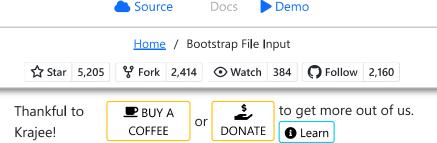


Bootstrap File Input

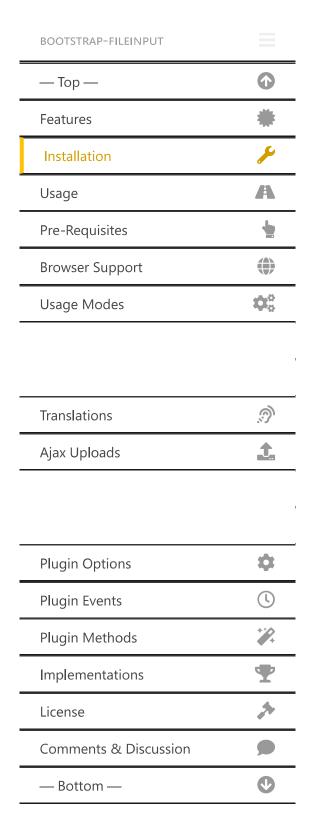


An enhanced HTML 5 file input for Bootstrap 5.x or Bootstrap 4.x or Bootstrap 3.x with file preview for various files, offers multiple selection, and more. The plugin allows you a simple way to setup an advanced file picker/upload control built to work specially with Bootstrap CSS3 styles. It enhances the file input functionality further, by offering support to preview a wide variety of files i.e. images, text, html, video, audio, flash, and objects. In addition, it includes AJAX based uploads, dragging & dropping files, viewing upload progress, and selectively previewing, adding, or deleting files.

View a complete demo.

NEW: Release v5.0.0 is a major rewrite that incorporates various new enhancements and features like resumable chunk uploads and more. A summary of the features available with this release are documented in the Release 5.0.0 milestone.

Bootstrap 5.x Support is available with release v5.2.0. From release v5.2.0, the bootstrap version



overridden via the new global property
\$.fn.fileinputBsVersion. Note that as of May
2021, an issue exists with Bootstrap 5.x on Modal
dialog initialization which is recorded in this issue
with a workaround.

Bootstrap 4.x Support is also available with release v4.4.4. The release 4.4.4 also includes various preview and styling enhancements including support for smaller screen devices.

Post release v4.0.0, the plugin supports AJAX based uploads using HTML 5 FormData and XHR2 protocol, which is supported in most modern browsers. It also has inbuilt support for AJAX based file deletion from the server. This thereby allows powerful features to append, add, remove files on the fly. The plugin also has added DRAG & DROP support for ajax uploads. In the event, the browser does not support FormData or XHR2, the plugin degrades it to a normal form submission.

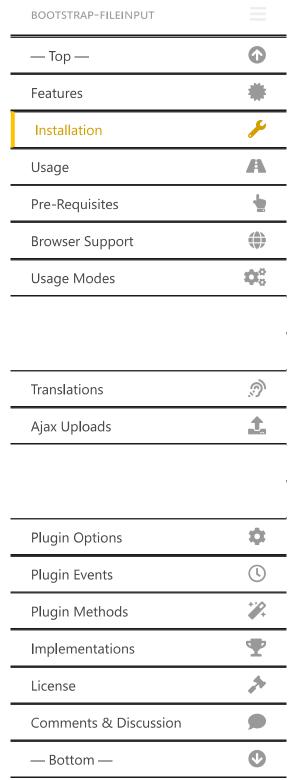
This plugin was initially inspired by this blog article and Jasny's File Input plugin. But the plugin has now matured with various additional features and enhancements to be a complete (yet simple) file management tool and solution for web developers.

Note

You can <u>refer this webtip</u> for an example of processing ajax based uploads using PHP.

Tip

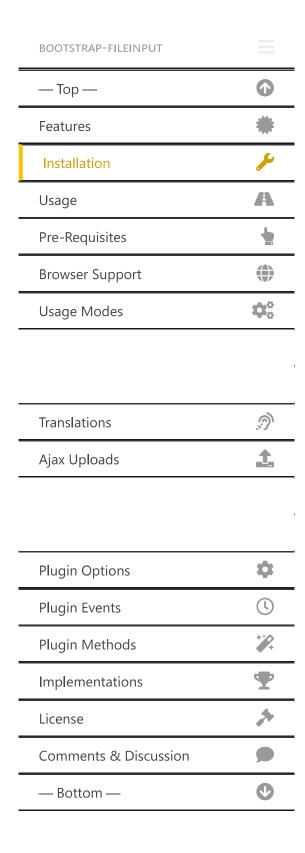
Not seeing the updated content on this page! Hard refresh your browser to clean cache for this page (e.g. SHIFT-F5 on Windows Chrome)



