Document Solutions PDF Viewer

日本語

Document Solutions PDF Viewer (**DsPdfViewer**) is a fast, modern JavaScript based PDF viewer and editor that runs in all major browsers. The viewer can be used as a cross platform solution to view (or modify - see *Support API* below) PDF documents on Windows, MAC, Linux, iOS and Android devices. DsPdfViewer is included in Document Solutions for PDF (DsPdf) - a feature-rich cross-platform PDF API library for .NET Core.

Client-side (Wasm) or server-side (DS.Documents.Pdf.ViewerSupportApi) *Support API* option allows you to turn DsPdfViewer into a powerful PDF editor that can be used to edit existing or create new PDFs, fill and save PDF forms, remove (redact) sensitive content, add or edit annotations and AcroForm fields, and more.

DsPdfViewer provides a rich JavaScript object model, see **docs/index.html** for the client API documentation.

Product highlights:

- Works in all modern browsers, including Edge, Chrome, FireFox, Opera, Safari
- Provides form filling and PDF editing tools (see *Support API*):
  - Customizable and mobile-friendly form filler
  - Annotation and form editors
  - Quick edits using secondary toolbars
  - PDF redaction
  - Signature verification
  - Real-time collaboration mode
  - Other editing features
- Works with frameworks such as React, Preact, Angular
- Supports form filling; filled forms can be printed or form data can be submitted to the server
- Supports XFA (XML Forms Architecture) forms
- Provides caret for text selection/copy, including vertical and RTL texts
- Includes thumbnails, text search, outline, attachments, articles, layers and structure tags panels
- Allows opening PDF files from local disks
- Supports annotations including text, free text, rich text etc.
- Supports redact annotations (including appearance streams), allows user to hide or show redacts
- Supports sound annotations
- Allows rotating and printing the rotated document
- Supports article threads navigation
- Supports file attachments
- Comes with several themes, new custom themes can be added
- Supports external CMaps
- ...and more.

See it in action

- Go to Document Solutions for PDF Viewer demos to explore the various features of DsPdfViewer, including features that rely on *Support API*. The demo site allows you to modify the demo code and immediately see the effect of your changes.
- The Document Solutions for PDF API demo site uses DsPdfViewer to display sample PDFs.

Important note

The @mescius/dspdfviewer package replaces @grapecity/gcpdfviewer, and provides the same functionality, ensures future enhancements, and is backwards compatible with @grapecity/gcpdfviewer. Existing licenses will continue to work with DsPdfViewer.

Latest changes

[8.1.1] - 09-May-2025

Fixed

- [Regression since v8.1.0] Sound annotations do not play when double-clicked. (DOC-6897)

[8.1.0] - 10-Apr-2025

Added

- [Editor] Table data extractor tool (requires server-side SupportApi): finds and extracts tabular data from a region selected by the user. The data can be copied to clipboard or saved to a file in different formats including CSV, TSV, JSON, XLSX, XML or HTML. (DOC-5185)
- [Editor] [Wasm] Added support for redact, export to PNG, converting annotations to content and PDF linearize (fast Web view) to client-side (Wasm) SupportApi. (DOC-6388)
- Added support for the PDF sound action. (DOC-6738)

Changed

- [Editor] Improved the 'Text' property editor (always show the value, allows for easier editing). (DOC-6740)

Fixed

- Text selection is not visible in Safari Browser on IOS 18. (DOC-6822)

[8.0.6] - 2-Apr-2025

Added

- Option to send PDFs containing pages with different orientations to the browser print preview as a single job. (DOC-6854)

```
// Enable single print job mode to force all pages to be printed together:
var viewer = new DsPdfViewer("#root", {
  printSingleJobMode: true
});
```

Fixed

- [Wasm] Position of annotations changes when saving the document using Wasm support API. (DOC-6848)

[8.0.5] - 20-Mar-2025

Added

- Added support for cancelling the onBeforeOpen event. (DOC-6755)

Fixed

- [Regression since v8.0.4] Loading PDFs is very slow in FireFox browser. (DOC-6814)
- Typings of modules i18next and moment are missing in DsPdfViewer installation. (DOC-6778)

