# Document Solutions PDF Viewer

A full-featured JavaScript PDF viewer and editor that comes with [Document Solutions for PDF](https://developer.mescius.com/document-solutions/dot-net-pdf-api).

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)](https://developer.mescius.com/document-solutions/dot-net-pdf-api) - a feature-rich cross-platform PDF API library for .NET Core.

[Support API](#support_api) is a server-side NuGet package [DS.Documents.Pdf.ViewerSupportApi](https://www.nuget.org/packages/DS.Documents.Pdf.ViewerSupportApi/) that allows you to easily set up an ASP.​NET Core or a Web Forms server that employs DsPdf to add PDF editing abilities to DsPdfViewer. When connected to a Support API server, DsPdfViewer becomes a powerful PDF editor that can be used to edit existing or create new PDF documents, fill and save PDF forms, remove (redact) sensitive content, add or edit annotations and AcroForm fields, and more.

DsPdfViewer provides a rich client side 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
* When connected to DsPdf on the server via Support API, provides:
  + Customizable and mobile-friendly form filler
  + Real-time collaboration mode
  + Annotation and form editors
  + Quick edits using secondary toolbars
  + PDF redaction
  + Signature verification
  + 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
* Fully supports file attachments (both attachment annotations and document level 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](https://developer.mescius.com/document-solutions/javascript-pdf-viewer/demos/) to explore the various features of DsPdfViewer, including features that rely on [Support API](#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](https://developer.mescius.com/document-solutions/dot-net-pdf-api/demos/) uses DsPdfViewer to display sample PDFs.

## Latest changes

## Important note

This is the initial release of the @mescius/dspdfviewer package. It 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.

## [7.0.0] - 07-Dec-2023

### Added

* Added DsPdfViewer class and @mescius/dspdfviewer package. Functionally DsPdfViewer is identical to GcPdfViewer. (DOC-5735)
* Added floating text search bar. (DOC-5406)
* Added localization resources and a localization example to the pdf viewer build.
* Added new option useFloatingSearchBar - enable a floating search bar instead of the sidebar search panel. Default value is true.

// To revert to using sidebar search panel instead of the floating search bar:  
var viewer = new DsPdfViewer("#root", { useFloatingSearchBar: false });

### Changed

* Some tooltips updated.

### 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
* dspdfviewer.js
* dspdfviewer.worker.js
* package.json
* index.html (sample page)
* Theme files:
  + themes/dark-yellow.css
  + themes/dark-yellow.jscss
  + themes/light-blue.css
  + themes/light-blue.jscss
  + themes/viewer.css
  + themes/viewer.jscss
* Predefined CMap files for Chinese, Japanese, or Korean text output:
  + resource/bcmaps/\*.bin
* TypeScript declaration files:
  + typings/.

## Documentation

Online documentation is available [here](https://developer.mescius.com/document-solutions/dot-net-pdf-api/docs/online/pdfviewer.html).

## Licensing

Document Solutions PDF Viewer is a commercially licensed product. Please [visit this page](https://developer.mescius.com/document-solutions/licensing) 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](https://developer.mescius.com/forums/documents-pdf).

## 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 a server-side library available as NuGet package [DS.Documents.Pdf.ViewerSupportApi](https://www.nuget.org/packages/DS.Documents.Pdf.ViewerSupportApi/). The full source code of this library together with C# demo projects for ASP.​NET Core and Web Forms is included in the src/ folder inside the 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.

When DsPdfViewer is connected to a Support API server, its editing features are enabled. The edits done on the client are accumulated, and when the user clicks 'save', the original PDF and the edits are sent to the server, the edits are applied (using DsPdf), and the modified PDF is sent back to the client.

To set up a Support API server on your own system and see it in action, download any of the samples from the [DsPdfViewer demo site](https://developer.mescius.com/document-solutions/javascript-pdf-viewer/demos/) (e.g. [Edit PDF](https://developer.mescius.com/document-solutions/javascript-pdf-viewer/demos/edit-pdf/purejs)), and follow the instructions in the readme.​md included in the downloaded zip.

NOTE that you will need a Document Solutions for PDF Professional license 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();

The End.