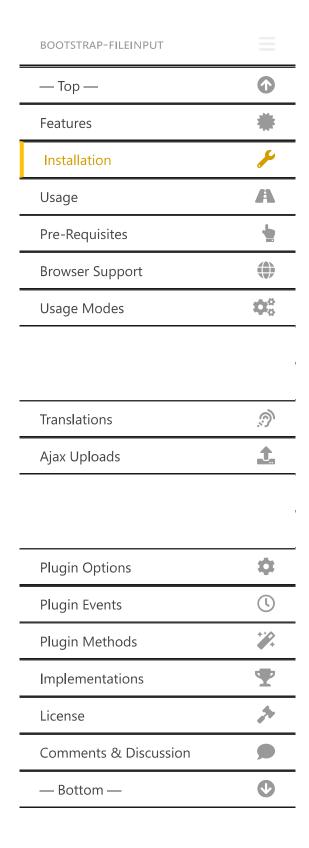
Features

File Input Features

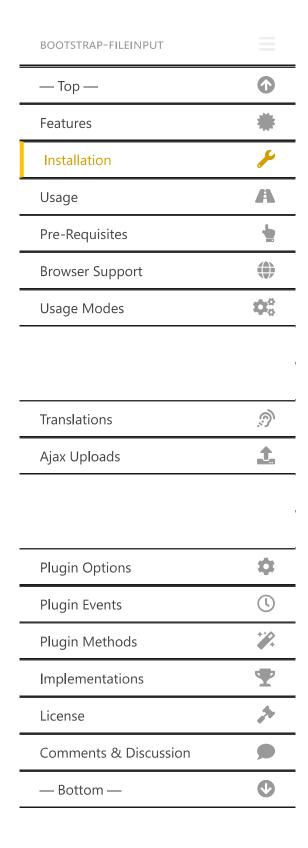
- The plugin will convert a simple HTML file input to an advanced file picker control. Will help fallback to a normal HTML file input for browsers not supporting JQuery or Javascript.
- The file input consists of the following three sections with options and templates to control the display:
 - file caption section: to display a brief information of the file(s) selected
 - file action buttons section: to browse, remove, and upload files.
 - file preview section: to display the selected files on client for preview (supports preview of image, text, flash, and video file types). Other file types will be displayed as normal thumbnails.
- The plugin automatically converts an input with type =
 file to an advanced file picker input if you set its
 class = file. All options to the input can be passed as
 HTML5 data attributes.
- Ability to select and preview multiple files. Uses HTML
 5 File reader API to read and preview files. Displays the progress of files being being loaded onto the preview zone, in case many files are chosen.
- Ability to copy and paste files or images from the clipboard on to the fileinput plugin (just focus on the file caption name input and paste your clipboard content).



- Offers predefined templates and CSS classes which can be changed to style your file-input display as per your needs.
- Ability to configure the plugin to show an initial preview of images/files with initial caption (more useful for record update scenarios). Refer the initialPreview, initialPreviewConfig, and initialCaption properties in the plugin options section for configuring this.
- Ability to zoom content as a detailed preview. See a slideshow of zoomed content in preview, maximize to borderless or fullscreen preview.
- Ability to sort/rearrange content in the initial preview via drag and drop.
- Ability to theme the widget entirely and control styling and layouts.
- Supports multi language widgets on same page through locales/translations.
- Option to show/hide any or all of the following:
 - caption section
 - preview section
 - upload button
 - remove button
- Customise the location of the target container elements to display the entire plugin, the caption container, the caption text, the preview container, preview image, and preview status.
- For text file previews, autowrap the text to the thumbnail width, and show a wrap indicator link to

