

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

## Lesson Outline

- Introduction - Introduce the topic and that the instruction will be covered using Vue.js
- Note the limitations of running a fully front end upload with respect to security and upload options available which could be overcome by using a server SDK
- Create a **preset** and explain how it is the front end authentication alternative that keeps your API\_SECRET and API\_KEY from being accessible
- Detail what types of files can be uploaded to a Cloudinary account, defining “Images”, “Videos” and “Raw Files”.
- Ensure students have downloaded and installed the Cloudinary NPM, as well as followed all of the details in the Environment Setup course.
- Ensure that students have installed Vue 2 and Vue CLI 3 and can navigate to the HelloWorld page on their browser
- Demo how the application page will function after the code changes are made and show how it will look in the DAM
- Using a HelloWorld.vue that has been modified to run the endpoint upload with axios point out code changes
  - State: look at data()
  - Template: look at how a form with a file input has been setup and point out a submit button that will remain disabled until a file is selected
  - Methods:
    - Code that detects that a file has been selected and enables the submit button
    - Code that serves as a submit handler and instantiates a FileReader to read the contents of the file
    - Code that loads the **contents of the file**, the **preset** and optional data like **tags** into a Form object
    - Code that imports axios and posts the form to the v1\_1 image upload endpoint which contains the **cloud\_name**
  - Console: look at how the image is displayed using the URL returned in a successful response and inspect the logged response to note what data is available for further programmatic work
- End the session, referring users to documentation that references the endpoint as well as working with presets.

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	Action	Narration
Intro	Show the Cloudinary, Vue.js, npm Logos on the screen.	Hello, and welcome to this tutorial describing how you can upload images to your Cloudinary Cloud using the Vue.js framework.  The same code can be used to upload video and raw data such as PDFs and JSON.
	Show a picture of transformed media from Cloudinary	Once media assets are uploaded to Cloudinary they may be easily managed, transformed and delivered across the web
	Show axios library on npm.	We'll be using the axios library to post data using a Cloudinary endpoint. Axios is a popular library for reasons such as cross site scripting support and automatic transforms of JSON data.
	Show logs for React, Svelte	The pattern for upload presented here can be generalized to other JavaScript frameworks such as React and Svelte.
Demo	Video walkthrough of finished app	Let's take a look at the Vue.js app we'll be building here before we look at how to build it.
	Pull up the finished app in the browser	In this app we'll provide a form that allows the user to select a file from their hard drive and upload to the cloud.
	Highlight the cloud_name and preset inputs.	We'll talk more in a bit about authorization for front end app with Cloudinary and where to obtain these values. The inputs for cloudname and preset provide the credentials needed to execute the upload.

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	Highlight the submit button.	<p>The submit button is disabled.</p> <p>We'll enable the submit button once the file has been chosen by the user. This will be evident when the name of the file appears next to the submit button.</p>
	Choose and open the cloudinary logo image.	<p>Let's start by choosing an image from the file system. I'm going to choose the Cloudinary logo. I'm only allowed to choose jpeg and png file formats. This is because, for this app I want to restrict the user to uploading images only. It's possible to upload video and raw files as well. All files in my file system would be selectable if I removed an "accept" attribute from my input HTML code.</p>
	Highlight the name of the file next to the submit button.	<p>We can now see the file name we've selected in the form</p>
	Highlight the submit button.	<p>And the submit button is enabled</p>
	Click Submit	<p>Let's click submit</p>
	Highlight the error	<p>If you try to submit with just the file you'll get a message that you need a cloud_name and preset to continue.</p>
	Enter a preconfigured cloud_name and preset. (file name should still be selected)	<p>I'll enter a cloudname and preset that I have already established. You'll get a cloudname for free when you register for Cloudinary account. We'll be describing how to create a preset shortly.</p>
	Click submit again	<p>The cloudinary logo is rendered on my page. Let's take a quick look at the console in Chrome Dev tools as there is some logging taking place in this app.</p>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	Open dev console or show screenshot of response data object	<p>This will give us some quick insight into where to get the URL to display the cloudinary image after the upload.</p> <p>The cloudinary upload API returns information about the uploaded asset. In the app, I am able to bind the value of the secured url to an image tag in the template.</p>
Media Library	Show a screenshot of the image uploaded to the ML	You can also find the image we just uploaded on the cloud. It has been placed in the folder we specified in the preset. The public id, which is a unique identifier has been assigned to it by Cloudinary during the upload.
	Image of preset options in docs with a hotlink.	The public id is configurable in the preset.
How to		Let's look at how to code this application. We'll start with security.
Security	Screenshot of the Web Console Dashboard and where the security credentials are located.	<p>Cloudinary provides robust security using API_KEY and API_SECRET to create signed uploads. Signed uploads require computation or configuration that is not possible on the Front End without exposing the KEY and SECRET.</p> <p>In order to create a fully Front End secure app, Cloudinary provides a configurable unsigned preset.</p>
	<p>Static page showing bullet points to some of the things you can do in a preset</p> <ul style="list-style-type: none"><li>• Create the public id based on the file name</li><li>• Upload to a specific folder in the cloud</li><li>• Use Addons like Amazon Rekognition, Google Tagging to add tags to</li></ul>	<p>In a fully front end app where we don't want to reveal API keys and secrets, we can create an <b>unsigned preset</b>. The preset is required to post data to cloudinary with the endpoint that we'll be using with axios. The preset has many options and can be configured to take many actions in the process of uploading the asset.</p>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial


Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	<ul style="list-style-type: none"><li>the asset<ul style="list-style-type: none"><li>• Use Amazon Moderation to reject an image if it contains unwanted content</li><li>• Transform the asset on upload</li></ul></li></ul>	
	Screenshot and hotlink to a video that explains OCR text recognition <a href="https://support.cloudinary.com/hc/en-us/articles/360004967272-Upload-Preset-Configuration">https://support.cloudinary.com/hc/en-us/articles/360004967272-Upload-Preset-Configuration</a>	If you're interested in using add ons in your preset, check out this video on creating a preset with the OCR text recognition add on.
	Video to create an unsigned preset that puts asset into a folder.	For this demonstration using Vue.js we must create an unsigned preset.
	Show where to get cloudname on CL Console Dashboard	After you've registered, you can look at your console dashboard to find your cloud_name. This cloud_name will be input into the form in our app.
	Video, click on gear, click on upload tab, scroll to bottom, highlight the unsigned enabled text	You can create your unsigned preset by clicking on the settings gear icon and then the upload tab. Scroll to the bottom and make sure that unsigned loading is enabled. You will have to click on a link to enable it the first time you try this.
	Click on add upload preset	Scroll to the link labeled "Add Upload Preset" and click on it
		You can leave the random name assigned to it, but I'm going to change it to "vue-upload". Then I'll selected signing mode "unsigned". Then I'll enter "vue-upload" for the name of the folder I want the assets uploaded to. This is optional, but handy for finding the assets once uploaded with the app. Finally don't forget to press the Save button.

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

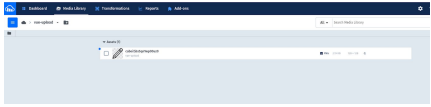
Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

		The first time you upload the folder will be created for you if it doesn't exist. This name of the preset is the value you'll enter into to the form in the app we're creating.
Look at Code		
	<p>Screenshot of steps to prepare the Vue.js HelloWorld.</p> <p>It should show npm create - hotlink to a Vue.js docs page</p> <p>Share a link to github repo or Code Sandbox: Code in CodeSandbox: <a href="https://ht1ig.csb.app/">https://ht1ig.csb.app/</a></p>	We're now ready to begin coding. Set up a HelloWorld application using the the Vue CLI3 and Vue 2. In order to focus on the code required to upload an image, we'll be modifying the HelloWorld component rather than creating a component from scratch.
Data	<p>In code, look at state/data code.</p> <pre>data() {   return {     results: null,     errors: [],     file: null,     filesSelected: 0,     cloudName: "",     preset: "",     tags: "browser-upload"   }; }</pre>	<p>The HelloWorld component has been modified to create a Cloudinary image upload component.</p> <p>I start by defining my data structures that make up the state of my application. This data includes <b>results</b> is the data object returned by Axios <b>errors</b> is an empty array which I can fill with error I encounter in running the code especially. I am especially interested in any errors encountered when I post the cloudinary api to upload my image. <b>file</b> is an object that will contain information about the file that is selected in the browser form. <b>filesSelected</b> is a count of the number of files selected. I'll only be processing the first file selected, but I'll use this count as a flag to indicate that the submit button should be disabled. When the count is &lt; 1, the submit button will remain disabled <b>cloudName</b> is initialized as an empty string that is bound to the form. <b>preset</b> is initialized as an empty string that is bound to</p>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

