

PDF Web Viewer

PDF/A conform

Version 4.3.5

pdftools

Contents

1 Introduction . . . 3
1.1 Description . . . 3
1.2 Features . . . 3
1.3 Supported input and output formats . . . 3
1.4 Conformance . . . 4
1.5 Supported browsers . . . 4

2 Installation and deployment . . . 5
2.1 Static asset management . . . 6
2.1.1 Base URL . . . 6
2.1.2 Manual copying . . . 6
2.1.3 Automated copying . . . 6
2.2 Basic usage . . . 7
2.3 Server configuration . . . 8
2.3.1 Windows IIS . . . 8
2.3.2 Apache . . . 9
2.4 License keys . . . 9
2.4.1 Troubleshooting . . . 9

3 User guide . . . 11
3.1 Graphical user interface . . . 11
3.1.1 Document bar . . . 12
3.1.2 Navigation bar . . . 12
Page navigation . . . 12
Page display . . . 12
3.1.3 Information bar . . . 13
Text search . . . 13
Information pane . . . 13
Thumbnails . . . 14
Outline . . . 14
Annotations . . . 14
3.1.4 Annotation bar . . . 14
Sticky note annotation . . . 14
Ink annotation . . . 15
Eraser tool . . . 15
Free-text annotation . . . 15
Highlight annotation . . . 15
Rectangle and ellipse annotations . . . 15
Stamp annotation . . . 15
Text stamp annotation . . . 15
Image stamp annotation . . . 15
3.1.5 Touch handling . . . 16
Touch gestures on document . . . 16
Touch gestures on annotation . . . 16
Touch gestures while creating annotation . . . 16
3.2 Viewing-only edition . . . 16
3.3 Range requests . . . 17
3.4 General limitations . . . 17
3.4.1 Color management . . . 17

3.4.2 System fonts . . . 18
3.5 Custom translations . . . 18
3.6 Keyboard shortcuts . . . 18
3.6.1 Example configurations for PdfWebViewerOptions . . . 18
3.7 Custom buttons . . . 19
3.7.1 Example configurations for PdfWebViewerOptions . . . 19

4 Interface reference . . . 21
4.1 Enumerations . . . 21
4.1.1 PdfPageLayoutMode . . . 21
4.1.2 StampAnnotationColor . . . 21
4.2 WebViewerOptions . . . 21

5 Version history . . . 27
5.1 Changes in Version 4 . . . 27
5.2 Changes in Version 3 . . . 29
5.3 Changes in Version 2 . . . 35

6 Licensing, copyright, and contact . . . 36

A How to Migrate from 3-Heights® . . . 37
A.1 Prerequisite . . . 37
A.2 Upgrade . . . 37

1 Introduction

1.1 Description

The PDF Web Viewer is a compact, high performance, high-quality viewer for PDF documents, offering a multitude of navigational and display options. It runs entirely in a web browser or JavaScript/WebAssembly container.

1.2 Features

- Responsive UI with no external dependencies and support for mobile devices
- Navigate manually (user action) or programmatically through a document
- Select among six different layout modes for one-page and facing-pages viewing
- Select different fit modes: actual size, fit to width, fit to page
- Pre-configure allowed zoom levels
- Rotate the page display
- Search the document for a given text
- Languages English, German, French, Italian are pre-configured and any custom translation can be plugged in
- Show thumbnails and use them for navigation
- Show the document outline (bookmarks) and use it for navigation
- Show annotations and use them for navigation
- Annotations and form fields can be filtered before saving to PDF or FDF
- Fill out form fields (check boxes, radio buttons, list boxes, drop down, text fields)
- Create, edit and delete annotations
  - Sticky-note annotations
  - Text highlight annotations (highlight, strike out, underline, squiggly)
  - Text annotations
  - Free-hand drawings
  - Eraser tool for deleting individual lines in free-hand drawings
  - Predefined text stamp annotations (Draft, Approved, etc.)
  - Use a PDF's pages as stamp annotations
  - Use an image as stamp annotation
  - Rectangle and ellipse annotations
- Touch handling for mobile devices
- Enter password to open encrypted PDF documents
- Read document from file, URL or blob
- Support loading and storing annotations in the Forms Data Format (FDF)
- Prevent the modification and deletion of all annotations except the currently configured author's
- Viewing-only mode with annotation editing and form filling disabled
- Print the document
- Easy evaluation without the necessity of a license key
- Available on npmjs.com

1.3 Supported input and output 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 Conformance

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 Supported browsers

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

'+' indicates the minimum supported version.

2 Installation and deployment

The PDF Web Viewer comes either as an npm package or as a ZIP archive containing an example JavaScript web application.

The npm package is provided via npmjs.com and can be installed with:

```
npm i @pdf-tools/four-heights-pdf-web-viewer --save
```

The npm package complies with Node version 14.x or higher and contains the following:

Subdirectory	Description
`css`	The compiled CSS
`doc`	Documentation
`es6`	Source code
`pdfwebviewer`	Static assets. See Static asset management
`scss`	Sass CSS source files
`umd`	Main JavaScript and source code mappings
`wasm`	WebAssembly and data files
`LICENSE.md`	General licensing agreement
`README.md`	Quickstart read-me file

As an alternative to the npm package, the PDF Web Viewer can be obtained from the "LICENSES & KITS"-area of your account at `https://www.pdf-tools.com/` as a self-contained example JavaScript web application in the form of a ZIP archive with the following content:

Subdirectory	Description
`doc`	License terms and documentation
`webapp`	■ `minimal.html`: A minimal example HTML with the default WebViewerOptions ■ `index.html`: An example HTML with a wide range of WebViewerOptions ■ `.htaccess`, `web.config`: Server configuration files for Apache and Windows IIS
`webapp/pdfwebviewer`	Static assets. See Static asset management Source code mapping files for development: ■ `pdf-web-viewer.development.js` ■ `pdf-web-viewer.development.map.js`

2.1 Static asset management

The static assets need to be served by the web server in order for the PDF Web Viewer to function. They are contained in the package's subdirectory `pdfwebviewer`:

- Main JavaScript file:
  - `pdf-web-viewer.min.js`
- Compiled CSS:
  - `pdf-web-viewer.css`
- WebAssembly, associated JavaScript, and data files:
  - `PdfViewing.data`
  - `PdfViewing_Main.js`, `PdfViewing_Main.wasm`
  - `PdfViewing_Worker.js`, `PdfViewing_Worker.wasm`
- Translation files:
  - `translations.en.json`
  - `translations.de.json`
  - `translations.fr.json`
  - `translations.it.json`

After installing or updating the PDF Web Viewer, ensure that the static assets are copied from the packet's `pdfwebviewer` subdirectory to the "base URL" location. Unused translation files can be omitted and custom translation files can be added. See Custom translations.

2.1.1 Base URL

The static assets need to be served with the targeted application from a "base URL". The base URL is configured by setting the JavaScript property `window.PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL`. The base URL can be a path relative to the main HTML file or an absolute path. For example:

- `"./path/to/pdfwebviewer"` (relative to `index.html`)
- `"/file/viewer/pdfwebviewer"` (absolute file path)
- `"file:///e:/electron-app/pdfwebviewer"` (absolute file path)

The base URL must be set prior to loading the PDF Web Viewer. For example, in the `head` element of the main HTML file as:

```
<script type="text/javascript">
  window.PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL = './pdfwebviewer/'
</script>
```

2.1.2 Manual copying

Manual copying of the assets files in a shell is possible, but not recommended, because it is easily forgotten. For example, when using the npm package, `PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL` is defined as `'./pdfwebviewer/'`, and your main directory for static assets is `static`:

```
cp -R ./node_modules/@pdf-tools/four-heights-pdf-web-viewer/pdfwebviewer ./static
```

2.1.3 Automated copying

When using the npm package, it is recommended to use a bundler which copied the static assets for you on every build.