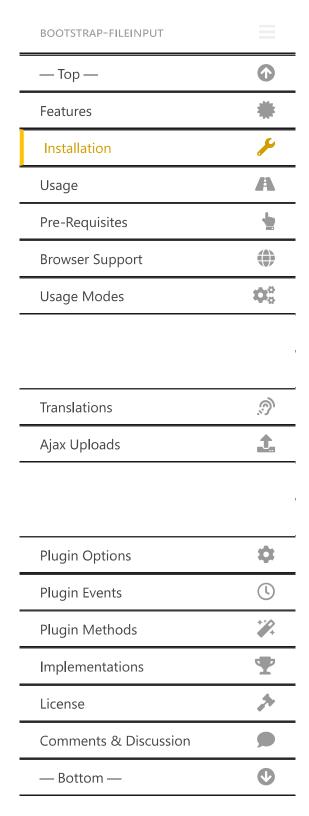


- Customise the messages for preview, progress, and files selected.
- Upload action defaults to form submit. Supports an upload route/server action parameter for custom ajax based upload.
- Triggers JQuery events for advanced development. Events currently available are filereset, fileclear, filecleared, fileloaded, and fileerror.
- Disabled and readonly file input support.
- Dynamically auto size the file captions for long file names exceeding container width.
- Raise new fileimageuploaded event that fires after image is completely loaded on the preview container.
- Autosize preview images when they exceed the size of the preview container.
- Completely templatized and extensible to allow configuration of the file-input the way the developer wants.
- Preview intelligence based on various file preview types. The inbuilt file support types are categorized as image, text, html, video, audio, flash, object, and other.
- allowedPreviewTypes: You can now configure which all file types are allowed to be shown as a preview. This defaults to ['image', 'html', 'text', 'video', 'audio', 'flash', 'object']. Thus all file types are treated as an object to preview by default. For exampleTo preview only image and video, you can set this to ['image', 'video']. To disable content preview for all file-types and show the previewIcon instead as a thumbnail, set this to null, empty, or false.



mime types can be displayed for preview. This defaults to null, meaning all mime types are supported. > NOTE: With release 2.5.0 you can now control which file types or extensions are allowed for upload by setting allowedFileTypes and allowedFileExtensions.

- layoutTemplates: Allows you to configure all layout template settings within one property. The layout objects that can be configured are: main1, main2, preview, caption, and modal.
- previewTemplates: All preview templates for each preview type have been combined into one property, instead of separate templates for image, text etc. The keys are the formats as set in allowedPreviewTypes and values are the templates used for previewing. There are default prebuilt templates for each preview file type (generic, image, text, html, video, audio, flash, object, and other). The generic template is used only for displaying initialPreview content using direct markup.
- previewSettings: Allows you to configure width and height for each preview image type. The plugin has default widths and heights predefined for each type i.e image, text, html, video, audio, flash, and object.
- fileTypeSettings: Allows you to configure and identify each preview file type using a callback. The plugin has default callbacks predefined to identify each type i.e image, text, html, video, audio, flash, and object.
- Replacing tags within templates has been enhanced.
 With this release it will automatically check for multiple occurrences of each tag to replace within a template string.
- Manipulate events and add your own custom validation messages easily by returning output to abort uploads in any of the other events.



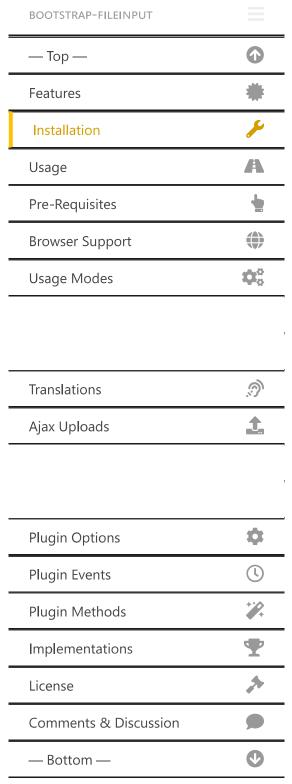
Note

Flash preview will require Shockwave flash to be installed and supported by the client browser. The flash preview currently works successfully with webkit browsers only. Video & Audio formats are however supported by all modern browsers that support the HTML5 video/audio tags. Note that browsers have limited number of video/audio formats supported by the HTML5 video element (e.g. mp4, webm, ogg, mp3, wav). The size of video files are recommended to be small (to be controlled through maxFileSize property) so that it does not affect the preview performance. You can copy a few files from the examples directory of this plugin repo, to test a few examples of flash and video files.

File Upload Features

With release 4.0.0, the plugin now also includes inbuilt support for AJAX Uploads and selectively adding or deleting files. AJAX upload functionality are built upon HTML5 FormData and XMLHttpRequest Level 2 standards. Most modern browsers do support this standard, but the plugin will automatically degrade to normal form based submission for unsupported browsers.

- Add functionality for AJAX based UPLOAD using HTML5 FormData (most modern browsers support it).
 Will degrade to normal Form Based File submission if this is not supported.
- To use AJAX Upload, one must set the <u>uploadUrl</u> property.
- Enhance plugin to now allow files to be added, appended, removed (based on FEEDBACK from many).
 Thus one can append files to preview.

