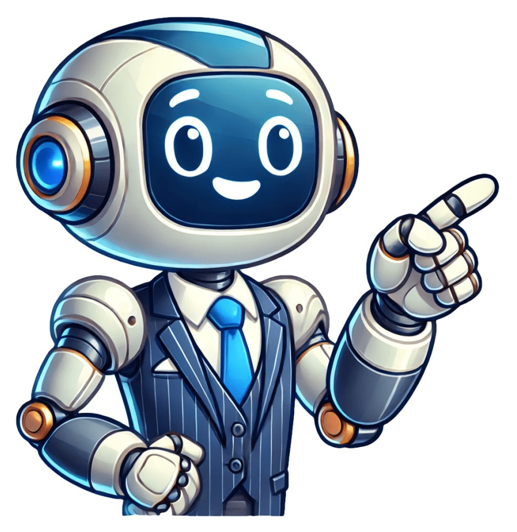
Click to prove
you're human



Live Demo Download Source Code# Yarn $ yarn add @tato30/vue-pdf # NPM $ npm i @tato30/vue-pdfAn easy PDF viewer component that makes it easier to embed PDF files into your Vue 3 applications.How to use it:1. Import the PDF viewer.import {usePDF, VuePDF} from 'VuePDF'export default { components: { VuePDF }, // ... }2. Add a PDF to your app. export default { components: { VuePDF }, setup(){ const { pdf, pages, info } = usePDF("myPDF.pdf") console.log(`Document has ${pages} pages`) console.log(`Document info: ${info}`) return { pdf } } }3. Available props to customize the PDF viewer.pdf?: PDFDocumentLoadingTask page?: number scale?: number rotation?: number fitParent?: boolean width?: number height?: number textLayer?: boolean imageResourcesPath?: string hideForms?: boolean intent?: string annotationLayer?: boolean annotationsFilter?: string[] annotationsMap?: object watermarkText?: string watermarkOptions?: WatermarkOptions highlightText?: string highlightOptions?: HighlightOptions4. Watermark options.{ columns?: number rows?: number rotation?: number fontSize?: number color?: string }5. Highlight options.{ ignoreCase?: boolean completeWords?: boolean }Preview: Changelog:v1.9.6 (03/27/2024)Exports style.css explicitly in package.jsonRemoved overflow: hidden from container's stylesv1.9.5 (02/25/2024)'Loaded' events has been added in order to have more control about rendering process of page's layersv1.9.4 (01/13/2024)Add types and support for null and undefined values when reactivity is used with usePDFv1.9.3 (01/03/2024)Highlight all matches found in the same HTML elementv1.9.1 (01/02/2024)Added intent propAnnotationsLayer's imageResourcesPath bind prop fixedv1.9.0 (12/17/2023)Added support for outlinesAdded highlight-text and hightlight-options propsAdded width and height propsAdded a check for undefinied value in usePDF's srcv1.8.1 (11/16/2023)Make usePDF reactivewatermark-options prop addedv1.7.4 (09/13/2023)v1.7.2 (08/24/2023)Added support for High DPI screensFixed the position style of loading slotv1.7.0 (07/25/2023)Added support for XFA FormsAdded watermark textannotation-map functionality was improvedv1.5.1 (06/15/2023)v1.5.0 (06/11/2023)fix: annotations values missed when re-renderv1.4.8 (06/07/2023)v1.3.3 (01/20/2023)FIX: usePDF info ref fixedv1.2.3 (10/11/2022) PDF documents are the preferred format for many things, including sharing information with formatting and enhanced data visualization. Therefore, it's not surprising that the ability to open and annotate PDF documents has become an increasingly demanded feature for any web application as soon as it grows in size and complexity.However, adding such a feature to a web application usually means incrementing the number of "moving parts" of a codebase by orders of magnitude: PDF is a complex file format, which may deem the task overwhelming for any development team.You can simplify this task significantly by making use of PSPDFKit for Web, a JavaScript PDF library that can be used with (and without) any JavaScript framework, including Vue.js(opens in a new tab). It supports all modern mobile and desktop browsers (Chrome, Firefox, Safari, and Edge) and multiple languages, and it makes good use of the latest technologies available — like WebAssembly — to make the experience as performant as possible.PSPDFKit for Web comes in two flavors: server-backed and standalone. This means you can set it up as a shared collaboration tool that's integrated with your server backend, or as a client-side library with all the features you may need for your PDF document handling.To allow developers to easily embed our PDF library in their applications, there are several integration examples available. In this article, you'll learn how to integrate our Vue.js PDF library, which you can clone from the public PSPDFKit repository(opens in a new tab). You'll build a small app in a single HTML file that will fetch all the assets needed to load and run PSPDFKit for Web in a Vue.js app.The final result will look like the image below in your browser. It'll consist of a simple UI that allows you to open, view, and annotate PDF documents from within your Vue.js app.To get the example running, you need the following tools:Create a new Vue.js projectSelect