For example, for webpack:

```
const path = require('path')
const CopyWebpackPlugin = require('copy-webpack-plugin')

const pdfwebviewerDir = path.join(
  path.dirname(require.resolve('@pdf-tools/four-heights-pdf-web-viewer')),
  '../pdfwebviewer'
)

module.exports = {
  ...
  plugins: [
    new CopyWebpackPlugin({
      patterns: [
        {
          from: '**/*',
          to: 'pdfwebviewer', // Should match PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL
          context: pdfwebviewerDir,
        }
      ],
    }),
  ],
}
```

For example, for Angular in `angular.json`:

```
"projects": {
  "angular": {
    "architect": {
      "build": {
        "options": {
          "assets": [
            {
              "glob": "**/*",
              "input":
                "./node_modules/@pdf-tools/four-heights-pdf-web-viewer/pdfwebviewer",
              "output":
                "./pdfwebviewer" // Should match PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL
            },
          ],
        },
      },
    },
  },
}
```

2.2 Basic usage

A simple web application that uses the PDF Web Viewer with the default WebViewerOptions is implemented as follows.

`index.html`:

```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script type="text/javascript">
      // path to static assets must be set before the viewer is loaded
      window.PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL = './pdfwebviewer/'
    </script>
    <link rel="stylesheet" type="text/css" href="./pdfwebviewer/pdf-web-viewer.css" />
    <title>PDF Web Viewer</title>
  </head>
  <body>
    <!-- HTM element containing the PdfWebViewer. -->
    <div id="pdfviewer" style="height: 100vh; width: 100vw"></div>
  </body>
</html>
```

`index.js`:

```
import { PdfWebViewer } from '@pdf-tools/four-heights-pdf-web-viewer'

const element = document.getElementById('pdfviewer')
const license = ''

// use default options
const options = {}

const pdfViewer = new PdfWebViewer(element, license, options)
```

Finally, don't forget to set a valid license key in `index.js` at the designated location. See License keys.

2.3 Server configuration

It is necessary to configure the correct content types for the `*.wasm` and `*.data` files.

2.3.1 Windows IIS

The content types can be configured for a Windows IIS web server by means of the following `web.config` file:

```
<?xml version="1.0" encoding="UTF-8"?>
  <configuration>
    <system.webServer>
      <staticContent>
        <mimeMap fileExtension=".mem" mimeType="application/octet-stream" />
        <mimeMap fileExtension=".data" mimeType="application/octet-stream" />
        <mimeMap fileExtension=".wasm" mimeType="application/wasm" />
      </staticContent>
    </system.webServer>
  </configuration>
```

In this file, you may additionally want to configure caching settings for the PDF Web Viewer's static assets.

Apart from `web.config` there is also `%SYSTEMROOT%\System32\inetsrv\Config\applicationHost.config`. Depending on the IIS version, this file already contains some of the MIME type mapping entries. In this case, the duplicate entries should be removed from `web.config`.

2.3.2 Apache

The content types can be configured for an Apache web server by means of the following `.htaccess` file:

```
AddType application/octet-stream ".mem"
AddType application/octet-stream ".data"
AddType application/wasm         ".wasm"
```

In this file, you might additionally want to configure caching settings for the PDF Web Viewer's static assets.

2.4 License keys

The PDF Web Viewer runs both with or without setting a license key. If no license is set, a watermark is applied by default.

If you don't want a watermark to be applied, a key must be obtained from the "LICENSES & KITS"-area of your account at https://www.pdf-tools.com/. To create an evaluation license key, do either of the following:

- Go to the the free trial page, select PDF Web Viewer, and create an evaluation license key. (You may have to create an account first.)
- Write to pdfsales@pdf-tools.com to request a key.

This key must be passed as a second argument to the `PdfWebViewer` constructor. See Basic usage.

Only keys starting with `<4H,V4,VIEWWEB,...` can be used with this version of the PDF Web Viewer.

A license key is bound to a specific domain, e.g. `example.com`. This domain is checked at runtime by the PDF Web Viewer and compared against the license.

Note that the PDF Web Viewer comes in two editions: The "Standard Edition" and the "Viewing-only edition", configured in the license key.

2.4.1 Troubleshooting

License key cannot be set

If the license key cannot be set, the call to the `PdfWebViewer` constructor results in an error.

Possible causes:

- There is a typographic error in the key.
- A copy-paste operation has altered the symbol characters in the key, such as the commas.
- The key has a wrong format.
- The key belongs to a different product.
- The key is an evaluation license key and it has expired.

Solution:

Make sure the key is issued for the PDF Web Viewer and has the following format:

```
<4H,V4,VIEWWEB,XXXXXXXXXXXXXXXXXXX>
```

If an evaluation license key has expired and you want to extend the period of evaluation, then please contact pdf-sales@pdf-tools.com for a new evaluation license key.

License check fails at runtime

In this case, the license can be set and the call to the constructor `PdfWebViewer` is successful. But later, when opening a PDF document, an error is generated with the message: "The current license does not permit running the product on this domain."

Possible causes:

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

Solution:

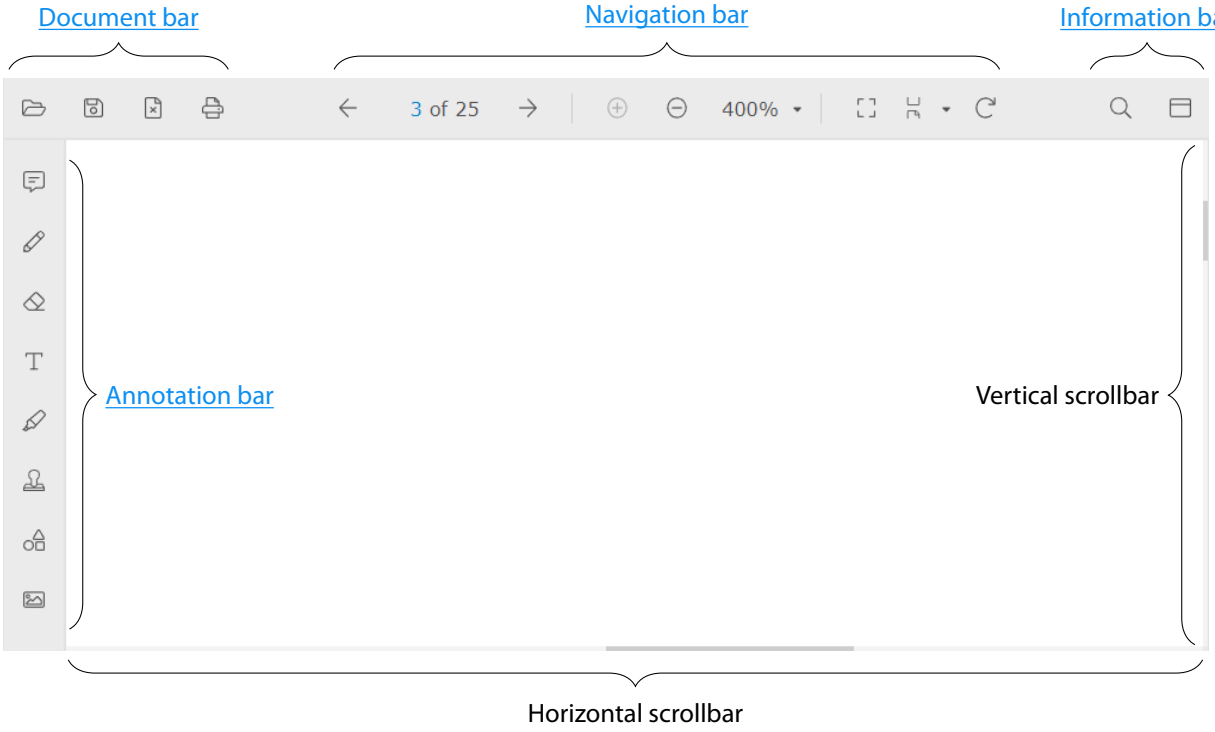
Make sure that the domain on which the PDF Web Viewer is running matches the domain to which the license key is registered.

