -1.
<code>itemType</code>	See PdfItemType to see the correspondence between itemType and type.

5.2.2 Annotation Item Enumeration

General properties of all annotation items	
<code>color</code>	A string containing the color and opacity in the HTML RGBA format (e.g. <code>rgba(255,0,0,1.0)</code> means the annotation color is red and the annotation is fully opaque)
<code>content</code>	Content of the annotation. This will also be the content that is being showed in the popup (if existing) of the annotation
<code>lastModified</code>	Date when the annotation was last modified
<code>originalAuthor</code>	Author of the annotation

page	Page on which the annotation is located
rect	Provides the annotation location on the screen relative to the viewer div-container. Zooming and scrolling invalidates this information and the annotation item has to be requested again to get the proper location of the annotation.

5.2.3 Freetext Annotation Item Enumeration

Special properties of freetext annotations	
fontColor	Font color of the freetext annotation
fontSize	Font size
fontName	Font name
richText	Richtext string. See the Richtext section for more information.

5.2.4 Selected Text Enumeration

Special properties of selected text	
selectedText	The text which has been selected.

5.3 Touch Handling on Mobile Devices

When the viewer is used on a mobile device, touch handling is enabled. Following is an overview how the touch handling is set up with regard to different contexts.

5.3.1 Touch handling on the document Enumeration

Different Touch Gestures on the Document	
Long Touch	A long touch on the document opens the context menu
Touch Move	Moving the finger around on the screen scrolls around the document. "Flicking" movement can be used to initiate inertia scrolling.
Tap	A short tap on menu items, context menu entries, buttons executes the equivalent of a mouse press on those items. Also if text is selected a tap resets the text selection.
Two Touches	Two fingers can be used to either adjust the zoom (by pinching or stretching the distance between the fingers).

5.3.2 Touch handling on a annotation Enumeration

Different Touch Gestures on the annotation

Long Touch	A long touch on the annotation opens a context menu for the given annotation
------------	--

5.3.3 Touch handling on a popup annotation Enumeration

Different Touch Gestures on the popup

Long Touch	A long touch on the popup opens a context menu for the popup
Tap	Tapping inside the text area enables editing of the popup text. Touching the 'x' button closes the popup annotation and saves the currently entered text
Touch Move	Touch and move on the popupannotation (except within the text area) of a popup annotation will move the position of the popupannotation

5.3.4 Touch handling while creating an annotation Enumeration

Touch Gestures When Creating Annotations

Sticky Note	After selecting sticky note from the context menu a sticky note gets created where the context menu was opened. If the mouse mode was programatically set to PLACE_STICKY then another touch places the sticky note at the touch location
Stamp Annotations	When creating a stamp annotation touch and drag to increase and decrease the size of the stamp annotation. The aspect ratio is fixed to the stamp selected.
Free Text Annotation	Similar to stamp touch and drag to create a free text annotation.
Ink Annotation (freehand)	Touch and drag the finger around the document to create a freehand annotation
Highlight Annotations	Select text with a long touch on text. Expanders will appear to allow increasing and decreasing the selection. Use another long touch on the selection to bring up the context menu and select which type of highlight annotation should be created.

5.4 Rich Text for Free Text Annotations

Free text annotations can either have plain text content or rich text content. This section specifies the format of rich text content that the 3-Heights™ PDF Web Viewer API supports.

Rich text content strings in PDF have an XHTML format with a restricted set of CSS2 styles. The following describes all allowed XML elements (in blue) and their allowed XML attributes (in green).

body The unique root element. Should contain one paragraph (p).

xmlns (Required) Value "<http://www.w3.org/1999/xhtml>"

xmlns:xfa (Required) Value "<http://www.xfa.org/schema/xfa-data/1.0/>"

xfa:APIVersion (Required) Value "[Acrobat:18.11.0](#)"

xfa:spec (Required) Value "[2.0.2](#)"

style (Optional) Value see [Supported CSS2 Styles](#).

p A paragraph element. In contrast to web-browser-style rendering of XHTML, a paragraph does not induce a line break. Can contain spans (span).

style (Optional) Value see [Supported CSS2 Styles](#).

span An element that defines a common style for its content. Can contain spans (span).

style (Optional) Value see [Supported CSS2 Styles](#).

b The enclosed text behaves as if **style="font-weight:bold"**.

i The enclosed text behaves as if **style="italic"**.

br A line break. This element must always occur in the self-closed form (
) and is not allowed to have any attributes.