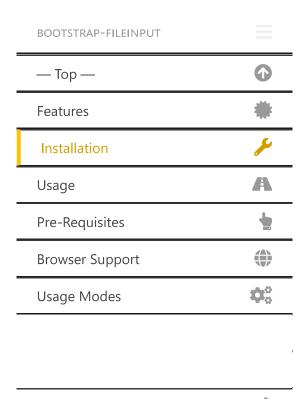


- Delete or upload files one by one OR in batch.
- If showPreview is set to false, or uploadUrl is not supported plugin will degrade to normal form based upload.
- Configurable indicators for file awaiting upload, file successfully uploaded, files errored in upload.
- Ability to add extra form data with ajax based uploads.
- Upload progress bar and individual thumbnail upload indicators.
- Ability to cancel and abort ongoing AJAX uploads.
- Build up initial preview content (e.g. gallery of saved images). You can set initial preview actions (prebuilt support for initial preview delete). Other custom action buttons can be set for initial preview thumbnails as well.
- Ensure plugin is still lean in size and optimized for performance inspite of the above features by optimally utilizing HTML5 & jquery features only.
- Automatically refresh preview with content from server as soon as an ajax upload finishes.

Installation

The bootstrap-fileinput plugin can be installed automatically or manually using one of these options:

Bower Package Manager



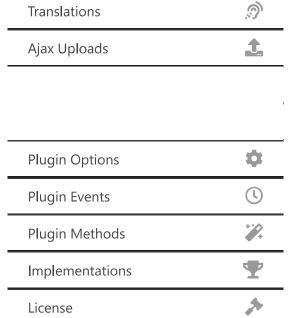
running:

\$ bower install bootstrap-fileinput

Node Package Manager

Installation via NPM is as simple as running:

\$ npm install bootstrap-fileinput



Composer Package Manager

You can install bootstrap-fileinput via composer package manager. Either run:

\$ php composer.phar require kartik-v/bootstrap-fileinput "dev-

or add:

•

"kartik-v/bootstrap-fileinput": "dev-master"

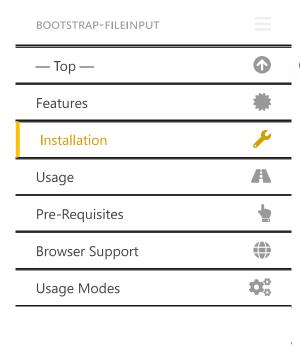
to your composer.json file

Manual Install

You can also manually install the plugin easily to your project. Just download the source ZIP or TAR ball and extract the plugin asset files and folders into your project.

Comments & Discussion

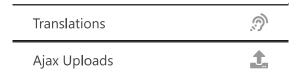
— Bottom —





Note

The plugin will automatically convert fields of [input type="file"] to a file input control, if you attach a css class file to the input. But, if you are initializing the plugin separately via javascript, then DO NOT ATTACH the css class file to the input (as it will result in duplicate initializations and the javascript code maybe skipped).



Usage: Step 1

Plugin Options

Plugin Events

U

Plugin Methods

Implementations

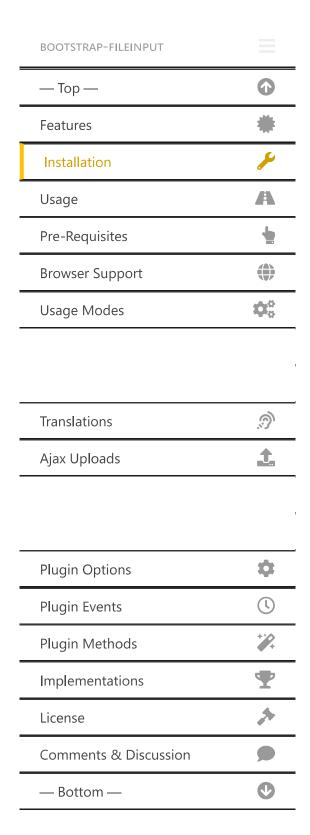
License

Load the following assets in your header.

```
1.
      <!-- bootstrap 5.x or 4.x is supported. You can also us
 2.
      <link rel="stylesheet" href="https://cdn.jsdelivr.net/n</pre>
 3.
4.
      <!-- default icons used in the plugin are from Bootstra
      <link rel="stylesheet" href="https://cdn.jsdelivr.net/n</pre>
 5.
 6.
 7.
      <!-- alternatively you can use the font awesome icon li
      <!-- link rel="stylesheet" href="https://use.fontawesom
8.
9.
10.
      <!-- the fileinput plugin styling CSS file -->
11.
      <link href="https://cdn.jsdelivr.net/gh/kartik-v/bootst"</pre>
12.
      <!-- if using RTL (Right-To-Left) orientation, load the
13.
      <!-- link href="https://cdn.jsdelivr.net/gh/kartik-v/bo
14.
15.
      <!-- the jQuery Library -->
16.
      <script src="https://code.jquery.com/jquery-3.6.0.min.j</pre>
17.
18.
19.
      <!-- buffer.min.js and filetype.min.js are necessary in
```

