1 Overview

ComPDFKit Web Viewer is a robust online PDF viewer library for developers who need to develop applications on the Web, which offers powerful JavaScript APIs for quickly viewing and editing any PDF online. It is feature-rich and battle-tested, making PDF files process and manipulation easier and faster for your online project.

1.1 Key Features

Viewer component offers:

- Standard page display modes, including Scrolling, Double page, and Cover mode.
- Navigation with thumbnails, outlines, and layers.
- Text search & selection.
- Zoom in and out.
- Switch between different themes, including light mode and dark mode
- Rotate right or left.

Annotations component offers:

- Create, edit, and remove kinds of annotations, including Notes, Free Text, Lines, Squares, Circles, Ink, and Stamps.
- Support for annotation appearances.
- Import and export annotations to/from XFDF.

Forms component offers:

- Create, edit and, remove form fields, including Text Fields, Check Boxes, and Signatures.
- Fill out PDF Forms.

Signatures component offers:

- Drawn signatures
- Image signatures
- Typed signatures

1.2 License

ComPDFKit Web Viewer is a commercial SDK, which requires a license to grant developer permission to release their apps. Each license is only valid for one root domain. Other flexible licensing options are also supported, please contact our marketing team to know more. However, any documents, sample code, or source code distribution from the released package of ComPDFKit to any third party is prohibited.

2 Get Started

It is easy to embed ComPDFKit in your web app with a few lines of JavaScript code. Take just a few minutes and get started.

The following sections introduce the structure of the installation package, how to run a demo, and how to make a web app with ComPDFKit Web Viewer.

2.1 Requirements

To integrate ComPDFKit Web Viewer in your browser, you must have a development environment and a browser.

- The latest stable version of Node.js.
- ComPDFKit Web Viewer supports most mainstream browsers, and it's better to use the latest version. IE browser is not supported currently.

2.2 Package Structure

The package of ComPDFKit Web Viewer includes the following files.

- ***"ComPDFKit-Web-Viewer-Demo"*** - A folder containing Web sample projects.
- ***"Lib"*** - Include ComPDFKit Web Viewer library.
- ***"ComPDFKit Web Viewer.md"*** - Developer guide and API reference.
- ***"Core&UI.txt"*** - Third-party code usage agreement.
- ***"Legal.txt"*** - Legal and copyright information.
- ***"Release_Note.txt"*** - Release information.

2.3 How to run a demo

ComPDFKit Web Viewer provides one demo for developers to learn how to call the SDK on the Web. You can find them in the ***ComPDFKit-Web-Viewer-Demo*** folder. This guide will show you how to run it in **VSCode**.

1. Open the **Demo** project in **VSCode**.
2. Install all dependencies that are needed in the **Demo**.

```
npm install
```

3. Run **ComPDFKit Web Viewer Demo** in the development environment

```
npm run dev
```

4. PDF will be opened and displayed.

2.4 How to Make a Web App in JavaScript With ComPDFKit Web Viewer

This section will help you to quickly get started with ComPDFKit Web Viewer to make a Web app with step-by-step instructions. Through the following steps, you will get a simple web application that can display the contents of a specified PDF file.

Before you start the following steps, create a new web project first.

2.4.1 Add ComPDFKit Web Viewer Package

1. Add the **@compdfkit** folder in the **lib** directory to the **root** directory or **assets** directory of your project. This will serve as the entry point for the ComPDFKit Web Viewer and will be imported into your project.
2. Add the **webviewer** folder that contains the required static resource files to run the ComPDFKit Web Viewer demo, and add the ***example*** folder which contains sample PDF files, to your project's **static resource** folder.

2.4.2 Display a PDF Document

1. Import the "***webviewer.js***" file in the **@compdfkit** folder into your project.
2. Initialize the ComPDFKit Web Viewer in your project by calling `ComPDFKitViewer.init()`.
3. Pass the PDF address you want to display and your license key into the `init` function.

```
// Import the JS file of ComPDFKit Web Viewer
import ComPDFKitViewer from "/@compdfkit/webviewer";

const viewer = document.getElementById('webviewer');
ComPDFKitViewer.init({
  pdfUrl: 'Your PDF Url',
  license: 'Input your license here'
}, viewer)
.then((core) => {
  const docViewer = core.docViewer;
  console.log('ComPDFKit Web Viewer loaded');
})
```

Note: *You need to contact ComPDFKit team to get the **license**.*

4. Once your project is running, you will be able to see the PDF file you want to display.

3 Guides

If you're interested in all of the ComPDFKit features, please go through our guides to quickly add PDF viewing, annotating, editing, form filling, and signing to your application. The following sections list some examples to show you how to add document functionalities to your web apps using our JavaScript APIs.