5.4.1 Supported CSS2 Styles

Rich text strings in PDF support a restricted set of CSS2 (Cascading Style Sheet) styles. All styles must be specified in a **style** attribute as follows:

style="<key>:<value>;<key>:<value>;..."

The following describes the supported **<key>**s and their possible **<value>**s:

text-align Specifies the text alignment. Possible values are:

- **left**
- **right**
- **center**
- **justify**

Example: **style="text-align:left"**

Note: The value given to **text-align** applies to the whole rich text string and therefore should be given only once in the **style** attribute of the **body** or of a single enclosing paragraph **p**.

text-decoration Specifies strike-through and underline text styles. Styles are combined by specifying multiple of the following values separated spaces:

- **line-through**: Draws a horizontal line through the words.
- **underline** or **word**: Underlines words either in a continuous line or as interrupted lines, one for each word, respectively.
- **double**: Draws the underline (continuous or word-wise) as a double line.

Example: **style="text-decoration:underline line-through"**

color Specifies the color for the text and text decoration (strike-through and underline). The value must have the following form: **#<rr><gg><bb>**, where **<rr>**, **<gg>**, and **<bb>** are 2-digit hexadecimal numbers ranging from **00** to **ff** that indicate the red, green, and blue value of the color in an RGB color-space.

Example: `style="color:#a2358c"`

font-family Specifies the font to be used. Currently the following values are supported:

- Helvetica
- Times
- Courier
- Symbol
- ZapfDingbats

Example: `style="font-family:Helvetica"`

font-size Specifies the size of the font in the format `<number>pt`, where `<number>` is a floating point number.

Example: `style="font-size:12pt"`

font-style The only supported values are `italic` and `normal`.

Example: `style="font-style:italic"`

font-weight The only supported values are `bold` and `normal`.

Example: `style="font-weight:bold"`

font This is an abbreviated form for setting `font-style`, `font-weight`, `font-size`, and `font-family` in a single style value. The values for the above four styles can be specified in any order, separated by spaces.

Example: `style="Helvetica 14pt bold"`

The default style settings are as follows:

```
text-align:left
font-family:Helvetica
font-size:12pt
font-style:normal
font-weight:normal
color:#000000
```

5.5 Localisation of the PDF Web Viewer API

When changing the localisation it is important to notice that the localisation only concerns the library part of the viewer, more specifically it has no effect on the sample (everything that is inside the `index.html` and the corresponding JavaScripts and CSS files). Changing the localisation will change the language of the default context menu (if enabled) and also the language of the default stamps.

To change the language edit the locale variable inside the `gwt:property` meta tag in `index.html`:

```
<meta name="gwt:property" content="locale=de">
```

There are localisations for following languages:

Localisation Value	Language
en	English (default)
de	German

fr	French
it	Italian

5.6 Stamp Annotations

When creating stamp annotations one can choose between default text stamps or adding custom image stamps. When creating default text stamps they will be created with their text localised (see [Localisation of the PDF Web Viewer API](#)). To create a default stamp use the [setInputMode](#) method with the [InputMode](#) set to `PLACE_STAMP` and one of the following strings as option:

- "SBApproved"
- "SBNotApproved"
- "SBDraft"
- "SBFinal"
- "SBCompleted"
- "SBConfidential"
- "SBForPublicRelease"
- "SBNotForPublicRelease"
- "SBForComment"
- "SBVoid"
- "SBPreliminaryResults"
- "SBInformationOnly"

5.7 Limitations of the 3-Heights™ PDF Web Viewer API

Due to the restrictions given in the web environment, there are some limitations to the capabilities of the PDF Web Viewer API, as listed below.