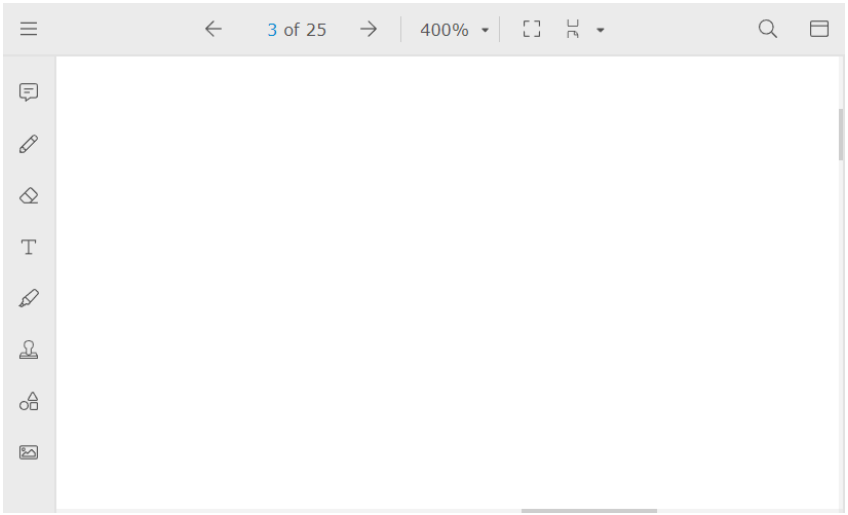
3 User guide

3.1 Graphical user interface

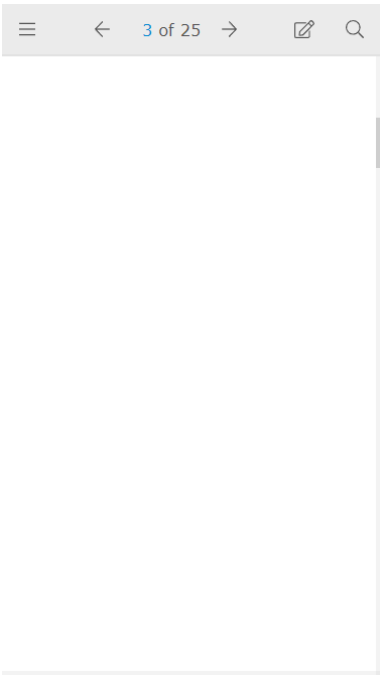
Once a PDF is opened, the graphical user interface of the PDF Web Viewer presents itself as depicted below. Depending on the device size, various elements collapse or disappear.

Main GUI elements – desktop





Main GUI elements – tablet



Main GUI elements – mobile

Various main attributes about the GUI, such as the language, the position of the annotation bar, and tooltips can be configured in the viewer.general.

3.1.1 Document bar

This bar comprises the following document-level operations:

- Open a new document. The PDF Web Viewer also allows to open a document by drag-and-drop.
- Save the currently displayed document. This triggers the browser-specific download action.
- Close the currently displayed document.
- Print the currently displayed document. A print dialog appears that lets the user select the pages to print. The selected pages are then rendered into images an handed over to the browser's print dialog.

The availability of each of the above actions can be configured in the viewer.permissions.

Modified documents: When trying to open a new document without saving the currently modified document, a dialog is shown to let the user choose whether to save or discard the current document.
When closing the web browser (tab) or navigating in the browser history, no such dialog is shown.

Printing: The rendering-based printing architecture has the following implications:

- The time and memory used while preparing the print depends on the number of selected pages.
- Certain elements such as text and vector graphics can turn out slightly blurred on the printout, because the printer's resolution can exceed the rendering resolution, which is set to 150 DPI.

3.1.2 Navigation bar

This bar comprises navigation and viewing operations.

Page navigation

A document's pages can be navigated by:

- The arrow buttons in the navigation bar.
- A scrolling action triggered by operating the scroll bars, the mouse wheel, and a swipe gesture on touch devices.
- The thumbnails, outline items, and annotation comments displayed in the Information pane.

Page display

Zoom

The view's zoom factor can be adjusted by:

- The "plus" and "minus" buttons.
- The zoom selection drop-down button. The selection can be configured in the viewer.general.
- The "CTRL and mouse wheel" action or a pinch gesture on touch devices.

Fit mode

The view can fit a document's page, and thereby change the zoom factor and the scroll position, by selecting one of the following fitting modes:

Actual size Set the zoom factor to 100%.

Fit width Set the zoom factor such that width of the current page fits the width of the view.

Fit page Set the zoom factor such that the current page fits into the view.

Page layout

The page layout can be chosen among the following:

- Continuous display: The scrollable area comprises all pages of a document.
  One column Display the pages in a single column.
  Two column left Display the pages side-by-side with odd-numbered pages on the left and even-numbered pages on the right.
  Two column right Display the pages side-by-side with odd-numbered pages on the right and even-numbered pages on the left.
- Single page display: The scrollable area is confined to a single page or two facing pages.
  Single page Display a single page.
  Two page left Display two pages side-by-side with the odd-numbered page on the left and the even-numbered page on the right.
  Two page right Display two pages side-by-side with the odd-numbered page on the right and the even-numbered page on the left.

The available selection of fit modes can be configured in the viewer.general.

Rotation

The page display can be rotated by means of the "rotate" button. This does not modify the document.

3.1.3 Information bar

The information bar comprises Text search and toggling of the Information pane.

Text search

Text can be searched in the document by clicking on the search button, which opens a text search area.

Text search is incremental: As soon as a text is typed into the search field, the first resulting hit is highlighted.

Hits can be navigated by the arrow buttons or by pressing "enter".

The search can be configured by clicking on the wrench button as follows:

Case sensitive Toggle the searche's case sensitivity.

Wrap search Toggle whether navigating through the search hits wraps from the end of the document to the beginning.

Information pane

The information pane's visibility is toggled by clicking on the information pane button. When visible, this bar offers three tabs.

Thumbnails

When activating the "Thumbnails" tab, then thumbnails of the document's pages are shown in the information pane. Clicking on a thumbnail navigates to the respective page.

Outline

When activating the "Outline" tab, then the document outline is shown in the information pane as a tree of potentially expandable outline items, also known as "bookmarks".

Clicking on an outline item navigates to the destination associated with this item. Depending on the type of destination, this may not only change the active page but also the zoom factor.

Note that not all documents have an outline.

Annotations

When activating the "Annotations" tab, then an overview of the document's annotations is shown in the information pane.

In this overview, every annotation is listed with the following constituents:

Icon Equivalent to the icons shown in the Annotation bar, additionally colored by the annotations main color.

Author Only shown if set for this annotation.

Date and time Indicates when the annotation was created.

Subject Only shown if set for this annotation.

Text This is the text associated with the annotation.