Vue 3, depending on your preference.Navigate to the project directoryAdd the PSPDFKit dependencyPrepare the PSPDFKit libraryCreate a directory under public:Copy the PSPDFKit library:cp -R ./node_modules/pspdfkit/dist/pspdfkit-lib public/js/pspdfkit-libEnsure your server has the Content-Type: application/wasm MIME type set. This is required for PSPDFKit.Create the component fileIn the src/components directory, create a file named PSPDFKitContainer.vue.Start by defining the pdfFile prop in your component. This prop is required and will hold the path to the PDF file you want to display. By passing this as a prop, you can dynamically load different PDF files into the viewer from outside the component:Create the loadPSPDFKit methodNext, define the loadPSPDFKit method. This method is responsible for loading the PSPDFKit viewer with the PDF file provided through the pdfFile prop. Since PSPDFKit.load() returns a Promise, define the method as async.Make sure to unload any existing PSPDFKit instance before loading a new one. This prevents memory leaks and ensures the correct PDF is always displayed: PSPDFKit.unload('.pdf-container'); container: '.pdf-container',Clean up on component destructionTo avoid lingering references when the component is destroyed, use the beforeDestroy lifecycle hook to unload PSPDFKit. This ensures the viewer instance is properly cleaned up: PSPDFKit.unload('.pdf-container');Load PSPDFKit when the component is mountedTo make sure PSPDFKit is loaded as soon as the component is mounted, call the loadPSPDFKit method within the mounted lifecycle hook. Once the method resolves with the instance of PSPDFKit, emit a loaded event with the instance as the payload. This event enables you to use the instance elsewhere in your application: this.loadPSPDFKit().then((instance) => { this.$emit('loaded', instance);Watch for changes to the pdfFile propTo handle dynamic changes to the pdfFile prop, use a watch handler. This allows the component to reload the PDF whenever the prop changes. Additionally, include a check to ensure the new value is valid before attempting to reload PSPDFKit: if (val) this.loadPSPDFKit();Here's the final code that combines all these steps: import PSPDFKit from 'pspdfkit'; PSPDFKit.unload('.pdf-container'); container: '.pdf-container', PSPDFKit.unload('.pdf-container'); this.loadPSPDFKit().then((instance) => { this.$emit('loaded', instance); if (val) this.loadPSPDFKit();This component is now fully capable of loading PSPDFKit, rendering PDFs, unloading when destroyed, and dynamically switching PDFs based on the provided pdfFile prop.Use the scoped flag to make sure this CSS rule only applies to DOM nodes within the component itself. You can still overwrite CSS rules from outside, but it's good practice to prevent accidental overrides across your project. This way, the component can also be freely used between various projects without having to reapply styles in each of them.Now that you've created the PSPDFKitContainer.vue component, the next step is to import and use it within the main Vue.js page, where you want to display the PDF.In your main Vue component (e.g. App.vue), add the template structure that includes a file input for selecting the PDF file and the PSPDFKitContainer component to display the selected file: The element is used to select the PDF file.The PSPDFKitContainer component displays the PDF, with pdfFile as a prop containing the file path.Import the PSPDFKitContainer componentIn the script section, import the PSPDFKitContainer component so that it can be used in the template: import PSPDFKitContainer from '@/components/PSPDFKitContainer'; pdfFile: this.pdfFile || '/example.pdf', // Default PDF file to display. window.URL.revokeObjectURL(this.pdfFile); // Clean up old file reference. // Update `pdfFile` with the new file selected by the user. this.pdfFile = window.URL.createObjectURL(event.target.files[0]);The pdfFile data property stores the path to the currently selected PDF.The openDocument method handles the file input change, generating a URL for the selected file and assigning it to pdfFile.Prepare your PDF documentAdd the example.pdf file to the public directory of your Vue.js project. The public directory serves static files directly at the root of your project, making it easy to access resources like