5.7.1 Color Management

The PDF Web Viewer API uses an RGB colorspace for rendering and converts all resources into this colorspace for rendering. When viewing the rendered output on a display screen this is not noticeable, as such screens can only display additive color (RGB colors) anyway. However there may be some noticeable differences when printing out the rendered output, due to printed pages using subtractive color (reflected light, e.g. cmyk colors).

Effect: The colors of non-rgb content (e.g. cmyk images) can differ slightly from the original when printing rendered output to physical printer.

5.7.2 System Fonts

Many PDF's rely on not embedding the used fonts in the pdf but instead referring to a common font, which is assumed to be installed on the local computer. Due to Browsers protection mechanisms a downloaded script like the PDF Web Viewer API is not allowed to directly access these fonts. Thus the PDF Web Viewer API analyzes the referenced fonts and tries to replace them with a standard font, which is packaged with the PDF Web Viewer API. However these standard fonts are limited and fonts which dont use a standard encoding or use uncommon glyphs (e.g. foreign alphabets such as cyrillic) cannot always be properly replaced.

Effect: Text in non-embedded fonts which uses special glyphs may be rendered using wrong glyphs or not be rendered at all.

Note: This issue can be avoided by converting all files to PDF/A before viewing. This ensures that all fonts are embedded in the PDF file.

6 Interface Reference

In the following sections we document the interface of the PdfWebViewerAPI when using JavaScript. The interface is contained in the PdfWebViewer.js file which glues the html frontend with the viewer backend.

6.1 Enumerations

6.1.1 PdfPageLayoutMode Enumeration

PdfPageLayoutMode Table

PdfPageLayoutMode	
SinglePage	Shows one single page, scrolling when reaching the top/bottom of the page will jump to the next/previous page.
OneColumn	Shows one column of pages, that can be scrolled through.
TwoColumnLeft	Shows a column of pairs of pages, that can be scrolled through. The first page is placed left.
TwoColumnRight	Shows a column of pairs of pages, that can be scrolled through. The first page is placed as a separate title page at the top.
TwoPageLeft	Shows a pair of pages, scrolling when reaching the top/bottom of the page will jump to the next/previous page pair. The first page is placed left.
TwoPageRight	Shows a pair of pages, scrolling when reaching the top/bottom of the page will jump to the next/previous page pair. The first page is placed as a separate title page at the top.

6.1.2 PdfFitMode Enumeration

FitMode Table

PdfFitMode	
FitPage	The window is zoomed to fit the whole page into the viewport.
FitWidth	The window is zoomed to fit the page's width into the viewport. If there are multiple columns of pages shown, the viewport will fit to that width.
ActualSize	The window is zoomed to reflect the true size of the page.

6.1.3 InputMode Enumeration

InputMode

InputMode	
DEFAULT(0)	Default behaviour of the viewer. Navigation is turned on and text and annotations can be selected
PLACE_STICKY(2)	Next click or touch will place a sticky not at that location
CREATE_HIGHLIGHT(3)	Create a highlight annotation for the selected text. Does nothing if no text is selected
CREATE_UNDERLINE(4)	Create a underline annotation for the selected text. Does nothing if no text is selected
CREATE_SQUIGGLY(5)	Create a squiggly annotation for the selected text. Does nothing if no text is selected
CREATE_STRIKE_OUT(6)	Create a strike-out annotation for the selected text. Does nothing if no text is selected
PLACE_FREE_TEXT(7)	Places a freetext annotation. Drag with your mouse or touch to span a rectangle to define the size of the freetext annotation. A single click creates a default sized freetext annotation.
START_INK(8)	Disables navigation and allows to draw ink lines. Several ink lines can be drawn. Use FINISH_INK to confirm and create the annotation.
FINISH_INK(9)	Creates an ink annotation based on the points placed during START_INK .
PLACE_STAMP(10)	Places a stamp (specified by the option argument in setInputMode . Drag with your mouse or touch to spawn a rectangle to define the size of the stamp. A single click creates a default sized stamp.)