		<p>the form.</p> <p><b>tags</b> is string containing a comma separated list of tags. It's optional but I'm using it to demonstrate how options can be added to the upload call and it is useful if you want custom tags.</p>
Template	<p>In codenow, look at template code.</p> <p>Highlight the sections of the template as they are explained</p>	<p>Since I'm reusing the HelloWorld component, I have completely removed the Vue.js Hello World template and replaced with HTML that will render a form to collect and submit the image that the user wants to upload.</p>
	<pre>&lt;!-- supply h2 heading --&gt; &lt;h2&gt;Upload an Image to Cloudinary&lt;/h2&gt;</pre>	<p>We start with a heading to set the context for the form.</p>
	<pre>&lt;!-- Create a form that will not submit to a server but will prevent submit and use the upload function as a handle--&gt; &lt;form v-on:submit.prevent="upload"&gt;</pre> <p>Highlight the v-on directive</p>	<p>If we were submitting this form to a server, we would need to add the enctype attribute and set it to "multipart/form-data", but since this is a single page Front End app, we are actually disabling the submit to server as that would just refresh the page. To process the form, we instead use the v-on directive to call for execution of the Vue.js method named "upload"</p>
	<pre>&lt;!-- bind cloud-name to the input --&gt; &lt;label for="cloudname"&gt;Cloud Name&lt;/label&gt; &lt;input id="cloudname" type="text" v-model="cloudname" placeholder="Enter cloud_name from dashboard" /&gt;  &lt;!-- bind preset to the input --&gt; &lt;label for="preset"&gt;Preset&lt;/label&gt; &lt;input id="preset" type="text" v-model="preset" placeholder="Enter preset from upload settings" /&gt;</pre>	<p>Vue.js provides 2 way binding with the v-model directive. We bind the cloud_name and preset form inputs to our Vue.js data.</p>
	<pre>&lt;!-- define the event to select an image file and when they have selected it call a function to handle this event --&gt; &lt;label for="file"&gt;File Input&lt;/label&gt; &lt;input   id="file" type="file"   @change="handleFileChange(\$event)" /&gt;</pre>	<p>Let's examine the parts of the file input tag.</p>
	<p>Highlight type="file"</p> <pre>type="file"</pre>	<p>Setting the input type to "file" causes the browser to render a button allows us to navigate the local file system to find a file. We chose the file by clicking "open". Once the file is chose the name is displayed next to the button</p>
	<p>Highlight the "accept" attribute</p> <pre>accept="image/png, image/jpeg"</pre>	<p>Setting the "accept" attribute to some specific image formats causes the navigation of the file system to enable only files that fit that format.</p>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	<p>Highlight the @change</p> <pre>@change="handleFileChange(\$event) "</pre>	<p>We can bind event listener to the input change event using @change. When a change to the input occurs the function specified is called.</p>
	<p>Highlight the v-if</p> <pre>&lt;p v-if="results &amp;&amp; results.secure_url"&gt;   &lt;img     :src="results.secure_url"     :alt="results.public_id"   /&gt; &lt;/p&gt;</pre>	<p>Finally when we get a successful response, we'll have access to the URL and public id, as well as other information that we can bind to the template.</p> <p>We can detect a successful response because the axios data will be referenced by our results data.</p> <p>In this app, we're using the secure url to render the uploaded image from Cloudinary by assigning it to the src of the img tag. You might imagine an app the uploaded image and then posted the public_id to a database to support other enterprise apps. The public id can be used to access, transform and deliver and uploaded image.</p>
	<p>Highlight the v-if</p> <pre>&lt;!-- display errors if not successful --&gt; &lt;section&gt;   &lt;ul     v-if="errors.length &gt; 0"&gt;     &lt;li       v-for="(error,index) in errors"</pre>	<p>If there is an error after the submit button is clicked or during upload, the errors array will be bound to an unordered list to inform the user of the problem.</p>



Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	<pre>:key="index"&gt;{{error}}&lt;/li&gt; i&gt; &lt;/ul&gt; &lt;/section&gt;</pre>	
Methods	<p>In code, look at methods: file select handler</p> <p>Highlight these sections of code as they are called out.</p>	<p>There are 2 methods that support file upload: <b>handleFileChange</b> that is called when the user selects a file and <b>upload</b> that is called when the user clicks the submit button.</p>
	<pre>handleFileChange: function(event) {   event.target.files);   //returns an array of files even though multiple not used   this.file = event.target.files[0];   this.filesSelected = event.target.files.length ; },</pre>	<p>The file input element returns an array of files. In this application we are only going to upload <b>one</b> file, although it would be a small change to upload multiple files.</p> <p>In the <b>handleFileChange</b> method extract the name of the file selected and reference with the state variable "files". The event target provides the array with the file information.</p> <p>This is not the contents of the file, just information about the file to be used to read the content.</p> <p>We also set the "filesSelected" counter to the number of files selected. If it's greater than one it will cause the submit button to be enabled.</p>
	<p>Create a graphic that shows the actions taken in the upload method</p> <ul style="list-style-type: none"><li>● reading in the data from the selected file using</li></ul>	<p>In the <b>upload</b> method we read the contents of the file and load it and other upload api data into a form which we post to axios. This method could be refactored, but it is left as is to fully encapsulate the idea of getting the data from the file and into Cloudinary.</p>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