3.1.4 Annotation bar

Annotations are remarks placed on a document's pages by a reader of the document. They are not part of the actual page content and can easily be removed.

The annotation bar contains buttons for creating new annotations. The following annotation types are supported:

- Sticky note annotation
- Ink annotation
- Eraser tool
- Free-text annotation
- Highlight annotation
- Rectangle and ellipse annotations
- Stamp annotation

Once a button is selected, the viewer goes into a mode in which clicking onto a page starts the creation of the selected type of annotation.

All annotation types support the entry of a text to be shown in a popup dialog or—in the case of Free-text annotations—directly on the page. An annotation popup dialog is a small window that shows the text, the author, the date and time of creation and the subject. A popup can have two states: closed (invisible) or open (visible).

Every annotation has a primary color.

Annotations can be stored directly in a PDF or outside of a PDF in an additional FDF file.

The remainder of this section describes the supported annotation types.

Sticky note annotation

A sticky note consist of an icon placed on a page.

Ink annotation

An ink annotations consists of free-hand lines drawn onto a page. The line color and thickness can be adjusted.

All lines pertaining to a single annotation are contained on the same page and have the same color and thickness.

Eraser tool

The eraser can be used to delete individual lines in Ink annotations. When deleting all lines, then the entire annotation is deleted.

Free-text annotation

Free-text annotations are the only annotations which do not have an annotation popup. The annotation text is drawn in a rectangular area directly on the page.

Free-text annotations support basic text formatting.

Highlight annotation

A highlight annotation is used to highlight text by either of the following methods:

- Overlay with a color
- Underline
- Underline with a squiggly line
- Strike through

Rectangle and ellipse annotations

Rectangle and ellipse annotations are used to draw rectangles and ellipses in a specified color onto the page.

Stamp annotation

Stamp annotations are used to place a predefined text or image onto the page.

Text stamp annotation

The following predefined texts can be selected from:

- "Approved" (default)
- "Not approved"
- "Draft"
- "Final"
- "Completed"
- "Confidential"
- "For Public Release"
- "Not for Public Release"
- "Void"
- "For Comment"
- "Preliminary Result"
- "Information only"

In contrast to other annotation types, text stamp annotations come in fixed predefined colors.

Image stamp annotation

When clicking on the image stamp annotation button, a file selection dialog opens for the user to select an image file.

The following image file formats are supported:

- BMP (1, 2, 4, 8, 24 bit)
- GIF (2 to 8 bit)
- JBIG2 (lossless compression)
- JPEG, JPEG2000 and JPEGLS (grayscale, RGB)
- PBM and PNG (1 to 8, 24 bit)
- TIFF:
  - Bitonal : uncompressed, CCITT G3, CCITT G32D, CCITT G4, LZW, ZIP, Packbits
  - Grayscale, RGB and CMYK: uncompressed, LZW, JPEG, JPEG (old), ZIP, Packbits

3.1.5 Touch handling

When the viewer is used on a mobile device, touch handling is enabled. The following is an overview of the touch handling in different contexts.

Touch gestures on document

Tap A short tap on a menu item or context menu executes the equivalent of a mouse click.

Touch move A touch and move with one finger scrolls the document up and down or left and right.

Two touches Two fingers can be used to adjust the zoom by pinching or stretching the distance between the fingers.

Two touches move A touch and move with two fingers move the document in any direction.

Touch gestures on annotation

Tap A short tap on a annotation opens a context menu for the given annotation type.

Touch move After selecting an annotation with a tap, a touch and move on the annotation (except highlight annotations) moves the position of the annotation.

Touch gestures while creating annotation

Sticky note After selecting the sticky note from the annotation bar, a tap places the sticky note at the tapped location.

Free-text annotation After selecting the free-text annotation from the annotation bar, touch and drag defines the annotation's rectangle.

Text stamp annotation After selecting the text stamp annotation from the annotation bar, touch and drag defines the stamp's size while preserving the correct aspect ratio for the selected text.

Highlight annotation Text is selected with a touch and drag on the text.

Ink annotation Touch and drag is used to draw the lines of an ink annotation.

3.2 Viewing-only edition

The PDF Web Viewer comes in a "viewing-only" edition, which is configured in the license key. This edition has the following limitations:

- Saving the PDF results in an error.

- Creating annotations results in an error.

Because of these limitations, it is recommended that you disable these functionalities by doing the following:

- In the viewer.permissions set `allowSaveFile` to `false`.
- In the WebViewerOptions remove all entries from the `modules`.

Currently the user interface allows to move, delete, and edit annotations. Saving the document is, however, not possible.

3.3 Range requests

When opening a PDF from an URL with the `openFile` method, the PDF Web Viewer is capable of using "HTTP range requests" to load only those parts of the PDF needed for the current display. This feature can cut down initial load times for large PDFs considerably.

Each range request make by the PDF Web Viewer has a chunk size of about 130 kB and is executed as a stateless HTTP request.

A web server needs to set the following flags to allow range requests:

Range request settings

Key	Value	Comment

`Access-Control-Expose-Headers`	`Content-Length`
	`Accept-Ranges`
	`Content-Range`
		These headers have to be contained in a response from the host in order to allow range requests.

`Access-Control-Allow-Headers`	`range`	The request from the viewer has to be allowed to contain `range` in the request header.

If the server does not support range requests, then the PDF Web Viewer needs to download the entire PDF prior to start displaying.

Also, the server should be able to access the PDF in a random-access way. Otherwise, each range request can force the server to re-load the entire file.

3.4 General limitations

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

3.4.1 Color management

The PDF Web Viewer uses an RGB color space for rendering. There may be some noticeable differences when printing a document that uses CMYK color spaces.

Effect: The colors of non-RGB content can differ slightly from the original when printing.

3.4.2 System fonts

A font referenced in a PDF can be either embedded in the PDF or it is assumed to be present on the host system. Due to browsers protection mechanisms, downloaded JavaScript such as the PDF Web Viewer is not allowed to directly access system fonts. The PDF Web Viewer analyzes the referenced fonts and replaces them with one of the standard fonts packaged with the PDF Web Viewer. Due to this replacement, glyph shapes can be altered or not rendered at all, e.g. for CJK, Cyrillic or other alphabets.

Effect: Texts that use non-embedded fonts which use special glyphs may be rendered distorted or not at all.

> **Note:** This issue can be avoided by converting a PDF to PDF/A before viewing. This ensures that all fonts are embedded.

3.5 Custom translations

The PDF Web Viewer can be parametrized in the predefined languages English, German, French, and Italian. Furthermore, a custom translation can be defined. Generally, translations are stored in files of the form `translations.[language].json` that have to reside in the same directory as the WebAssembly (i.e. `window.PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL`). The language used must be set in `language` from GeneralOptions. If a key in the `translations.[language].json` is missing, an error is thrown. Translation files that are not used can be removed.

3.6 Keyboard shortcuts

The PDF Web Viewer provides the possibility to customize keyboard shortcuts. The interface reference for the specifiable shortcuts can be found in shortcuts. In some cases, a shortcut is only effective when the corresponding option is switched on. For example, `shortcuts.print` is only effective when `viewer.permissions.allowPrinting` is `true`.