images, styles, and PDF files.With everything set up, run your Vue application:When you load a PDF, the PSPDFKit viewer will display the document, allowing you to read, annotate, and print it directly from your browser.While the setup above works well for displaying a PDF, it's also essential to manage the PSPDFKit instance when switching between different PDF files. To handle this, watch for changes in the pdfFile prop and reload the viewer when a new file is selected. You also need to handle the instance returned by PSPDFKit.load() for advanced functionality like annotations.Update the template to handle loaded eventsModify the template to listen for the loaded event emitted by the PSPDFKitContainer component: The @loaded="handleLoaded" directive listens for when the PSPDFKit instance is ready.In the script, add a method to handle the loaded event, which provides the PSPDFKit instance for further interaction:import PSPDFKitContainer from '@/components/PSPDFKitContainer'; pdfFile: this.pdfFile || '/example.pdf', window.URL.revokeObjectURL(this.pdfFile); this.pdfFile = window.URL.createObjectURL(event.target.files[0]); console.log("PSPDFKit has loaded: ", instance); // Perform any operations with the PSPDFKit instance, like adding annotations.Here's a step-by-step guide to adding custom annotations to your PDF in a Vue.js app using PSPDFKit.First, add a data() function to your component to store the PSPDFKit instance: instance: null, // This will hold the PSPDFKit instanceCustomize the toolbar to include a custom button that adds an ink annotation. Modify the loadPSPDFKit method to include the custom toolbar button, and set up the onPress handler: PSPDFKit.unload('.pdf-container'); const instance = await PSPDFKit.load({ container: '.pdf-container', title: 'Add Ink Annotation', className: 'addInkAnnotation', name: 'addInkAnnotation', console.error('PSPDFKit instance is not available'); const inkAnnotation = new PSPDFKit.Annotations.InkAnnotation({ lines: PSPDFKit.Immutable.List([PSPDFKit.Immutable.List([new PSPDFKit.Geometry.DrawingPoint({ x: 0, y: 0 }), new PSPDFKit.Geometry.DrawingPoint({ x: 100, y: 100 }), const createdAnnotations = await instance.create(inkAnnotation); console.log('Created Ink Annotations:', createdAnnotations); console.error('Error creating ink annotation:', error); this.instance = instance; this.$emit('loaded', instance); console.error('Error loading PSPDFKit:', error);Update the mounted lifecycle hook and the watch property to ensure the instance is updated when needed: this.loadPSPDFKit().then((instance) => { this.instance = instance; // Store the PSPDFKit instance. this.$emit('loaded', instance); // Emit an event when loaded. this.loadPSPDFKit().then((instance) => { this.instance = instance; // Update the instance when the PDF changes.Rebuild your Vue.js app and load a PDF. The toolbar will now include a button labeled Add Ink Annotation. When clicked, it adds an ink annotation with predefined drawing points to the first page of your PDF. Adjust the drawing points and other parameters as necessary to fit your specific requirements.This post focuses on adding an ink annotation, but PSPDFKit supports various annotation types, like text and images. Find out more about how to use PSPDFKit.Instance#create() by browsing the PSPDFKit for Web API reference.In this blog, you learned how to integrate PSPDFKit's JavaScript PDF library with the Vue.js framework. Once you've got it up and running, you can enable additional features and customizations in your application:At PSPDFKit, we offer a commercial, feature-rich, and completely customizable Vue.js PDF library that's easy to integrate and comes with well-documented APIs to handle advanced use cases. Try it for free, or visit our web demo to see it in action. PSPDFKit for Web is a JavaScript library that allows developers to integrate powerful PDF viewing and annotation capabilities into web applications, including those built with Vue.js. Yes, you can use PSPDFKit for Web as a standalone library without a backend, making it flexible for various application setups. PSPDFKit primarily supports PDF files, providing extensive features for viewing, annotating, and manipulating PDF documents. Yes, PSPDFKit offers a free trial that allows you to explore its features before committing to a purchase. You can report issues or request support through the PSPDFKit support page on their website. 