	<p>the JavaScript FileReader API</p> <ul style="list-style-type: none"><li>• creating a <b>FormData</b> object to hold the file data and the information required by Cloudinary to handle a post to its upload endpoint</li><li>• Posting the form to the Cloudinary upload endpoint</li></ul>	
	<pre>let reader = new FileReader();</pre>	Instantiate the FileReader
	<pre>reader.addEventListener( "load",function(){   UPLOAD THE FILE }).bind(this),false);</pre>	Attach listener that is triggered when the data from the file is loaded into memory.
	<pre>if (this.file &amp;&amp; this.file.name) {   reader.readAsDataURL(this     .file);}</pre>	Verify that a file has been selected and that it has a name and then call the reader function <b>readAsDataURL</b>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

		<p>We attach a listener to the reader to listen for the load event.</p> <p>The code to handle the load event is set up to so that <b>this</b> references the state of the Vue.js component using the bind function.</p> <p>When this event has been fired, we are ready to extract the data from the <b>reader.result</b>.</p> <p>We create a FormData object and add the preset, any optional data like a custom tag, and the file data.</p> <p>Next, we build out the upload endpoint by adding our own cloudName to the URL</p>
--	--	--

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

		<p>We're going to use the axios library to post data to Cloudinary. We begin by creating a request object that encapsulates URL, Method and Data</p> <p>Axios returns a promise. Therefore as we post the data to Cloudinary we code for the success and failure of the call. If it's successful we reference the data object in the returned response with our stateful, reactive <b>results</b> variable.</p> <p>If it fails we push the error to our stateful, reactive errors array</p> <p>In addition, we log the response for both success and failure.</p>
Console	In the console, look at the response to see where the public id and secure url are coming from. Show this through video or a static page that shows the response object in the chrome dev tools	Vue.js reactive data takes care of binding the response data to the template so that the image is displayed

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>

Docs	<p>Documentation page on the image/upload endpoint</p> <p>Bring up the docs and provide hotlinks</p> <p><a href="https://cloudinary.com/documentation/vue_image_and_video_upload">https://cloudinary.com/documentation/vue_image_and_video_upload</a></p> <p><a href="https://codepen.io/team/Cloudinary/pen/QgpyOK">https://codepen.io/team/Cloudinary/pen/QgpyOK</a></p> <p>Provide a link to a repo with the code demo'd here.</p>	<p>Look at the documentation which describes the upload method supplied by cloudinary. You'll notice that we supply an upload widget and a jQuery SDK. You can wrap these in a Vue.js component, but there will be some overhead in terms of the code that you would need build if you're using a bundler like Webpack.</p> <p>The link to the Codepen give an example of using pure JavaScript and XHR. You can also wrap this in a Vue.js component.</p> <p>If you want to take advantage of axios to load call the upload endpoint, you can use reference this github repo. Axios offer event handling like XHR if you want to show a status bar to help your user get feedback on the upload.</p>
	<p>Documentation showing limits of using presets for uploads.</p> <p>Show this documentation with hotlinks</p> <p><a href="https://cloudinary.com/documentation/upload_images#upload_presets">https://cloudinary.com/documentation/upload_images#upload_presets</a></p> <p><a href="https://support.cloudinary.com/hc/en-us/articles/208335975-How-safe-secure-is-it-to-use-unsigned-upload-from-web-browsers-or-mobile-clients-">https://support.cloudinary.com/hc/en-us/articles/208335975-How-safe-secure-is-it-to-use-unsigned-upload-from-web-browsers-or-mobile-clients-</a></p> <p><a href="https://support.cloudinary.com/hc/en-us/articles/204046472-Which-upload-parameters-are-allowed-when-using-unsigned-upload">https://support.cloudinary.com/hc/en-us/articles/204046472-Which-upload-parameters-are-allowed-when-using-unsigned-upload</a></p> <p>=</p>	<p>If you want to read more about using presets look at the documentation on preset options. Preset are designed for both the front end and backend.</p> <p>Not all of the options are available to unsigned presets. You may want to read the documentation to discover some of the limitations and possible security issues.</p>

Title: Using Vue.js as a Cloudinary Front End Upload Solution - Developer Tutorial

Audience: Developers using the Vue.js who need to understand the basics of using the image upload API endpoint within the Vue.js framework.

Goal/Focus: To demonstrate how the image upload API can be used in a purely front end application solution and to point out how a backend solution can provide more options.

Creator: Becky Peltz, Developer Instructional Designer for Cloudinary

<https://www.beckypeltz.me/vue-cloudinary-file-upload/index.html>

<https://github.com/rebeccapeltz/vue-cloudinary-file-upload>