In order to specify your own keyboard shortcuts, the common structure is used:

`KeyboardShortcutBinding`

```
KeyboardShortcutBinding {
  key: string
  altKey?: boolean
  ctrlKey?: boolean
  shiftKey?: boolean
}
```

3.6.1 Example configurations for PdfWebViewerOptions

Use default options for shortcuts:

```
"viewer": {
}
```

Disable all shortcuts:

```
"viewer": {
  "shortcuts": null
}
```

Disable specific shortcuts:

```
"viewer": {
  "shortcuts":{
    "print": null,
    "rotateView": null
  }
}
```

Change key binding:

```
"viewer": {
  "shortcuts":{
    "zoomIn": { "key": "+" },
    "zoomOut": { "key": "-" },
    "nextPage": { "key": "PageDown" },
    "previousPage": { "key": "PageUp" },
    "save": {
      "key": "s",
      "altKey": false,
      "ctrlKey": true,
      "shiftKey": false,
    }
  }
}
```

3.7 Custom buttons

The PDF Web Viewer provides the possibility to add user-defined buttons to the Document bar, the Information bar, and the Annotation bar.

In order to specify your own custom buttons, the common structure is used:

`CustomButton`

```
CustomButton {
  icon: string
  text: string
  onClick: () => void
}
```

3.7.1 Example configurations for PdfWebViewerOptions

Adding a close button to the Information Bar:

```
const options = {
  viewer: {
    customButtons: {
```

```
      documentbar: [],
      informationbar: [
        {
          icon: 'images/close-icon.svg'
          text: 'Close document'
          onClick: () => {
            pdfViewer.close()
          }
        }
      ],
      annotationbar: []
    }
  }
}
const pdfViewer = new PdfWebViewer(document.getElementById('pdfviewer'), '', options)
```

4 Interface reference

4.1 Enumerations

4.1.1 PdfPageLayoutMode

Name	Description
SINGLE_PAGE	Display a single page at a time.
ONE_COLUMN	Display a column of pages.
TWO_COLUMN_LEFT	Display two columns of pages next to each other. The first page is on the left.
TWO_COLUMN_RIGHT	Display two columns of pages next to each other. The first page is on the right.
TWO_PAGE_LEFT	Display two pages next to each other. The first page is on the left.
TWO_PAGE_RIGHT	Display two pages next to each other. The first page is on the right.

4.1.2 StampAnnotationColor

Name	Description
GREEN	The stamp annotation has color green.
RED	The stamp annotation has color red.
BLUE	The stamp annotation has color blue.

4.2 WebViewerOptions

The PDF Web Viewer provides many options to be configured in `index.html`.

`viewer`

`viewer.general`

`viewer.general.user` Set the name of the user.
Type: `string`
Default: `''`, i.e. no user set

`viewer.general.language` Set language of the tooltips as a two letter language code. See also Custom translations.
Type: `string`
Allowed values: `'en'`, `'de'`, `'fr'`, `'it'` and custom languages.
Default: `'en'`

viewer.general.promptOnUnsavedChange If an opened document has unsaved changes and the close or open button is pressed an 'unsaved changes' dialog appears. This dialog can be suppressed by setting `false`.
Type: `boolean`
Default: `true`

viewer.general.pageShadow Define the page shadow. The color is a CSS color string. The unit of the other values are in screen pixel.
Type: CanvasShadowStyles
Default: not set
Example:

```
CanvasShadowStyles {
  shadowBlur: number;
  shadowColor: string;
  shadowOffsetX: number;
  shadowOffsetY: number;
  }
```

viewer.general.currentPageShadow Define the page shadow for the current page. The color is a CSS color string. The unit of the other values are in screen pixel.
Type: CanvasShadowStyles
Default: not set
Example:

```
CanvasShadowStyles {
  shadowBlur: number;
  shadowColor: string;
  shadowOffsetX: number;
  shadowOffsetY: number;
  }
```

viewer.general.tooltips Tooltips can be turned on with `'title'` and turned off with `'none'`. If you want use your own CSS stylesheet to configure the tooltips, set `'css'`.
Type: `string`
Allowed values: `'none'`, `'title'`, `'css'`
Default: `'title'`

viewer.general.rectangularTextSelection Rectangular text selection can be turned on with `'automatic'` and turned off with `'none'`.
Type: `string`
Allowed values: `'none'`, `'automatic'`
Default: `'none'`

viewer.sidebar

viewer.sidebar.thumbnailNavigation Define if thumbnails are visible in the Information pane.
Type: `boolean`
Default: `true`

viewer.sidebar.outlineNavigation Define if outlines are visible in the Information pane.
Type: `boolean`
Default: `true`

viewer.sidebar.annotationNavigation Define if annotations are visible in the Information pane.
Type: `boolean` | `object`
Default: `true`

viewer.sidebar.annotationNavigation.textMarkup.preview Define if the preview for text markup annotations is enabled in the Information pane.
Type: `string`
Allowed values: `'none'`, `'short'`
Default: `'none'`

viewer.permissions

viewer.permissions.allowFileDrop Specify if the files can be opened with drag and drop.
Type: `boolean`
Default: `true`

viewer.permissions.allowSaveFile Specify if the files can be saved.
Type: `boolean`
Default: `true`

viewer.permissions.allowCloseFile Specify if the files can be close.
Type: `boolean`
Default: `false`

viewer.permissions.allowOpenFile Specify if the files can be opened.
Type: `boolean`
Default: `true`

viewer.permissions.allowPrinting Specify if the files are allowed to be printed.
Type: `boolean`
Default: `false`

viewer.permissions.enableSearch Set if a file can be searched through.
Type: `boolean`
Default: `true`

viewer.customButtons For further information and examples, see Custom buttons.

viewer.customButtons.documentbar Add buttons to the document bar.
Type: `CustomButton[]`
Default: empty array

viewer.customButtons.informationbar Add buttons to the information bar.
Type: `CustomButton[]`
Default: empty array

viewer.customButtons.annotationbar Add buttons to the annotation bar.
Type: `CustomButton[]`
Default: empty array

annotations

annotations.defaultStampWidth Set width of the Stamp annotation when it is initially created. The unit of the value is in PDF point.

Type: `number`
Default: `120`

`annotations.hideAnnotationSubject` Annotation subject is not visible.
Type: `boolean`
Default: `false`

`annotations.hideOnDelete` If the value is set to `true`, if the delete annotation button is pressed for an annotation, it is not deleted, but is flagged as hidden. In this case, the annotation remains visible in the annotation pane.
Type: `boolean`
Default: `false`

`annotations.onlyAuthorCanEdit` Changes to annotations can only be applied when the current user is the author of the annotation at hand.
Type: `boolean`
Default: `false`

`shortcuts` For further information and examples, see Keyboard shortcuts.

`shortcuts.zoomIn` Zoom In
Type: `KeyboardShortcutBinding`
Default: `{ key: '+' }`

`shortcuts.zoomOut` Zoom Out
Type: `KeyboardShortcutBinding`
Default: `{ key: '-' }`

`shortcuts.nextPage` Next Page
Type: `KeyboardShortcutBinding`
Default: `{ key: 'PageDown' }`

`shortcuts.previousPage` Previous Page
Type: `KeyboardShortcutBinding`
Default: `{ key: 'PageUp' }`

`shortcuts.firstPage` Go to First Page
Type: `KeyboardShortcutBinding`
Default: `{ key: 'Home' }`

`shortcuts.lastPage` Go to Last Page
Type: `KeyboardShortcutBinding`
Default: `{ key: 'End' }`

`shortcuts.scrollUp` Scroll Up
Type: `KeyboardShortcutBinding`
Default: `{ key: 'ArrowUp' }`

`shortcuts.scrollDown` Scroll Down
Type: `KeyboardShortcutBinding`
Default: `{ key: 'ArrowDown' }`