11 Jun 202414 minutes to read You might need to open and view the PDF files from various location. In this section, you can find the information about how to open PDF files from URL, database and as base64 string. Opening a PDF from URL If you have your PDF files in the web, you can open it in the viewer using URL. Step 1: Create a Simple PDF Viewer Sample in Vue Start by following the steps provided in this link to create a simple PDF viewer sample in Vue. This will give you a basic setup of the PDF viewer component. Step 2: Modify the PdfViewerController.cs File in the Web Service Project Create a web service project in .NET Core 3.0 or above. You can refer to this link for instructions on how to create a web service project. Open the PdfViewerController.cs file in your web service project. Modify the Load() method to open it in the viewer using URL public IActionResult Load([FromBody] Dictionary jsonData) { // Initialize the PDF viewer object with memory cache object PdfRenderer pdfviewer = new PdfRenderer(_cache); MemoryStream stream = new MemoryStream(); object jsonResult = new object(); if (jsonObject != null && jsonObject.ContainsKey("document")) { if (bool.Parse(jsonObject["isFileName"])) { string documentPath = GetDocumentPath(jsonData["document"]); if (!string.IsNullOrEmpty(documentPath)) { byte[] bytes = System.IO.File.ReadAllBytes(documentPath); stream = new MemoryStream(bytes); } else { string fileName = jsonData["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0]; if (fileName == "http" || fileName == "https") { WebClient WebClient = new WebClient(); byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]); stream = new MemoryStream(pdfDoc); } else { return this.Content(jsonData["document"] + " is not found"); } } } else { byte[] bytes = Convert.FromBase64String(jsonObject["document"]); stream = new MemoryStream(bytes); } } jsonResult = pdfviewer.Load(stream, jsonObject); return Content(JsonConvert.SerializeObject(jsonResult)); } Step 3: Set the PDF Viewer Properties in React PDF viewer component Modify the serviceUrl property of the PDF viewer component with the accurate URL of your web service project, replacing with the actual URL of your server.Modify the documentPath with the correct PDF Document URL want to load. import { provide } from "vue"; import { PdfViewerComponent as EjsPdfviewer, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from '@syncfusion/ej2-vue-pdfviewer'; // Replace the "localhost:44396" with the actual URL of your server const serviceUrl = " ; // Replace correct PDF Document URL want to load const documentPath = " ; provide('PdfViewer', [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner]); import { PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from '@syncfusion/ej2-vue-pdfviewer'; export default { name: 'app', components: { 'ejs-pdfviewer': PdfViewerComponent }, data() { return { // Replace the "localhost:44396" with the actual URL of your server serviceUrl: " , // Replace correct PDF Document URL want to load documentPath: " }; }, provide: { PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner] } } View sample in GitHub Opening a PDF from base64 data The following code steps how the PDF file can be loaded in PDF Viewer as base64 string. Step 1: Create a Simple PDF Viewer Sample in Angular Start by following the steps provided in this link to create a simple PDF viewer sample in Angular. This will give you a basic setup of the PDF viewer component. Step 2: Use the following code snippet to load the PDF document using a base64 string. LoadDocumentFromBase64 import { provide, ref } from "vue"; import { PdfViewerComponent as EjsPdfviewer, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from '@syncfusion/ej2-vue-pdfviewer'; const pdfviewer = ref(null); // Replace the "localhost:44396" with the actual URL of your server const serviceUrl = " ; provide('PdfViewer', [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner]); // Event triggers on the Export FDF button click. const load = function () { pdfviewer.value.ej2Instances.load('data:application/pdf;base64,' + AddBase64String, null); } LoadDocumentFromBase64 import { PdfViewerComponent, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from '@syncfusion/ej2-vue-pdfviewer'; export default { name: 'App', components: { 'ejs-pdfviewer': PdfViewerComponent }, data() { return { // Replace the "localhost:44396" with the actual URL of your server serviceUrl: " } }, methods: { // Event triggers on the Export PDF button click. load: function () { pdfviewer.value.ej2Instances.load('data:application/pdf;base64,' + AddBase64String, null); } }, provide: { PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner] } } View sample in GitHub Opening a PDF from database To load a PDF file from SQL Server database in a PDF Viewer, you can follow the steps below Step 1: Create a Simple PDF Viewer Sample in Vue Start by following the steps provided in this link to create a simple PDF viewer sample in Vue. This will give you a basic setup of the PDF viewer component. Step 2: Modify the PdfViewerController.cs File in the Web Service Project Create a web service project in .NET Core 3.0 or above. You can refer to this link for instructions on how to create a web service project. Open the PdfViewerController.cs file in your web service project. Import the required namespaces at the top of the file: using System.IO; using System.Data.SqlClient; Add the following private fields and constructor parameters to the PdfViewerController class, In the constructor, assign the values from the configuration to the corresponding fields private IConfiguration _configuration; public readonly string _connectionString; public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache, IConfiguration configuration) { _hostingEnvironment = hostingEnvironment; _cache = cache; _configuration = configuration; _connectionString = _configuration.GetValue("ConnectionString"); } Modify the Load() method to open it in the viewer using URL public IActionResult Load([FromBody] Dictionary jsonData) { // Initialize the PDF viewer object with memory cache object PdfRenderer pdfviewer = new PdfRenderer(_cache); MemoryStream stream = new MemoryStream(); object jsonResult = new object(); if (jsonObject != null && jsonObject.ContainsKey("document")) { if (bool.Parse(jsonObject["isFileName"])) { string documentPath = GetDocumentPath(jsonData["document"]); if (!string.IsNullOrEmpty(documentPath)) { byte[] bytes = System.IO.File.ReadAllBytes(documentPath); stream = new MemoryStream(bytes); } string documentName = jsonObject["document"]; string connectionString = _connectionString; System.Data.SqlClient.SqlConnection connection = new System.Data.SqlClient.SqlConnection(connectionString); //Searches for the PDF document from the database string query = "SELECT FileData FROM Table WHERE FileName = '" + documentName + "'"; System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand(query, connection); connection.Open(); using (SqlDataReader reader = command.ExecuteReader()) { if (reader.Read()) { byte[] byteArray = (byte[])reader["FileData"]; stream = new MemoryStream(byteArray); } } } else { byte[] bytes = Convert.FromBase64String(jsonObject["document"]); stream = new MemoryStream(bytes); } } jsonResult = pdfviewer.Load(stream, jsonObject); return Content(JsonConvert.SerializeObject(jsonResult)); } Open the appsettings.json file in your web service project, Add the following lines below the existing "AllowedHosts" configuration { "Logging": { "LogLevel": { "Default": "Information", "Microsoft.AspNetCore": "Warning" } }, "AllowedHosts": "*", "ConnectionString": "Your connection string for SQL server" } Replace Your Connection string from SQL server with the actual connection string for your SQL Server database The System.Data.SqlClient package must be installed in your application to use the previous code example. You need to modify the connectionString variable in the previous code example as per the connection string of your database. View sample in GitHub Vue.js(opens in a new tab) is a frontend JavaScript framework for building single-page applications (SPAs) and user interfaces (UIs), and it's the second-most starred GitHub repository(opens in a new tab). It enables users to create rapid prototypes and build fast and reliable applications.In this blog post, we'll use Vue.js to create a PDF viewer with PDF.js. PDF.js(opens in a new tab) is an open source JavaScript library that allows you to view PDF files in your browser.In the first part, we'll