- The viewer zoom factor changes when the layout mode is changed. (DOC-6554)
- 'TypeError: Cannot read properties of null' can occur in specific scenarios. (DOC-6846)

[8.0.4] - 26-Feb-2025

Changed

- [Reply Tool] Enabled adding replies and statuses to locked annotations. (DOC-6773)

Fixed

- [Reply Tool] Fixed a permission check issue when deleting a reply using the Delete key. (DOC-6773)
- TypeError when clicking on the page with ReplyTool enabled. (DOC-6753)
- DsPdfViewer container element does not scroll if there is no PDF in the viewer. (DOC-6766)
- [API Docs] Some code examples formatting is incorrect. (DOC-6768, DOC-6769, DOC-6772)
- [iOS 17] Specific PDF fails to load on iOS 17.5 (loads fine on iOS 18.0 and above). (DOC-6724)

[8.0.3] - 09-Feb-2025

Added

- [Reply Tool] Automatic focusing on the clicked annotation in the replies list. Now, when you click on an annotation, the corresponding entry in the replies list will be focused and brought into view.
- [Reply Tool] Added new Reply Tool event bus events. (DOC-6732)
  - replytool-before-delete: triggered before an annotation is deleted using the Reply Tool. This event is cancelable, allowing prevention of deletion.
  - replytool-before-scrollintoview: triggered before the Reply Tool scrolls to a reply item's HTML element. This event is cancelable, enabling control over the scrolling behavior.
  - replytool-text-label-click: triggered when the user clicks a text label, before executing the associated Reply Tool actions (e.g., displaying the inline text editor for editable annotations). This event is cancelable, allowing customization or prevention of the default behavior.
  - replytool-item-click: triggered when the user clicks a list item in the Reply Tool, either a parent note item or a reply item. The event can be canceled by setting the cancel property of the event arguments object to true.
  - replytool-item-keydown: triggered when the user presses a key while a list item in the Reply Tool is focused (either a parent note item or a reply item). The event can be canceled by setting the cancel property of the event arguments object to true, preventing the default Reply Tool behavior.
  - replytool-placeholder-activate: triggered when the user activates the reply placeholder by clicking on it or pressing any key while it is focused. The event can be canceled by setting the cancel property of the event arguments object to true, preventing the default Reply Tool behavior (e.g., opening the reply editor).
  - replytool-author-label-click: triggered when the user clicks the author label, before executing the associated Reply Tool actions (e.g., opening the inline text editor for editable annotations). The event can be canceled to prevent the default behavior.
  - replytool-item-status-change: triggered when the user changes the status of a reply annotation. It can be canceled by setting args.cancel to true. The event also allows overriding the target annotation using args.annotation or modifying the status via args.status. Possible statuses: 'None', 'Accepted', 'Cancelled', 'Completed', 'Rejected'.
  - replytool-post-reply: triggered when the user adds a reply annotation via the "Post Reply" button or by pressing Ctrl+Enter after editing the reply text. The event can be canceled by setting args.cancel to true and allows modifying the reply annotation using args.annotation.

```
// Example:
viewer.eventBus.on("replytool-before-delete", function(args) {
    // Check if the entire annotation is being deleted (not just a comment)
    if (!args.removeCommentOnly) {
```

```
        // Confirm deletion with the user
        if (!confirm("Are you sure you want to delete the annotation with ID " + args.annotation.id +
"?")) {
            // Cancel the deletion if the user declines
            args.cancel = true;
        }
    }
});
```

- Added annotationFilter option for Reply Tool. Allows customizing which annotations are displayed in the Reply Tool by providing a filter function.

```
// Example: show only text annotations in the Reply Tool
var viewer = new DsPdfViewer("#root", {
    replyTool: {
        annotationFilter: (annotation) => annotation.annotationType === 1 // AnnotationType.TEXT
    }
});
```

Changed

- JPN localization updated. (DOC-6727)
- [Reply Tool] Improved behavior for annotations without comments. (DOC-6718)
- The "Add a comment..." placeholder is now hidden unless the annotation is selected or focused.