`shortcuts.scrollLeft` Scroll Left
Type: `KeyboardShortcutBinding`
Default: `{ key: 'ArrowLeft' }`

shortcuts.scrollRight Scroll Right
Type: KeyboardShortcutBinding
Default: { key: 'ArrowRight' }

shortcuts.releaseSelection Release Selection
Type: KeyboardShortcutBinding
Default: { key: 'Escape' }

shortcuts.copy Copy Selected Text
Type: KeyboardShortcutBinding
Default: { key: 'c', ctrlKey: true }

shortcuts.print Open Print Dialog
Type: KeyboardShortcutBinding
Default: { key: 'p', ctrlKey: true }

shortcuts.cancelPrint Cancel Printing
Type: KeyboardShortcutBinding
Default: { key: 'Escape' }

shortcuts.resetZoom Zoom to 100%
Type: KeyboardShortcutBinding
Default: { key: 'z', altKey: true }

shortcuts.fitToPage Fit to Page
Type: KeyboardShortcutBinding
Default: { key: 'p', altKey: true }

shortcuts.fitToWidth Fit to Width
Type: KeyboardShortcutBinding
Default: { key: 'w', altKey: true }

shortcuts.rotateView Rotate View
Type: KeyboardShortcutBinding
Default: { key: 'r', altKey: true }

shortcuts.save Save or Save Callback
Type: KeyboardShortcutBinding
Default: { key: 's', ctrlKey: true }

shortcuts.search Search
Type: KeyboardShortcutBinding
Default: { key: 'f', ctrlKey: true }

shortcuts.searchNext Next Search Match
Type: KeyboardShortcutBinding
Default: { key: 'F3' }

shortcuts.searchPrevious Previous Search Match
Type: KeyboardShortcutBinding
Default: { key: 'F3', shiftKey: true }

shortcuts.closeSearch Close Search
Type: KeyboardShortcutBinding
Default: { key: 'Escape' }

shortcuts.toggleSidePane Toggle Side Pane
Type: KeyboardShortcutBinding
Default: `{ key: 's', altKey: true }`

shortcuts.showAnnotations Show Annotation Pane
Type: KeyboardShortcutBinding
Default: `{ key: 'a', altKey: true }`

shortcuts.showOutline Show Outline Pane
Type: KeyboardShortcutBinding
Default: `{ key: 'o', altKey: true }`

shortcuts.showThumbnails Show Thumbnails Pane
Type: KeyboardShortcutBinding
Default: `{ key: 't', altKey: true }`

5 Version history

5.1 Changes in Version 4

Changes in Version 4.3

- **Fixed** Updated default license key
- **Fixed** Stop emitting error on thumbnail loading if document is closed before operation is finished
- **Fixed** Colors rendered in different than expected hues in specific color spaces
- **Fixed** Annotations not always saved on ipad
- **Fixed** Copy content key combination
- **New** Redaction icon for custom button to call the SDK redaction feature
- **Fixed** Image stamps can now be resized down to their original size if smaller than our minimal defined size
- **Fixed** Memory exception happening on large documents in single page mode
- **Fixed** issue with saved shape annotations with 1pt width stroke disappearing when editing and reloading multiple times.
- **Fixed** transparent sticky annotations now have a fallback black color.
- **New** Safari optimisations, HTTP options are now provided on a separate build.

Changes in Version 4.2

- **New** `WebViewerOption` to disable the main and the annotation toolbar. This can be used to create your own custom toolbars.
  ```
  viewer.general.disableMainToolbar: boolean
  viewer.general.disableAnnotationToolbar: boolean
  ```
- **New** `WebViewerOption` to disable the the annotation lock feature.
  ```
  viewer.permissions.allowLockAnnotations: boolean = true
  viewer.permissions.allowEditLockedAnnotations: boolean = false
  ```
- **New** method `PdfWebViewer.getSelectedText(): string`.
  Returns the selected text.
- **New** method `PdfWebViewer.print(): void`.
  Opens the print dialog.
- **New** method `PdfWebViewer.zoomIn(): void`.
  Sets the zoom to the closest higher predefined zoom Level.
- **New** method `PdfWebViewer.zoomOut(): void`.
  Sets the zoom to the closest smaller predefined zoom Level.
- **New** method `PdfWebViewer.toggleInformationPane(): void`.
  Toggles the visibility of the information pane.
- **New** event `textSelected: string` for `PdfWebViewer`.
- **New** event `documentClosed: void` for `PdfWebViewer`.

Changes in Version 4.1

- **New** `WebViewerOption` to highlight the current page in the PDF Web Viewer.
  ```
  viewer.general.currentPageShadow
    shadowBlur: number
    shadowColor: string
  ```

```
shadowOffsetX: number
shadowOffsetY: number
```

- **New** `WebViewerOption` to define custom permissions for annotations.
  - `annotation.annotationPermissionCallback: (annotation: Annotation, user: string) => boolean`
- **Improved** eraser tool. The radius for the eraser can now be adjusted in the ui.

Patch 4.0.1

- **Fixed** an issue where `hasChanges()` returned `null` when multiple types of changes were made.
- **Fixed** an issue that prevented form fields from being filled out.
- **Fixed** textsearch no longer interprets braces as part of a regular expression unless regular expression option is enabled.
- **Fixed** rotation for stampannotations on rotated pages.
- **Fixed** interaction with annotations on right half of the viewport in SinglePage mode.

Changes in Version 4.0

- **New** `WebViewerOption` to disable view rotation.
  - `viewer.permissions.allowRotateView`
- **New** `WebViewerOption` to disable page layout mode.
  - `viewer.permissions.enablePageLayoutMode`
- **New** `WebViewerOption` to open the page navigation with a specific tab.
  - `viewer.sidebar.selectedNavigation?: 'thumbnail' | 'outline' | 'annotation'`
- **Improved** edit existing annotations
  - Shape annotations
  - Ink annotations
  - Highlight annotations
- **New** tool to create rectangular highlight annotations.
- **New** tool to create rectangular highlight annotations.
- **New** properties for `HttpOptions` for the `open` method:
  - `mode?: 'cors' | 'no-cors' | 'same-origin'`
  - `cache?: 'default' | 'no-cache' | 'reload' | 'force-cache' | 'only-if-cached'`
  - `credentials?: 'include' | 'same-origin' | 'omit'`
  - `redirect?: 'manual' | 'follow' | 'error'`
  - `referrerPolicy?: 'no-referrer' | 'no-referrer-when-downgrade' | 'origin' | 'origin-when-cross-origin' | 'same-origin' | 'strict-origin' | 'strict-origin-when-cross | 'unsafe-url'`
- **New** properties for `ViewOptions` for the `open` method:
  - `initialPageNumber?: number`
  - `initialZoom?: number`
- **Improved** general visual appearance and user experience.

5.2 Changes in Version 3

Patch 3.11.1

- **New** option `WebViewerOptions.viewer.permissions.allowCopyText` to prevent copying of text from the PDF document. The default value is **true**.
- **Fixed** When calling `startSearch` multiple times, the search input field is now updated with the new search term.
- **Fixed** an issue that prevented the PDF Web Viewer from evaluation without configuring a license key.