Comments & Discussion

- Bottom -



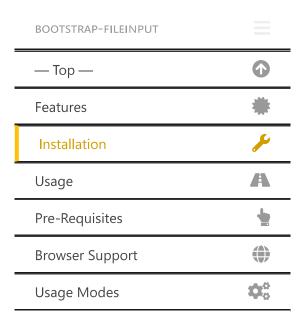
```
23.
           do not load these scripts then the mime type parsi
24.
           and some basic file content parsing signatures. --
25.
      <script src="https://cdn.jsdelivr.net/gh/kartik-v/boots"</pre>
      <script src="https://cdn.jsdelivr.net/gh/kartik-v/boots</pre>
26.
27.
28.
      <!-- piexif.min.js is needed for auto orienting image f
29.
          wish to resize images before upload. This must be l
      <script src="https://cdn.jsdelivr.net/gh/kartik-v/boots</pre>
30.
31.
32.
      <!-- sortable.min.js is only needed if you wish to sort
          This must be loaded before fileinput.min.js -->
33.
34.
      <script src="https://cdn.jsdelivr.net/gh/kartik-v/boots</pre>
35.
      <!-- bootstrap.bundle.min.js below is needed if you wis
36.
          dialog. bootstrap 5.x or 4.x is supported. You can
37.
      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1</pre>
38.
39.
40.
      <!-- the main fileinput plugin script JS file -->
      <script src="https://cdn.jsdelivr.net/gh/kartik-v/boots</pre>
41.
42.
43.
      <!-- following theme script is needed to use the Font A
      <!-- script src="https://cdn.jsdelivr.net/gh/kartik-v/b
44.
45.
      <!-- optionally if you need translation for your langua
46.
      <script src="https://cdn.jsdelivr.net/gh/kartik-v/boots</pre>
```

If you noticed, you need to load the jquery.min.js and bootstrap.min.css in addition to the fileinput.min.css and fileinput.min.js. For enabling the default icons used in the plugin load the Bootstrap 5.x icon library bootstrap.min.css. Alternatively, the Font Awesome theme can be enabled by loading FontAwesome Icons CSS file and its theme file fa5/theme.min.js. The locale file for your language LANG.js can be optionally included for translating for your language if needed. In case you need RTL (Right-To-Left) orientation, then you need to load the fileinput-rtl.min.css stylesheet file after fileinput.min.css

Optional Dependent Plugins

 The piexif.min.js file is the source for the <u>Piexifjs plugin by hMatoba</u>. It is required to be **Translations**

- Bottom -



- orientation tag. This library is also needed for restoring the exif data to the image files when using the image resize feature of the **bootstrapfileinput** plugin.
- The sortable.min.js file is the source for the Sortable plugin by rubaxa. It is required to be loaded before fileinput.min.js if you wish to sort the thumbnails in the initial preview.

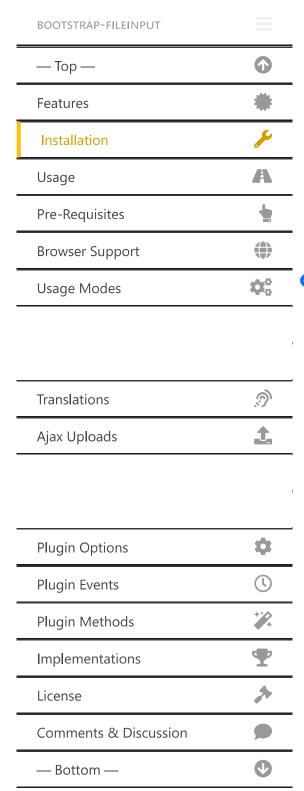
Usage: Step 2a

Initialize the plugin on your page. For example to initialize using javascript - the following code can be placed in document.ready or at any place you wish on your page.

```
// Optional setup: bootstrap library version will be au
1.
 2.
     // loaded bootstrap JS library bootstrap.min.js. But if
     // bootstrap version yourself, then you can set the fol
 3.
     // the plugin init script (available since plugin relea
4.
5.
     $.fn.fileinputBsVersion = "3.3.7"; // if not set, this
 6.
 7.
8.
     // initialize plugin with defaults
     $("#input-id").fileinput();
9.
10.
     // with plugin options
11.
12.
     $("#input-id").fileinput({'showUpload':false, 'previewF
```

The #input-id is the identifier for the input (e.g. type=file) on your page, which is hidden automatically by the plugin.