6.1.4 PdfItemType Enumeration

Below is a list of all the different

PdfItemType

PdfItemType	
UNKNOWN(0)	Unknown item type
TEXT(1)	Sticky note
LINK(2)	Link annotation
FREE_TEXT(3)	Free text annotation
HIGHLIGHT(9)	Text highlight annotation
UNDERLINE(10)	Text underline annotation

PdfItemType

SQUIGGLY(11)	Text squiggly annotation
STRIKE_OUT(12)	Text strikethrough annotation
STAMP(13)	Stamp annotation
INK(15)	Ink annotation
POPUP(16)	Popup annotation
SELECTED_TEXT(3000)	This is an item that contains the currently selected text. It is not an annotation and hence can not be deleted or moved.

6.2 PDF Viewer Application Program Interface

Listed below are the property names. In order to get/set them append get or set before the property name i.e. for getting the `PageNo` property use `getPageNo` and to set it use `setPageNo`.

6.2.1 Properties

PageCount

Property (get): `int PageCount`

The number of pages in the opened document. 0 if no Document open.

Rotation

Property (get, set): `int Rotation`
Default: `0`

The "rotate" angle in multiples of 90 degrees. Each individual page is rotated by the given angle.

BorderSize

Property (get, set): `double BorderSize`
Default: `6.0`

The border displayed in between pages in user units.

FitMode

Property (get, set): `PdfFitMode FitMode`

The FitMode used for displaying the document. See also [PdfFitMode](#).

PageLayoutMode

Property (get, set): PdfPageLayoutMode PageLayoutMode
Default: PageLayoutDocument

The PageDisplayMode used for displaying the document. See also [PdfPageLayoutMode](#).

PageNo

Property (get, set): int PageNo

The topmost pagenumber on the viewport.

Zoom

Property (get, set): double Zoom
Default: 1.0

The zoomFactor that the document is displayed with.

IgnoringPreferences

Property (get, set): bool IgnoringPreferences
Default: false

When this property is set to **True** before opening a file, the viewer will ignore all viewing preferences that are embedded in said file. This includes open destination and the preferred layout mode.

ProductVersion

Property (get): String ProductVersion

Get the version of the 3-Heights™ PDF Web Viewer API in the format "A.C.D.E".

6.2.2 Methods

create

Method: Promise create(string viewerContainer)

Parameter:

viewerContainer [string] Name of the div container in which the viewer should be displayed

Returns:

A promise returning the viewer object if resolved or returning an error message if rejected.

Constructor of of the viewer. The argument is the name of an element in which the viewer should be displayed.

open

```
Method: Promise open(Uint8Array buffer, String password)
```

Parameters:

buffer [Uint8Array] Buffer holding the pdf in memory

password [String] Password needed to decrypt the pdf

Opens a new PDF file. Closes the currently opened file.

openFDF

```
Method: Promise openFDF(Uint8Array pdfBuffer, Uint8Array fdfBuffer, String password)
```

Parameters:

pdfBuffer [Uint8Array] Buffer holding the pdf in memory

fdfBuffer [Uint8Array] Buffer holding the fdf in memory

password [String] Password needed to decrypt the pdf

Opens a PDF file with an FDF file. The opened file will contain the annotations defined in the FDF and display them in the PDF. Closes the currently opened file.

openBlob

```
Method: Promise openBlob(blob blob, string password)
```

Parameters:

blob [blob] Blob of the file

password [string] Password needed to decrypt the pdf

Opens a PDF file with a blob. Closes the currently opened file.

close

```
Method: Promise close()
```

Closes the currently opened file.