Changes in Version 3.11

- **New** method `PdfWebViewer.getPageNumber(): number | null`.
  Get the currently most visible page.
- **New** method `PdfWebViewer.getZoom(): number | null`.
  Get the viewer zoom.
- **New** method `PdfWebViewer.getFitMode(): PdfFitMode | null`.
  Get the current fit mode.
- **New** method `PdfWebViewer.getPageLayoutMode(): PdfPageLayoutMode | null`.
  Get the current page layout mode.
- **New** method `PdfWebViewer.getRotation(): number | null`.
  Get the viewer rotation.
- **New** method `PdfWebViewer.hasChanges(): DocumentChange | null`.
  Check whether there are unsaved changes in the document.
- **New** method `PdfWebViewer.showInformationPane(selectedPane?: 'thumbnail' | 'outline' | 'annotation'): void`.
  Opens the information pane and selects the given pane
- **New** method `PdfWebViewer.hideInformationPane(): void`.
  Closes the information pane
- **New** method `PdfWebViewer.startSearch(text: string, searchOptions: SearchOptions): void`.
  Opens the search tool bar and starts a new search with the given text
- **New** method `PdfWebViewer.nextSearchMatch()`.
  Highlights the next search result
- **New** method `PdfWebViewer.previousSearchMatch()`.
  Highlights the previous search result
- **New** method `PdfWebViewer.endSearch()`.
  Stops the current search and closes the search tool bar
- **New** events for `PdfWebViewer`.
  `pageNumberChanged: number`
  `zoomChanged: number`
  `rotationChanged: number`
  `fitModeChanged: PdfFitMode`
  `pageLayoutModeChanged: PdfPageLayoutMode`
  `documentChanged: void`

Patch 3.10.1

- **Fixed** annotation filtering yielding unpredictable results when filtering annotations from PDF files with duplicate annotation names.
- **Fixed** saving to FDF no longer returns errors if any annotations are illegal to save to FDF, but only if it is actually being saved.

Changes in Version 3.10

- **New** eraser tool for deleting individual lines in free-hand drawings.
- **New** translation keys available for the eraser tool:
  - `eraser.begin`
  - `eraser.deleteFreeDrawingLines`
- **New** keyboard shortcut to cancel the current annotation tool. The default shortcut binding is Escape.
  - `shortcuts.cancelEditAnnotation`
- **New** method `open(pdfFile, fdfFiles?, password?, renderOptions?, viewOptions?)`.
  Open a PDF and optionally one or more associated FDFs.
- **Deprecated** `openFile` Use the new `open` method instead.
- **Deprecated** `openFDF` Use the new `open` method instead.
- **New** method `save(saveOptions)`.
  Return the opened PDF as `Blob` to be used, e.g., to save the file to disk.
- **Deprecated** `saveFile` Use the new `save` method instead.

Changes in Version 3.9

- **New** preview for text markup annotations (highlight, underline, strike out) in annotation information pane.
  `sidebar.annotationNavigation.textMarkup.preview`

Changes in Version 3.8

- **New** support for custom buttons, configurable with the option.
  `viewer.customButtons.documentbar`
  `viewer.customButtons.informationbar`
  `viewer.customButtons.annotationbar`

Changes in Version 3.7

- **New** support for rectangular text selection, configurable with the option.
  `viewer.general.rectangularTextSelection`

Changes in Version 3.6

- **Changed** rendering of form fields. Interactive form fields are now highlighted.
- **New** support for rotating free-text annotations.
- **New** support for custom shortcuts. Specifiable shortcuts in the WebViewerOptions:
  - `shortcuts.zoomIn`
  - `shortcuts.zoomOut`
  - `shortcuts.nextPage`

    - `shortcuts.previousPage`
    - `shortcuts.firstPage`
    - `shortcuts.lastPage`
    - `shortcuts.scrollUp`
    - `shortcuts.scrollDown`
    - `shortcuts.scrollLeft`
    - `shortcuts.scrollRight`
    - `shortcuts.releaseSelection`
    - `shortcuts.copy`
    - `shortcuts.print`
    - `shortcuts.cancelPrint`
    - `shortcuts.resetZoom`
    - `shortcuts.fitToPage`
    - `shortcuts.fitToWidth`
    - `shortcuts.rotateView`
    - `shortcuts.save`
    - `shortcuts.search`
    - `shortcuts.searchNext`
    - `shortcuts.searchPrevious`
    - `shortcuts.closeSearch`
    - `shortcuts.toggleSidePane`
    - `shortcuts.showAnnotations`
    - `shortcuts.showOutline`
    - `shortcuts.showThumbnails`
- **New** translation key `contextbar.openPopup`.

Changes in Version 3.5

- **New** support for evaluation without setting any license key.
- **Changed** behavior when operating without a productive license. A watermark is shown if no license key is set or when setting an evaluation license key. An evaluation license with disabled watermark can be provided upon request.

Changes in Version 3.4

- **New**: The PDF Web Viewer is now available on npmjs.com

Changes in Version 3.3

- **Improved** Ink annotations:
    - Improved resizing behavior
    - Improved bounding box computation for large line widths
    - Improved drawing of small ink lines
    - Line caps and joins changed to "round"

Changes in Version 3.2

- **Improved** mobile and responsive view:
    - Responsive collapsing of toolbars

    - The outline-pane is now available for small mobile devices
    - The annotations-pane is now available for small mobile devices
- **Changed** CSS: the CSS has been completely redesigned.
    - Existing customer-specific style adjustments may have to be revised.
    - The font family is no longer overwritten but inherited from the document.

Changes in Version 3.1

- **New** support for tooltips for all supported languages (English, German, French and Italian). With the new option `WebViewerOptions.viewer.general.tooltips` a tooltip mode can be set. Allowed values:
    - Turn on tooltips: `'title'`
    - Turn off tooltips: `'none'`
    - Use proper CSS to configure the tooltips: `'css'`
- **Improved** translation for French.
- **New** support for a close button. This button can be enabled or disabled with the new option `WebViewerOptions.permissions.allowCloseFile`. The new callback `onCloseFileButtonClicked?()` can optional be used.
- **Changed** icons for freetext annotation and fitmode page.
- **New** support for external translation files in json format. Supported keys are:
    - `openFile.dropFileHere`
    - `openFile.openFileDisabled`
    - `openFile.openDocument`
    - `openFile.selectFile`
    - `openFileError.description`
    - `openFileError.ok`
    - `loadFile.title`
    - `saveFile.title`
    - `passwordForm.description`
    - `passwordForm.ok`
    - `passwordForm.cancel`
    - `passwordForm.passwordRequiredError`
    - `passwordForm.invalidPasswordError`
    - `applicationLoader.title`
    - `applicationError.title`
    - `applicationError.reload`
    - `applicationError.defaultMessage`
    - `applicationError.invalidLicense`
    - `unsavedChanges.description`
    - `unsavedChanges.save`
    - `unsavedChanges.dontSave`
    - `unsavedChanges.cancel`
    - `search.inputPlaceholder`
    - `search.optionCaseSensitive`
    - `search.optionWrap`
    - `search.optionRegularExpression`
    - `search.previousMatch`
    - `search.nextMatch`
    - `search.options`
    - `sideNavigation.thumbnails`
    - `sideNavigation.outline`
    - `sideNavigation.annotation`

- sideNavigation.annotation.page
- sideNavigation.annotation.history
- pageLayoutMode.oneColumn
- pageLayoutMode.singlePage
- pageLayoutMode.twoColumnLeft
- pageLayoutMode.twoColumnRight
- pageLayoutMode.twoPageLeft
- pageLayoutMode.twoPageRight
- fitMode.page
- fitMode.width
- fitMode.none
- toolbar.openDocument
- toolbar.closeDocument
- toolbar.print
- toolbar.saveDocument
- toolbar.nextPage
- toolbar.previewsPage
- toolbar.pageOf
- toolbar.zoom
- toolbar.zoomIn
- toolbar.zoomOut
- toolbar.rotateView
- toolbar.search
- toolbar.toggleSidePane
- annotText.add
- annotFreeDrawing.add
- annotFreeDrawing.opacity
- annotFreeDrawing.strokeWidth
- annotFreeDrawing.undo
- annotFreeDrawing.addNew
- annotFreeDrawing.deleteLine
- annotFreeText.add
- annotFreeText.edit
- annotFreeText.fontFamily
- annotFreeText.fontSize
- annotFreeText.fontColor
- annotFreeText.bgColor
- annotFreeText.borderWidth
- annotFreeText.alignLeft
- annotFreeText.alignCenter
- annotFreeText.alignRight
- annotFreeText.bold
- annotFreeText.italic
- annotFreeText.underline
- annotFreeText.lock
- annotFreeText.unlock
- annotFreeText.subjectPlaceholder
- annotHighlight.add
- annotHighlight.highlight
- annotHighlight.underline
- annotHighlight.squiggly
- annotHighlight.strikeOut