1 Ajax Uploads **O Plugin Options** (1) Plugin Events Plugin Methods 7 **Implementations** License Comments & Discussion



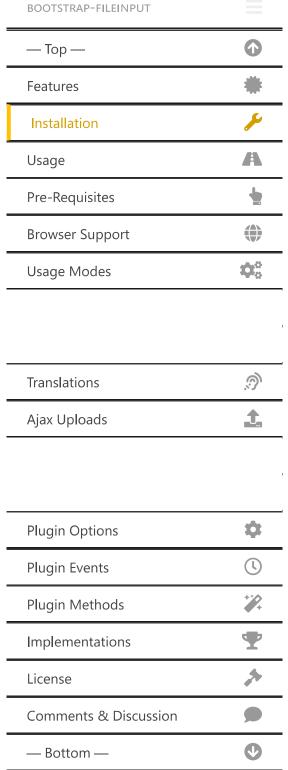
input, through HTML 5 data attributes to your input field. Note that for this case, you need to attach the CSS class file to the input.

As noted earlier, when initializing plugin via javascript (step 2a), you MUST NOT attach the CSS class file to the input.

1. <input id="input-id" type="file" class="file" data-prev

Pre-Requisites

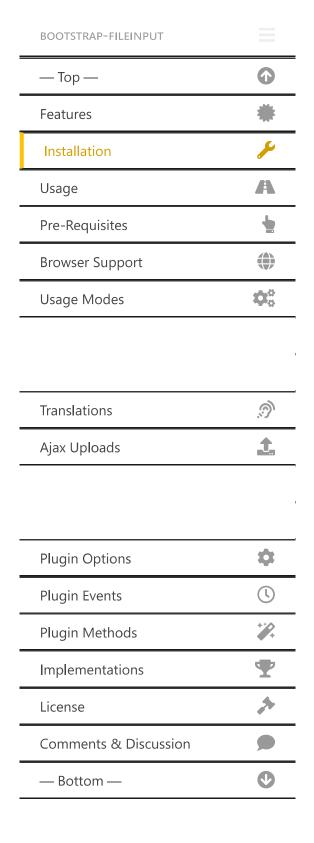
- Bootstrap 5.x or Bootstrap 4.x or Bootstrap 3.x.
 However, the plugin can be customized for any CSS framework using templates. Bootstrap 5.x framework is supported since release v5.5.0. Bootstrap 4.x framework is supported since release v4.4.4.
- Latest <u>JQuery</u>
- Most modern browsers supporting HTML5 file inputs and FileReader API including CSS3 & JQuery. For details, refer the <u>browser support section</u>.
- For file previews to work, the browser must support
 the HTML5 FileReaderAPI else the plugin will autodegrade to a normal file input. For Internet Explorer,
 one must use IE versions 10 and above. IE9 and below
 will work as a normal file input, and will not support
 multiple file selection or the HTML 5 FileReader API.
- With release 4.0, AJAX uploads are supported. AJAX uploads require that the browser support HTML5
 FormData and XHR2 (XMLHttpRequest 2). Most modern browsers support FormData and XHR2. The plugin will automatically degrade to normal form based submission for browsers not supporting AJAX uploads.



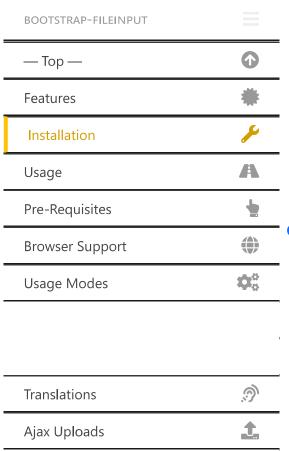
Browser Support

The plugin uses HTML 5 features to achieve various functionalities. Most modern browsers support these features. However, to know if your browser supports these functions you must run through these checks once below.