3.1 Viewer

The viewer provides a fast and battle-tested rendering engine with a wide range of advanced features including display modes, content selection, zoom level settings, and page navigation, which offers developers a way to quickly embed a highly configurable PDF viewer in web apps.

3.1.1 Display Modes

ComPDFKit Web Viewer provides viewing PDFs in different display modes. Let's see what the display modes are and call the following method to display in different display modes. The corresponding parameters show in the following table.

- Single Mode

  Show one page at a time for users to continuously scroll up and down to navigate through the pages.

- Two-up Mode

  Display two pages side-by-side and continuously scroll up and down to navigate through the pages.

- Cover Mode

  Show the cover page on the first page during two-up.

```
docViewer.webViewerSwitchSpreadMode(mode)
```

Name	Required	Type	Description
mode	yes	number	Display Modes Parameter, Single Mode: 0, Two-up Mode: 1, Cover Mode: 2

- Full Screen Mode

  It's a mode to view PDFs in full-screen, which fills the entire screen with the PDF content. The following method is given to start the full-screen presentation mode.

  ```
  docViewer.requestFullScreenMode()
  ```

3.1.2 PDF Navigation

Provide thumbnails, bookmarks, and layers to navigate PDF content.

- Page Navigation

  After loading a PDF document, you can programmatically interact with it, which allows you to view and transition between the pages like scrolling and jumping to specific pages.

```
// Previous page
docViewer.previousPage()

// Next page
docViewer.nextPage()

// Jump to a page
docViewer.pageNumberChanged()
```

- Scroll Mode

  ComPDFKit Web Viewer has two scrolling modes: Vertical and Horizontal. You can set the corresponding mode by using the following method.

```
docViewer.webViewerSwitchScrollMode(mode)
```

This is the parameters to apply different scroll modes.

Name	Required	Type	Description
mode	yes	number	Scroll Mode Parameter, Vertical: 0, Horizontal: 1

- Thumbnails

  Provide methods to render PDF pages as thumbnail images

3.1.3 Text Search & Selection

ComPDFKit Web Viewer offers developers an API for programmatic full-text search, as well as UI for searching and highlighting relevant matches. You can use the method below to search for content in PDFs with ComPDFKit Web Viewer.

```
docViewer.search(value)
```

Name	Required	Type	Description
value	yes	string	The Searched Content

3.1.4 Zooming

ComPDFKit Web Viewer provides super zoom out and in to unlock more zoom levels, and pinch-to-zoom or double tap on the specific area to perform a smart page content analysis, or you can programmatically interact with it by using the following method.

Use the following methods to zoom in/out by a certain zoom value or scale.

```
// Zoom in
docViewer.zoomIn()

// Zoom out
docViewer.zoomOut()ComPDFKit Web Viewer provides super zoom out and in to unlock more zoom levels, and pinch-to-zoom or double tap on the specific area to perform a smart page content analysis, or you can programmatically interact with it by using the following method.

Use the following methods to zoom in/out by a certain zoom value or scale.

// Scale changed
docViewer.webViewerScaleChanged(scale)
```

Name	Required	Type	Description
scale	yes	number	Zoom Scale, Zoom In: 10, Zoom Out: 0.5

3.2 Annotations

In addition to its primary textual content, a PDF file can contain annotations that represent links, form elements, highlighting circles, textual notes, and so on. Each annotation is associated with a specific location on a page and may offer interactivity with the user. Annotations allow users to mark up and comment on PDFs without altering the original author's content.

ComPDFKit Web Viewer supports most annotation types defined in PDF Reference and provides APIs for annotation creation & Editing, importing/exporting, flattening, properties access and modification, appearance setting, and drawing.

3.2.1 Annotation Types

ComPDFKit Web Viewer supports annotation types below. All annotations we supported are standard annotations (as defined in the PDF Reference) that can be read and written by many apps, such as Adobe Acrobat.

- Note
- Free text
- Shapes: Square, circle, line, and arrow
- Markup: Highlight, underline, strikeout, and squiggly
- Ink
- Stamp: Nearly 20 standard stamps and dynamic stamps

3.2.2 Create & Edit Annotations

ComPDFKit Web Viewer includes a wide variety of standard annotations, and each of them is added to the project in a similar way. Before creating annotations, you need to initialize a pdf document in your project.