Fixed

- Incorrect document display at high zoom values. (DOC-6728)
- [Reply Tool] Incorrect position of timestamp for the reply in the Reply Tool panel. (DOC-6716)
- [Reply Tool] The comment popup behaves incorrectly if the Reply Tool and other toolbars are used together. (DOC-6729)
- [Reply Tool] Issues with styling in Dark and Dark-Yellow Themes. (DOC-6737)
- [Reply Tool] Inconsistent behavior when deleting comments for annotations other than sticky notes. (DOC-6735)
- [Layers] Incorrect display of PDF layers in some cases. (DOC-6733)

[8.0.1] - 25-Dec-2024

Added

- Added text selection capability to FreeText annotations. (DOC-6659)

Fixed

- Proximity search like [“My favorite” ONear(1) "is" ONear(5)"."] does not work as expected. (DOC-6579)
- Open search panel stops working when another document is picked from the document list. (DOC-6666)
- The same text can be replaced several times. (DOC-6674)

[8.0.0] - 03-Dec-2024

Added

- Added text replace feature: use Ctrl+H to open the search bar in replace mode, or click the expand button in the search bar to toggle between search and replace modes. (DOC-5200)
- Added display of timestamps in user comments. (DOC-6233)
- [Reply Tool] Added useRelativeDates and dateFormat options. (DOC-6586)

```
// Example 1: Disable relative dates and use absolute dates instead.
var viewer = new DsPdfViewer("#root", { replyTool: { useRelativeDates: false } });
// Example 2: Use a custom format for absolute dates.
```

```
var viewer = new DsPdfViewer("#root", { replyTool: { useRelativeDates: false, dateFormat: "yyyy.mm.dd HH:MM" } });
```

- Added hidePopupTimestamp option. (DOC-6587)

Changed

- Enhanced proximity search with support for NEAR and ONEAR operators, added the ability to quote phrases for "full phrase" proximity search. (DOC-6390)

Fixed

- [CommentTimeStamp] Incorrect handling of future dates in relative date formatting. (DOC-6588)

See `CHANGELOG.md` for previous release notes.

Installation

To install the latest release version:

```
npm install @mescius/dspdfviewer
```

To install from the zip archive:

Go to https://developer.mescius.com/document-solutions/dot-net-pdf-api/download and follow the directions on that page to get your 30-day evaluation and deployment license key. The license key will allow you to develop and deploy your application to a test server. Unzip the viewer distribution files (list below) to an appropriate location accessible from the web page where the viewer will live.

Viewer zip includes the following files:

- `README.md`: this file
- `CHANGELOG.md`
- `docs.html`: documentation
- `dspdfviewer.js`
- `dspdfviewer.worker.js`
- `wasmSupportApi.js`
- `wasmSupportApiServer.js`
- `DsPdf.js`
- `DsPdf.wasm`
- `package.json`
- `index.html`: example page, no SupportApi
- `index-wasm.html`: example page with Wasm SupportApi, can be opened from a desktop file manager
- `index-wasm-server.html`: example page with Wasm SupportApi, requires a web server, see run_wasm.cmd
- `run_wasm.cmd`: example script to open index-wasm-server.html
- `docs/`: documentation files
- `localization/`: localization example
- `resources/`: various resources
- `themes/`: theme files
- `typings/`: TypeScript declarations

Documentation

Online documentation is available here.

Licensing

Document Solutions PDF Viewer is a commercially licensed product. Please visit this page for details.

Getting more help

Document Solutions PDF Viewer is distributed as part of Document Solutions for PDF. You can ask any questions about the viewer, or report bugs using the Document Solutions for PDF public forum.

More details on using the viewer

Adding the viewer to an HTML page:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <!-- Limit content scaling to ensure that the viewer zooms correctly on mobile devices: -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-
scale=1.0, user-scalable=no" />
    <meta name="theme-color" content="#000000" />
    <title>Document Solutions PDF Viewer</title>
    <script type="text/javascript" src="lib/dspdfviewer.js"></script>
    <script>
        function loadPdfViewer(selector) {
            var viewer = new DsPdfViewer(selector,
              {
                /* Specify options here */
                renderInteractiveForms: true
              });
            // Skip the 'report list' panel:
            // viewer.addReportListPanel();
            viewer.addDefaultPanels();
            // Optionally, open a PDF (will only work if this runs from a server):
            viewer.open('HelloWorld.pdf');
            // Change default viewer toolbar:
            viewer.toolbarLayout.viewer.default = ['$navigation', '$split', 'text-selection', 'pan', '$zoom',
'$fullscreen',
              'save', 'print', 'rotate', 'view-mode', 'doc-title'];
            viewer.applyToolbarLayout();
        }
    </script>
  </head>
  <body onload="loadPdfViewer('#root')">
    <div id="root"></div>
  </body>
</html>
```

How to license the viewer:

Set the DsPdfViewer Deployment key to the DsPdfViewer.License property before you create and initialize DsPdfViewer. This must precede the code that references the js files.