look at how to create the PDF viewer with an open source library. In the second part, we'll provide a step-by-step guide on how to integrate the PSPDFKit Vue.js PDF viewer library into the Vue.js project. Our viewer library provides some benefits beyond what PDF.js provides, including:A prebuilt UI — Save time with a well-documented list of APIs when customizing the UI to meet your exact requirements.Annotation tools — Draw, circle, highlight, comment, and add notes to documents with 15+ prebuilt annotation tools.Multiple file types — Support client-side viewing of PDFs, MS Office documents, and image files.30+ features — Easily add features like PDF editing, digital signatures, form filling, real-time document collaboration, and more.Dedicated support — Deploy faster by working 1-on-1 with our developers.PDF viewers are essential tools for enhancing user experience in web applications by allowing users to view and interact with PDF documents directly within the browser. This seamless integration eliminates the need to download files or open them in a separate application, saving time and creating a more cohesive experience.A PDF viewer lets users view documents without leaving the current tab, which is particularly useful for applications where document interaction is frequent. PDF viewers typically offer features such as zoom, page navigation, and text selection, allowing users to interact with and read content more effectively.For Vue.js developers, there are both open source and commercial options available for implementing a PDF viewer:Open source libraries like PDF.js, vue-pdf, and pdfvuer offer essential functionality and are straightforward to integrate, making them suitable for basic document viewing needs.Commercial solutions like Nutrient offer advanced features such as annotations, form filling, and real-time collaboration, providing a more feature-rich experience for complex applications.This article will explore both open source and commercial options, guiding you through building a Vue.js PDF viewer with PDF.js and integrating Nutrient for a more robust solution.To get started, you'll need:There are two main PDF.js wrappers available to make the task of creating a PDF viewer with Vue.js easier: vue-pdf(opens in a new tab) and pdfvuer(opens in a new tab). Let's look at their respective advantages and disadvantages and decide which one to use.It has around 163K monthly downloads on npm.It's easy to use, and it has an easy setup.It currently doesn't support Vue 3.The library owner isn't responsive.When printing your files, you may get a bug(opens in a new tab). There's been a solution for this since 2019, but the PR(opens in a new tab) isn't merged into the main branch.It's not actively maintained.It lacks documentation and examples.It has around 22K monthly downloads on npm.It has support for Vue 2 and Vue 3.It lacks documentation and examples.It doesn't have many options to customize your project.In this blog post, we'll use the pdfvuer library because it supports both version 2 and version 3 of Vue.js.To work with Vue.js, we need to install Vue CLI(opens in a new tab) (command-line interface), which is standard tooling for Vue.js. It helps you create, build, and run Vue.js applications.You can install the CLI using npm(opens in a new tab), which comes with Node.js, or yarn(opens in a new tab):You can check the version of Vue by running the following:In this blog post, we're using Vue CLI version 4.5.15.Vue CLI gives us an easy way to create our projects by using the following command:vue create pdfvuer-exampleHere, we're using the create option with the name of the project we want to create (pdfvuer-example).It'll then ask some configuration questions.Select Default (Vue 3) ([Vue 3] babel, eslint) from the list.Now, change the directory to pdfvuer-example:Run the command below to install the pdfvuer library via npm or yarn. This will work with Vue 3:npm install pdfvuer@next --saveAdd your PDF document to the public directory. You can use our demo document as an example.Now, go to your App.vue file inside the src directory and add the following code: import pdf from 'pdfvuer';In the script tag, we're importing and exporting the pdf component from the pdfvuer library.In the template tag, we're creating a pdf element by passing the src attribute