- annotStamp.add
- annotStamp.chooseStamp
- annotShape.add
- annotShape.addRectangle
- annotShape.addEllipse
- annotShape.strokeColor
- annotShape.strokeWidth
- annotShape.strokeStyle
- annotShape.bgColor
- annotImage.add
- annotPopup.delete
- annotPopup.lock
- annotPopup.unlock
- annotPopup.color
- annotPopup.subjectPlaceholder
- annotPopup.contentPlaceholder
- contextbar.rotate
- contextbar.editPopup
- contextbar.addPopup
- contextbar.deletePopup
- contextbar.lock
- contextbar.unlock
- contextbar.delete
- contextbar.confirmDelete
- contextbar.cancelDelete
- contextbar.copyText
- contextbar.highlightText
- contextbar.underlineText
- contextbar.squigglyText
- contextbar.strikeOutText
- errors.invalidLicense
- stamptext.approved
- stamptext.notApproved
- stamptext.draft
- stamptext.final
- stamptext.completed
- stamptext.confidential
- stamptext.forPublic
- stamptext.notForPublic
- stamptext.void
- stamptext.forComment
- stamptext.preliminaryResults
- stamptext.informationOnly
- print.title
- print.all
- print.current
- print.range
- print.invalid
- print.print
- print.cancel
- border.none

5.3 Changes in Version 2

- **New** support for filling form fields:
  - text fields
  - check boxes
  - radio buttons
  - list boxes
  - drop down

  This feature can be enabled or disabled with the new option `WebViewerOptions.forms.enabled`.
- **New** support for pre-configuring a PDF whose pages are to be made available as stamp annotations.
- **New** viewing-only mode with disabled editing of annotations and form filling capability. This mode is entered automatically if the license does not support editing. Additionally, it can be enabled or disabled with the new option `WebViewerOptions.viewer.general.viewOnly`.
- **New** support for preventing the modification and deletion of all annotations except the currently configured author's. This restriction can be enabled or disabled with the new option `WebViewerOptions.annotation.onlyAuthorCanEdit`.
- **New** feature double-click selects word.
- **Improved** scrolling sensitivity for mobile devices.
- **Changed** side pane. Title texts are substituted by icons.
- **Changed** page mode selection user interface. Texts are substituted by icons.
- **New** translation for French.

6 Licensing, copyright, and contact

Pdftools is a world leader in PDF (Portable Document Format) software, delivering reliable PDF products to international customers in all market segments.

Pdftools provides server-based software products designed specifically for developers, integrators, consultants, customizing specialists and IT departments. Thousands of companies worldwide use our products directly and hundreds of thousands of users benefit from the technology indirectly via a global network of OEM partners. The tools can be easily embedded into application programs and are available for a multitude of operating system platforms.

Licensing and copyright The PDF Web Viewer is copyrighted. This user's manual is also copyright protected; It may be copied and given away provided that it remains unchanged including the copyright notice.

Contact

PDF Tools AG
Brown-Boveri-Strasse 5
8050 Zürich
Switzerland
http://www.pdf-tools.com
pdfsales@pdf-tools.com

A How to Migrate from 3-Heights®

This chapter explains step by step how to migrate from the previous product, the 3-Heights® Web Viewer, to the PDF Web Viewer.

A.1 Prerequisite

- Reference the PDF Web Viewer as an NPM package or download the ZIP archive `PdfWebViewer-4.3.0.zip`. (See also Installation and deployment.)
- Obtain a corresponding new license key. See License keys.

A.2 Upgrade

1. Replace the static assets of the 3-Heights® WebViewer, e.g. in case you use the ZIP archive:
   - Delete the old static assets of the `pdfwebviewer` directory of your 3-Heights® WebViewer.
   - Extract files from downloaded `PdfWebViewer-4.3.0.zip` archive.
   - Navigate to sub-directory `webapp/pdfwebviewer/` and copy its content to the `pdfwebviewer` directory of your 3-Heights® WebViewer.
2. Replace the license key in `index.html`.
3. Replace the base path definition

   ```
   window.PDFTOOLS_WEBVIEWER_BASEURL
   ```

   with

   ```
   window.PDFTOOLS_FOURHEIGHTS_PDFVIEWING_BASEURL
   ```

4. If necessary, adopt the following changes in the structure of the options. (If you want to use new options, please consult Interface reference.)
   - Renamed:
     - `author` → `viewer.general.user`
     - `promptOnUnsavedChanges` → `viewer.general.promptOnUnsavedChange`
     - `defaultFreetextBgColor` → `annotation.colors.backgroundColors`
     - `defaultFreetextFontColor` → `annotation.colors.foregroundColors`
     - `enableThumbnailNavigation` → `viewer.sidebar.thumbnailNavigation`
     - `enableOutlineNavigation` → `viewer.sidebar.outlineNavigation`
     - `enableAnnotationNavigation` → `viewer.sidebar.annotationNavigation`
     - `defaultBorderSize` → `annotation.defaultBorderWidth`
   - Moved:
     - `allowFileDrop` → `viewer.permissions.allowFileDrop`
     - `allowSaveFile` → `viewer.permissions.allowSaveFile`
     - `allowOpenFile` → `viewer.permissions.allowOpenFile`
     - `enableSearch` → `viewer.permissions.enableSearch`
     - `pageLayoutModes` → `viewer.general.pageLayoutModes`
     - `language` → `viewer.general.language`
     - `annotationBarPosition` → `viewer.general.annotationBarPosition`
     - `textSelectionColor` → `viewer.general.textSelectionColor`
     - `searchMatchColor` → `viewer.general.searchMatchColor`

- strokeWidths → annotation.strokeWidths
- highlightOpacity → annotation.highlightOpacity
- stamps → annotation.stamps
- defaultStampWidth → annotation.defaultStampWidth
- highlightColors → annotation.colors.highlightColors
- foregroundColors → annotation.colors.foregroundColors
- backgroundColors → annotation.colors.backgroundColors
- defaultHighlightAnnotationColor → annotation.colors.defaultHighlightAnnotationColor
- defaultInkColor → annotation.colors.defaultInkColor
- defaultStickyNoteColor → annotation.colors.defaultStickyNoteColor
- defaultBackgroundColor → annotation.colors.defaultBackgroundColor
- defaultForegroundColor → annotation.colors.defaultForegroundColor
- defaultHighlightColor → annotation.colors.defaultHighlightColor
- fontSizes → annotation.font.fontSizes
- fontFamilies → annotation.font.fontFamilies
- defaultFreetextFontFamily → annotation.font.defaultFreetextFontFamily
- defaultFreetextFontSize → annotation.font.defaultFreetextFontSize
- defaultFontFamily → annotation.font.defaultFontFamily
- defaultFontSize → annotation.font.defaultFontSize
- onOpenFileButtonClicked → viewer.callbacks.onOpenFileButtonClicked
- onSaveFileButtonClicked → viewer.callbacks.onSaveFileButtonClicked