```
  // Add your license
  DsPdfViewer.LicenseKey = 'xxx';
  // Add your code
  const viewer = new DsPdfViewer("#viewer1", { file: 'helloworld.pdf' });
  viewer.addDefaultPanels();
```

Support API

Support API is our PDF processing engine that enables the PDF editing features of DsPdfViewer. Two options are available when configuring the viewer with *Support API*:

- **Server-based Support API**: requires DS.Documents.Pdf.ViewerSupportApi NuGet package which needs to run on a .NET server. The viewer connects to it via the supportApi property and uses that connection to perform any edits. To set up a Support API server on your system and see it in action, download any of the samples from the DsPdfViewer demo site (e.g. Edit PDF), and follow the instructions in the `readme.md` included in the downloaded zip. The full C# source code of ViewerSupportApi together with demo projects for .NET 8 and Web Forms is included in the `src/` folder inside the NuGet package (the **WEB_FORMS** constant is defined for the Web Forms target). In your server solution you can either reference the package, or include the source project and reference it instead.
- **Client-based Support API**: uses the `DsPdf.wasm` WebAssembly binary that runs in the client browser. This configuration does not require a server connection, all editing is done on the client. You also do not need to download any additional components as the Wasm binaries are included in this package.
  To try the Wasm option, you can simply double click the `index-wasm.html` file (included in this package) in your file manager. Also included is `run_wasm.cmd` that uses http-server to load `index-wasm-server.html`, showing a more flexible approach to using the Wasm option, see comments in `index-wasm-server.html` for details.
  Due to its client-based nature, collaboration features are not available in the Wasm configuration. Also, in this release (v7.2) the following features are not yet supported, and will be disabled in the viewer UI if the Wasm configuration is used:
  - Adding/applying redacts
  - Electronic signatures
  - Export to raster or SVG images
  - Converting annotations to content

  If you are using IIS, you will need to explicitly allow IIS to load .wasm files in your `web.config` in order to use `DsPdf.wasm`:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<system.webServer>
  <staticContent>
    <remove fileExtension=".wasm" />
    <mimeMap fileExtension=".wasm" mimeType="application/wasm" />
  </staticContent>
  <httpCompression>
    <dynamicTypes>
      <add mimeType="application/wasm" enabled="true" />
    </dynamicTypes>
  </httpCompression>
</system.webServer>
</configuration>
```

Note that you need a Document Solutions for PDF Professional license in order to use *Support API* in your apps.

Keyboard shortcuts

Viewer mode

- `Ctrl +` - zoom in
- `Ctrl -` - zoom out
- `Ctrl 0` - zoom to 100%
- `Ctrl 9` - zoom to window
- `Ctrl A` - select all
- `R` - rotate clockwise

- `Shift R` - rotate counterclockwise
- `H` - enable pan tool
- `S` - enable selection tool
- `V` - enable selection tool
- `Ctrl O` - open local PDF
- `Ctrl F` - text search
- `Ctrl P` - print
- `Home` - go to start of line
- `Ctrl Home` - go to start of document
- `Shift Home` - select to start of line
- `Shift Ctrl Home` - select to start of document
- `End` - go to end of line
- `Ctrl End` - go to end of document
- `Shift End` - select to end of line
- `Shift Ctrl End` - select to end of document
- `Left` - go left
- `Shift Left` - select left
- `Alt Left` - go back in history
- `Right` - go right
- `Shift Right` - select right
- `Alt Right` - go forward in history
- `Up` - go up
- `Shift Up` - select up
- `Down` - go down
- `Shift Down` - select down
- `PgUp` - page up
- `PgDown` - page down
- `Shift PgUp` - select page up
- `Shift PgDown` - select page down