with the name of the PDF file. We're also passing the page attribute with the number of the page we want to display. It'll show the first page of our document.When you start the app, you'll see the PDF rendered in the browser. Use the following command to run the project:You can see the app running at ��� Tip: You can access the source code for this project on GitHub(opens in a new tab).As you can see, this library didn't give us many options to customize the PDF viewer. We can only display the first page of our document, and there's no way to navigate through the pages.We'll move on to the next part of the article to see how PSPDFKit for Web can help us.PSPDFKit offers a versatile PDF library that can be used to build a Vue.js PDF viewer. It provides more than 30 out-of-the-box features, including:You can integrate it into your new or existing Vue.js projects with a couple of steps.Now, let's go back to our tutorial and see how to integrate PSPDFKit into your Vue.js project.Create a new Vue.js project for PSPDFKit integration:vue create pspdfkit-vue-projectThis will ask some configuration questions.Select Default (Vue 3) ([Vue 3] babel, eslint) from the list, and change the directory to pspdfkit-vue-project:Install pspdfkit as a dependency with npm or yarn:Now, we can start building our Vue.js project. First, let's create a js directory under the public directory. Go to your terminal and run:Copy the PSPDFKit for Web library assets to the public/js directory:cp -R ./node_modules/pspdfkit/dist/pspdfkit-lib public/js/pspdfkit-libThis will copy the pspdfkit-lib directory from within node_modules/ into the public/js/ directory to make it available to the SDK at runtime.Add the PDF document you want to display to the public directory. You can use our demo document as an example.Add a component wrapper for the PSPDFKit library and save it as src/components/PSPDFKitContainer.vue: import PSPDFKit from 'pspdfkit'; * The component receives the `pdfFile` prop, which is type of `String` and is required. * We wait until the template has been rendered to load the document into the library. this.loadPSPDFKit().then((instance) => { this.$emit('loaded', instance); * We watch for `pdfFile` prop changes and trigger unloading and loading when there's a new document to load. * Our component has the `loadPSPDFKit` method. This unloads and cleans up the component and triggers document loading. PSPDFKit.unload('.pdf-container'); // To access the `pdfFile` from props, use `this` keyword. container: '.pdf-container', * Clean up when the component is unmounted so it's ready to load another document (not needed in this example). PSPDFKit.unload('.pdf-container');Let's look at what's happening in our component:The template section is rendering a div with the pdf-container class. This will help us declaratively bind the rendered DOM to the underlying component instance's data.The script section is defining a Vue.js instance named PSPDFKit and creating methods for mounting, loading, and unloading PDF files into the pdf-container.The style section is defining the height of the container.Now, replace the contents of src/App.vue with the following: import PSPDFKitContainer from '@/components/PSPDFKitContainer'; pdfFile: this.pdfFile || '/example.pdf', * Render the `PSPDFKitContainer` component. * Our component has two methods — one to check when the document is loaded, and the other to open the document. console.log('PSPDFKit has loaded: ', instance); // To access the Vue instance data properties, use `this` keyword. window.URL.revokeObjectURL(this.pdfFile); this.pdfFile = window.URL.createObjectURL(In the template section, we have a file upload input and the PSPDFKitContainer component.Vue.js uses directives to handle some types of functionality. For the input field, we're using the 'v-on' directive to attach an event listener to the element. In our case, it's the 'change' event. There's a shortcut to 'v-on' that removes the keyword and uses an '@' symbol instead.v-on:change="openDocument"v-on:loaded="handleLoaded"Similar to the input field, for the PSPDFKitContainer component, we're using the v-bind directive to bind the pdfFile property to the pdfFile property of the component and attaching an event listener for the loaded event:In the script section, you can see the implementation