Note: *When adding an annotation, the coordinate origin is at the bottom left corner of the page.*

```
// Import the JS file of ComPDFKit Web Viewer
import ComPDFKitViewer from "/@compdfkit/webviewer";

const viewer = document.getElementById('webviewer');
ComPDFKitViewer.init({
  pdfUrl: 'Your PDF Url',
  license: 'Input your license here'
}, viewer)
.then((core) => {
  const docViewer = core.docViewer;
  console.log('ComPDFKit Web Viewer loaded');
})
```

- Note

  Add a sticky note (text annotation) to a PDF Document page by using the following method.

```
docViewer.addAnnotations({
  type: 'text',
  page: 1,
  textColor: '#FF0000',
  fontSize: 16,
  color: '#FF0000',
  fontName: 'Helvetica',
  transparency: 1,
  fillTransparency: 0,
  content: 'test',
  rect: '0,0,50,50',
})
```

- Free Text

  Add a free text annotation to a PDF Document page by using the following method.

```
docViewer.addAnnotations({
  type: 'freetext',
  page: 1,
  textColor: '#000000',
  fontSize: 16,
  color: '#FF0000',
  fontName: 'Helvetica',
  transparency: 1,
  fillTransparency: 0,
  content: 'test',
  rect: '0,0,50,50',
  alignment: 0,
  content: 'test',
  rect: '0,0,50,50'
})
```

- Shapes

  Add a shape annotation like a rectangle, circle, line, or arrow to a PDF document page by using the following method.

```
//Square
docViewer.addAnnotations({
  type: 'square',
  page: 1,
  lineWidth: 2,
  transparency: 0.8,
  borderColor: '#FF0000',
  rect: '0,0,100,50'
})

// Circle
docViewer.addAnnotations({
  type: 'circle',
  page: 1,
  lineWidth: 2,
  transparency: 0.8,
  borderColor: '#FF0000',
  rect: '0,0,100,50'
})
```

```
// Line
docViewer.addAnnotations({
  type: 'line',
  page: 1,
  lineWidth: 2,
  transparency: 0.8,
  borderColor: '#FF0000',
  linePoints: '0,0,100,50'
})
```

- Markup

  Add a highlight annotation to a PDF Document page by using the following method, and add other markup annotations in a similar way.

  Note: *The value of* `quadPoints` *are (left,top), (right,top), (left,bottom), and (right,bottom). The coordinate origin is at the bottom left corner of the page.*

```
  docViewer.addAnnotations({
    type: 'highlight',  // highlight, underline, strikeout, squiggly
    page: 1,
    transparency: 0.8,
    quadPoints: "716,385,959,385,716,303,959,303",
    rect: "716,303,959,385",
    borderColor: "#FF0000",
    content: "test",
  })
```

- Ink

  Ink is the annotation to draw freely on PDFs with kinds of colors. Follow the method below to obtain our ink annotation.

```
docViewer.addAnnotations({
  type: 'line',
  page: 1,
  lineWidth: 2,
  transparency: 0.8,
  borderColor: '#FF0000',
  inklist:
'9.20,34.00,9.20,34.00//9.20,33.00,9.20,33.00//9.20,31.00,9.20,31.00//10.20,
31.00,10.20,31.00//11.20,30.00,11.20,30.00//12.20,28.00,12.20,28.00//13.20,2
7.00,13.20,27.00//13.20,26.00,13.20,26.00//15.20,24.00,15.20,24.00//17.20,23
.00,17.20,23.00//20.20,21.00,20.20,21.00//22.20,19.00,22.20,19.00//25.20,18.
00,25.20,18.00//29.20,18.00,29.20,18.00//33.20,18.00,33.20,18.00//34.20,18.0
0,34.20,18.00//36.20,18.00',
  rect: '9.20,18,36.20,34'
})
```

- Stamp

  Add standard and text stamps to a PDF document page by using the following method. Provide more than 20 standard stamps and dynamic time stamps.

```
// standard
docViewer.addAnnotations({
```

```
  type: "stamp",
  page: 1,
  rect: "205,379,435,431",
  stampType: "standard",
  standardStampType: "NotApproved"
})

// text
docViewer.addAnnotations({
  type: "stamp",
  page: 1,
  rect: "625,387,815,423",
  stampType: "text",
  textStampFirststring: "REVISED",
  textStampSecondstring: "28/07/2023 07:28:28",
  textStampColor: "1",
  textStampShape: "0"
})
```

3.3 Forms

A PDF document may contain any number of form fields that allow a user to enter information on a PDF page. An interactive form (sometimes referred to as an AcroForm) is a collection of fields for gathering information interactively from the user. Under the hood, PDF form fields are a type of PDF annotation called widget annotations.