Editing modes

- `Delete` - delete selected annotation/field
- `Esc` - unselect annotation/field
- `Ctrl Z` - undo
- `Ctrl Y` - redo
- `Ctrl Shift Z` - redo

Toolbar items

The default viewer toolbar items layout (items starting with '$' are inherited from the base viewer, other items are PDF viewer specific.):

```
['open', '$navigation', '$split', 'text-selection', 'pan', '$zoom', '$fullscreen', 'rotate', 'view-mode',
'theme-change', 'download', 'print', 'save', 'hide-annotations', 'doc-title', 'doc-properties', 'about']
```

The full list of the PDF-viewer specific toolbar items:

```
'text-selection', 'pan', 'open', 'save', 'download', 'print', 'rotate', 'theme-change', 'doc-title', 'view-
mode', 'single-page', 'continuous-view'
```

The full list of base viewer toolbar items:

```
'$navigation' '$refresh', '$history', '$mousemode', '$zoom', '$fullscreen', '$split'
```

You can get default base viewer items by using the getDefaultToolbarItems() method, e.g.:

```
const toolbar: Toolbar = viewer.toolbar;
let buttons = toolbar.getDefaultToolbarItems();
buttons = buttons.filter(b => b !== '$refresh');
```

To specify a custom set of toolbar items, use the toolbarLayout property and applyToolbarLayout() method, e.g.:

```
viewer.toolbarLayout.viewer = {
  default: ["$navigation", 'open', '$split', 'doc-title'],
  fullscreen: ['$fullscreen', '$navigation'],
  mobile: ["$navigation", 'doc-title']
};
viewer.toolbarLayout.annotationEditor = {
  default: ['edit-select', 'save', '$split', 'edit-text'],
  fullscreen: ['$fullscreen', 'edit-select', 'save', '$split', 'edit-text'],
  mobile: ['$navigation']
};
viewer.toolbarLayout.formEditor = {
  default: ['edit-select-field', 'save', '$split', 'edit-widget-tx-field', 'edit-widget-tx-password'],
  fullscreen: ['$fullscreen', 'edit-select-field', 'save', '$split', 'edit-widget-tx-field', 'edit-widget-tx-
password'],
  mobile: ['$navigation']
};
viewer.applyToolbarLayout();
```

Here is an example of how to create your own custom toolbar button:

```
const toolbar: Toolbar = viewer.toolbar;
toolbar.addItem({
  key: 'my-custom-button',
  iconCssClass: 'mdi mdi-bike',
  title: 'Custom button',
  enabled: false,
  action: () => {
    alert("Custom toolbar button clicked");
  },
  onUpdate: (args) => ({ enabled: viewer.hasDocument }),
});
viewer.toolbarLayout.viewer.default =  ['$navigation', 'my-custom-button'];
viewer.applyToolbarLayout();
```

Using the viewer in frameworks such as React, Preact, Angular etc.

Add a reference to the viewer script:

```
<body>
  <script type="text/javascript" src="/lib/dspdfviewer/dspdfviewer.js"></script>
  ...
```

Add the placeholder to your App template, e.g.:

```
<section id="pdf"></section>
```

Render the viewer:

```
let viewer = new DsPdfViewer('section#pdf');
viewer.addDefaultPanels();
```

DioDocs PDFビューワ

DioDocs PDFビューワは、Webアプリケーションのクライアントサイドで PDFファイルを読み込んで表示する、JavaScriptベースのPDFビューワです。また、バックエンドのAPIを使って、PDFファイルを編集することもできます。

サンプル

使い方については、下記をご参照ください。

- デモ

インストール方法

```
npm install @mescius/dspdfviewer
```

日本語版での動作保証は、日本語版サイトで公開しているバージョンのみとなります。

ドキュメント

ドキュメントについては、下記をご参照ください。

- ヘルプ
- APIリファレンス

製品情報

価格、ライセンスについては、下記をご参照ください。

- 製品情報

サポート

ナレッジベース、テクニカルサポートについては、下記をご参照ください。

- サポート＆サービス
- サブスクリプションサービス利用規約

The End.