of the handleLoaded and openDocument methods. Also, there's a data function that returns the pdfFile property.The data keeps track of reactive state within the current component. It's always a function and returns an object. The object's top-level properties are exposed via the component instance.In the style section, we have styles for custom file input, and there are some general styles for the app.You can see the application running on localhost:8080. If you can't see your PDF file rendered in the browser, make sure you actually uploaded a PDF file inside the public directory.In the demo application, we can open different PDF files by clicking the Open PDF button. We can add signatures, annotations, stamps, and more. Tip: All the finished code is available on GitHub(opens in a new tab). 😎 You can find the example code for Vue 2 in the vue-2 branch.In this tutorial, we looked at how to build both a Vue.js PDF viewer with an open source library and a PSPDFKit Vue.js PDF viewer that allows you to display and render PDF files in your Vue.js application.We created similar how-to blog posts using different web frameworks and libraries:Open source Vue.js libraries are good options if you want to build the UI and features yourself. However, this can get complicated easily, as you may not get the support you need. Opting for a commercial solution lets you focus on other areas of your business and move up the value chain.At PSPDFKit, we offer a commercial, feature-rich, and completely customizable JavaScript PDF library that's easy to integrate and comes with well-documented APIs to handle advanced use cases. Try it for free, or visit our demo(opens in a new tab) to see it in action. You can install pdfvuer using npm with the command npm install pdfvuer@next --save or with yarn using yarn add pdfvuer@next. PSPDFKit offers more than 30 features including annotation tools, PDF form filling, document editing, and real-time collaboration, along with dedicated support. Add the pdfvuer library to your project, then use the component in your Vue template to display the PDF by specifying the src attribute with the path to your PDF file. Use a file input element to upload PDF files, create an object URL for the selected file, and pass it to the PSPDFKit component in your Vue.js application. Since my last post on building PDF Viewer with Vue.js, I have been hard at work to develop a library that just works: Vue PDF Viewer. After being in private beta for a few months, we officially launched Vue PDF Viewer earlier this month. In this article, I'll be sharing with you how to render a PDF in Vue.js in just minutes, along with a few use cases to help you make the most out of this library. Why Vue PDF Viewer? Let's start with why Vue PDF Viewer stands out. Here's a quick rundown of its key features: Built for Vue: Designed specifically for Vue.js developers, Vue PDF Viewer is designed using familiar Vue's component-based structure, syntax and state management for Vue.js developers to customize natively. Quick Setup: Go from zero to a fully functional PDF viewer in just 3 simple steps. Customizable: Supports themes, responsive layouts, and custom icons, allowing you to tailor the viewer to fit your app's design. High Performance: Optimized for handling multiple PDFs without sacrificing speed or performance. Now, let's dive into the step-by-step process to get started with Vue PDF Viewer. Displaying a PDF Viewer Component Vue PDF Viewer supports Vue 3. It also works seamlessly across a wide range of environments, including: Vue 3 – Composition API (TypeScript, JavaScript) Vue 3 – Options API (TypeScript, JavaScript) Vue 3 – Server-Side rendering (TypeScript) Nuxt VitePress Quasar For this demo, I'll show you how to add the Vue PDF Viewer component to a Vue app. Step 1: Adding the Library To get started, add the Vue PDF Viewer library to your project. It's a quick and easy installation. bun add @vue-pdf-viewer/viewer Remark: You can use yarn, npm or pnpm. Step 2: Importing the Component Next, we'll import the PDF viewer component into your Vue application. import { VPdfViewer } from "@vue-pdf-viewer/viewer"; Step 3: Render the Component Finally, render the component and pass in the required props to display your PDF document. import { VPdfViewer } from "@vue-pdf-viewer/viewer";