Note: The promise returned by `open`, `openFDF`, `openBlob` and `close` returns void if resolved or an error message if rejected

saveFile

Method: `saveFile(bool asFDF)`

Parameter:

asFDF [bool] If set to true, the file will be saved as FDF instead of PDF

Save the currently opened file.

showThumbnails

Method: `showThumbnails(bool showThumbnails)`

Parameter:

showThumbnails [bool] True if thumbnails show be displayed, false otherwise

Show or hide the thumbnail view in the viewer.

isOpen

Method: `bool isOpen()`

Whether a file is opened. A file counts as open as soon as opening it has completed successfully until closing it has started.

addImageStampTemplate

Method: `addImageStampTemplate(string name, Uint8Array image, bool addToContextMenu)`

Add a stamp with a custom image to the viewer. The stamp will be added to the context menu and can also be created with `setInputMode` with the stamp name as `option`.

Parameters:

name [string] Name of the stamp - shows up in the context menu. Has to be unique. If a stamp with the same name already exists it will be overwritten.

image [Uint8Array] Image of the desired stamp.

addToContextMenu [bool] Set to false to prevent adding a new context menu entry. The stamp will still be placeable via `setInputMode(PLACE_STAMP, "name")` where `PLACE_STAMP` refers to the input mode enum `InputMode` and `name` is the name parameter above.

setInputMode

Method: `setInputMode(PdfInputMode inputMode, string option)`

Allows to control the the behaviour of mouse and touch control of the viewer. Depending on the `InputMode` chosen, touch and mouse have different functions. For more information about the behaviour of different input modes see [InputMode](#).

disableContextMenu

Method: `disableContextMenu(bool disable)`

Used to disable the default context menu.

isContextMenuDisabled

Method: `bool isContextMenuDisabled()`

search

Method: `Promise search(string toSearch, number startPage, number startIndex)`

Parameters:

toSearch [string] String that should be searched for

startPage [number] Page on which the search starts

startIndex [number] Index within the page at which the search is started

Returns:

A promise returning a `SearchResult` if the promise resolved or a an error message if it was rejected. See [configureSearcher](#) on how to set up the searcher Searches for the provided text in the currently opened document.

configureSearcher

Method: `configureSearcher(bool matchCase, bool wrap, bool previous, bool useRegex)`

Parameters:

matchCase [bool] Set whether the search should consider the case when matching

wrap [bool] Set whether the search should wrap around the document when searching

previous [bool] If set to true, the search goes backwards

useRegex [bool] Set whether the search string should be interpreted as regular expression

Sets up the searcher.

setLicenseKey

```
Method: Promise setLicenseKey(string license)
```

Parameter:

license [string] License key

Returns:

A promise returning void if resolved or returning an error message if rejected

Set the license key of the viewer.

setAnnotationDefault

```
Method: setAnnotationDefault(Object annotationDefault)
```

Parameter:

annotationDefault [Object] Set the default values for annotations. The default properties that can be set are:

- **color**: Color of the annotation
- **content**: Content associated with the annotation (shows up in the popup for inside the freetext annotation)
- **originalAuthor**: Author of the annotation
- **fontColor**: Color of the font (freetext only)
- **borderWidth**: Set border or line width for annotations
- **richText**: Rich text string for freetext annotations

Use this method right before setting the input mode to change the default appearance of the annotation. Calling the method repeatedly always overwrites the previous settings.

hasChanges

```
Method: bool hasChanges()
```


6.2.3 Events

Events: There are several events which one can subscribe to. The Viewer API provides an event bus with an `addEventListener` method. The signature of this method is `addEventListener(eventName, function(e) {...})`

`eventBus.addEventListener`

Method: `eventBus.addEventListener(string eventName, function callback)`

Event Names

Events	
<code>firstVisiblePage</code>	Fires when the first visible page changed.
<code>lastVisiblePage</code>	Fires when the last visible page changed.
<code>zoom</code>	Fires when zoom has changed.
<code>fitMode</code>	Fires when the fit mode changed.
<code>pageLayoutMode</code>	Fires when the page layout mode changed.
<code>rotation</code>	Fires when the rotation changed
<code>busyState</code>	This event fires with true or false, depending on whether the viewer is busy rendering or not
<code>selectedItems</code>	When an item has been selected e.g. an annotation this event fires with the information about the selected item.
<code>onError</code>	This error fires when the viewer crashes without the option of recovery. This means to reload the viewer instance