Functionality	Description	Support
File Input Multiple	Allow users to select multiple files using the native HTML file input.	Browsers
Input File Directory	Uses the webkitdirectory attribute on the <input type="file"/> element. This allows entire directory with file contents (and any subdirectories) to be selected.	Browsers
HTML 5 File API	Allow reading and previewing files on the preview pane using the plugin	Browsers
HTML 5 Blob	Allow managing/constructing file blobs for enabling the resumable upload functionality for the plugin.	Browsers
HTML 5 Blob Slice	Allow slicing file blobs for enabling the file chunks upload	Browsers



HTML 5 XHR2 & FormData	Allow using ajax uploads with ability to append / delete files and track using a progress bar.	Browsers
HTML5 Drag Drop	The plugin uses HTML 5 drag and drop ability to drag and drop files from your PC/device into the file input dropzone. This is supported both for form based and ajax based uploads. However for form based uploads it is restricted to browsers like Chrome and Mozilla that support assigning FileList object to the native file input.	Browsers
HTML5 Canvas	HTML 5 Canvas offers a method of generating fast, dynamic graphics using JavaScript. The plugin uses HTML5 canvas for managing image files via JavaScript. This is required if you wish to resize image files before upload.	Browsers
HTML5 Download Attribute	The download attribute on an hyperlink specifies that the target will be downloaded when a user clicks on the hyperlink. The value of the attribute will be the name of the	Browsers



use the download action button for each thumbnail in a more easy and effective manner.

Usage Modes

Largely speaking, the plugin can be configured in one of the following two different modes for upload. **IMPORTANT**: Do not try to combine the modes below to receive file data as you will receive inconsistent and/or erroneous output.

Mode 1: Form Submission

Plugin Options

Plugin Events

O

Plugin Methods

Implementations

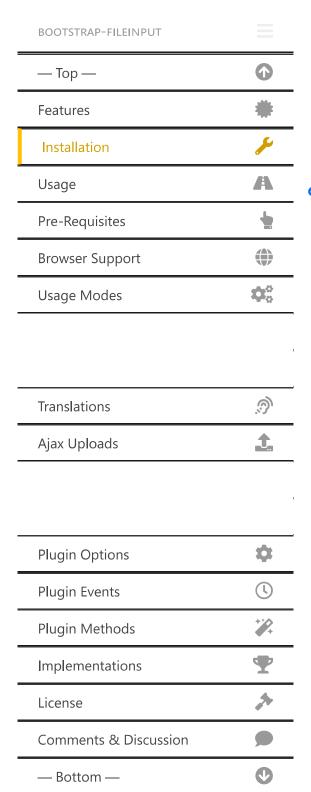
License

Comments & Discussion

— Bottom —

In this mode, you do not set the <u>uploadUrl</u> property. The plugin will use the **native file input** to store files and it can be read after a normal FORM submission (you must envelop the input within a FORM). This is useful for single file uploads OR simpler scenarios of multiple file uploads. Configuration is straightforward as you can read all data POSTED from a native form submission. However, note that the **native file input** is read only and cannot be modified or updated by external code. Especially for multiple file input selections, ONE cannot append files to an already selected file list. If one tries to select files on an already selected file input, it will overwrite and clear the previous selection. Similarly, one cannot selectively remove/delete files that have been added before upload in this mode.

Note



based uploads it is restricted to browsers like Chrome and Mozilla that support assigning FileList object to the native file input.

Mode 2: Ajax Submission

In this mode, you MUST SET the <u>uploadUrl</u> property to a VALID ajax processing server action/URL. If the <u>uploadUrl</u> is set, then the plugin automatically assumes an ajax upload for the scenario. The plugin offers advanced features for ajax submission that is not available in form submission. Features like appending/removing files in preview zone, getting a progress bar for your uploads, image resizing etc. are all possible ONLY IN THIS mode. Your browser must support HTML5 FormData/XHR2 for this to work and your server code that processes the ajax call must return a valid JSON response.

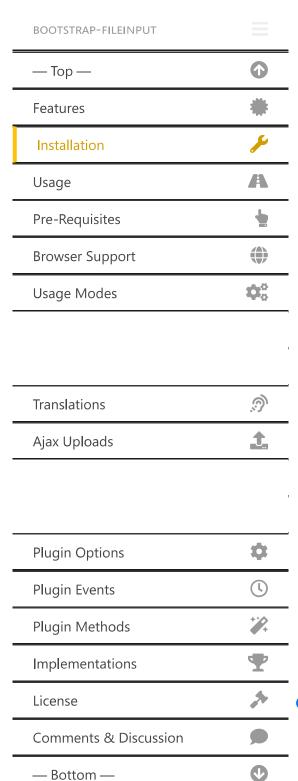
Note

As an advanced scenario, the plugin allows you to process the ajax upload even if there are no files selected but a valid uploadExtraData is sent with the ajax response. The events filebatchpreupload, filebatchuploadsuccess, filebatchuploadcomplete, or filebatchuploaderror will be triggered in this case. It will not have any data for the files selected, but will allow the extra data to be sent.