ComPDFKit Web Viewer is a JavaScript library that fully supports reading, filling, creating, and editing PDF forms and provides utility methods to make working with forms simple and efficient.

3.3.1 Supported Form Fields

ComPDFKit Web Viewer now supports the form fields below by the PDF specification.

- Text Fields: It is a box or space for text fill-in data typically entered from a keyboard. The text may be set to a single line or multiple lines.
- Check Boxes: Check boxes could represent one or more checkboxes that can be checked or unchecked.
- Signatures: Allow users to add electronic signatures to PDF documents.

More supported form fields are coming soon. Please contact us if you have other form field requirements.

3.3.2 Create & Edit Form Fields

Creating form fields works the same as adding any other annotation, as can be seen in the guides for programmatically creating annotations. But you need to change the annotation type to the corresponding form fields, such as `text-field` or `check-box`.

- Text Fields

Text fields could be created, customized, named, filled, downloaded, hidden, and deleted. Except for the field, ComPDFKit Web Viewer provides options to change the text color, background color, font, font size, single/multiple lines, and alignment of the text in the text field. Here is the sample code below to set edit a text field.

```
docViewer.addAnnotations({
  type: "text-field",
  format: "0",
  rect: "721.40,538,863.40,626",
  fieldName: "Text Field1",
  isHidden: "0",
  backgroundColor: "#93B9FD",
  textColor: "#000000",
  fontName: "Helvetica",
  fontSize: "14",
  alignment: "0",
  multiline: false,
  page: 1
})
```

- Check Boxes

  Checkboxes could be created, customized, named, filled, downloaded, hidden, and deleted. Except for the field, ComPDFKit Web Viewer provides options to set the shape of the marker that appears inside the check box including check, circle, cross, diamond, square, or star. Here is the sample code below to set edit a check box.

```
docViewer.addAnnotations({
  type: "check-box",
  rect: "625.40,392,651.40,419",
  fieldName: "Check Box1",
  isHidden: "0",
  borderColor: "#43474D",
  backgroundColor: "#93B9FD",
  lineWidth: "3",
  style: "0",
  select: "0",
  page: 1
})
```

3.3.3 Fill Form Fields

ComPDFKit Web Viewer supports the AcroForm standard and provides a simple UI to fill out each form element. Click the corresponding form field and then fill in text or select options using either the onscreen keyboard or an attached hardware keyboard. Then click any blank area on the page to deselect the form element, which will commit the changes.

3.4 Signatures

It's a legal way to get consent or approval on electronic documents or forms. With Signatures, you can sign any PDF digital document conveniently and reliably without printing. Various types of signatures are supported.

3.4.1 Electronic Signatures

Electronic signatures could be seen as the evidence of the document signatory's agreement, and let you easily and securely sign documents with your signature. You can choose from three different ways to create your signature: draw it with a trackpad or mouse, type it with your preferred font style, or upload a photo of your handwritten signature. You can also customize the color and stroke width of your signature to make it look more authentic.

- Drawn

Drawn signatures let you create your signature with a trackpad or mouse. You can draw your signature as if you were using a pen or a pencil, and adjust the color and stroke width as you like. Drawn signatures are ideal for creating natural and personal signatures that match your handwriting style.

- Image

Image signatures let you upload a photo of your handwritten signature and use it to sign PDF documents. Image signatures are perfect for using your existing signature without having to redraw it every time.

- Typed

Typed signatures let you type your name and choose a font style that suits you. You can select from a variety of fonts that mimic handwriting, and change the color and stroke width as well. Typed signatures are convenient and fast for creating simple and elegant signatures that look professional.

4 Support

4.1 Reporting Problems

Thank you for your interest in ComPDFKit Web Viewer, the only easy-to-use but powerful development solution to integrate high quality PDF rendering and editing capabilities to your web app. If you encounter any technical questions or bug issues when using ComPDFKit Web Viewer, please submit the problem report to the ComPDFKit team. More information as follows would help us to solve your problem:

- ComPDFKit Web Viewer version.
- The Framework you used in your project.
- Detailed descriptions of the problem.
- Any other related information, such as an error screenshot.

4.2 Contact Information

Home Link:
https://www.compdf.com

Support & General Contact:
Email: support@compdf.com

Thanks,
The ComPDFKit Team