Modes Compared

BOOTSTRAP-FILEINPUT	
— Тор —	•
Features	*
Installation	<i>></i>
Usage	A
Pre-Requisites	_
Browser Support	*
Usage Modes	⊅ °
Translations	<u>.</u>
Ajax Uploads	1
Plugin Options	*
Plugin Events	U
Plugin Methods	%
Implementations	T
License	<i>A</i>
Comments & Discussion	9
— Bottom —	•

Support single & multiple file upload	~	~
Preview files using HTML 5 FileAPI	~	✓
Read files directly via form submission	~	×
Individual file delete icon for each preview thumbnail	√ <u>1</u>	✓ <u>2</u>
Individual file upload icon for each preview thumbnail	×	✓
Requires valid JSON response back from server	×	✓
Requires browser support for HTML5 FormData/XHR2	×	✓
Server code to process ajax and send JSON Response	×	✓
Drag & Drop Files using drop zone	√ <u>3</u>	~
Ability to append files to already selected list	×	~
Ability to delete files to already selected list	v 1	~
Progress bar for uploads	×	~



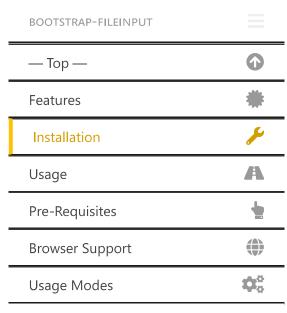
Upload stats for uploads	×	~
Resumable / Chunk uploads	×	~
Read additional form data	Directly via form submit	Via uploadExtraData
Resize image files before upload	×	~

- 1 Via initialPreviewConfig (not applicable for client selected files that are yet to be uploaded).
- 2 For both server uploaded files (via initialPreviewConfig) and client selected files at runtime.
- 3 Drag & drop for form based uploads is supported since release v4.4.8. However for form based uploads it is restricted to browsers like Chrome and Mozilla that support assigning FileList object to the native file input.

Translations

As shown in the usage section, translations are now enabled with release 4.1.8. Follow these steps to enable translations for your language:

 Load your language locale script file on your page from /js/locales/<lang>.js. This script must be loaded after the core fileinput.js file, where <lang> is



new language by submitting a github pull request to add to this folder. Check the sample locale file to copy and create a translation configuration for your own language.

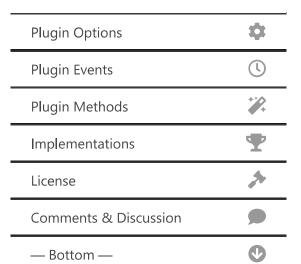
• Configure the plugin's <u>language</u> property to the same language code identified by <lang> above.

Ajax Uploads



You need to setup the server methods to parse and return the right response via AJAX. You can setup uploads in asynchronous OR synchronous modes as described below.

Asynchronous Uploads

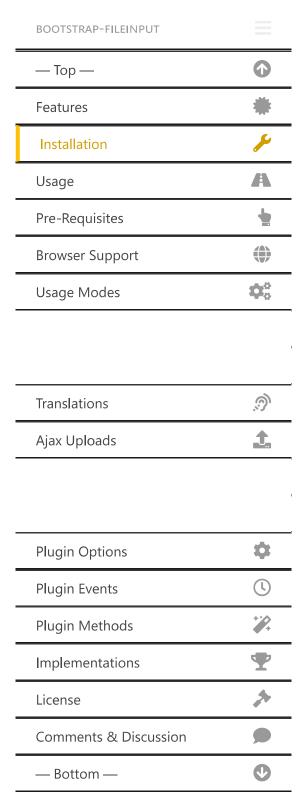


This is the **default mode**, whereby the <u>uploadAsync</u> property is set to true. When uploading multiple files, the asynchronous mode allows triggering parallel server calls for each file upload. You can control the maximum number of files allowed at a time to be uploaded by setting the maxFileCount property. In asynchronous mode, progress of each thumbnail in the preview is validated and updated.

Receiving Data (on server)

Your server method as set in <u>uploadUrl</u> receives the following data from the plugin

• **file data**: The file data is sent to the server in a format very similar to how you would receive data via a native HTML file input (via form submission). For example in



input. If you do not set a name attribute for your input, the name is defaulted to file_data. Note that multiple file uploads require that you set multiple property to true for your input. So in PHP you would receive the file data as \$_FILES['file_data']

extra data: The plugin can send additional data to your server method. This can be done by setting uploadExtraData as an associative array object in key value pairs. So if you have setup uploadExtraData= {id:'kv-1'}, in PHP you can read this data as \$_POST['id'].

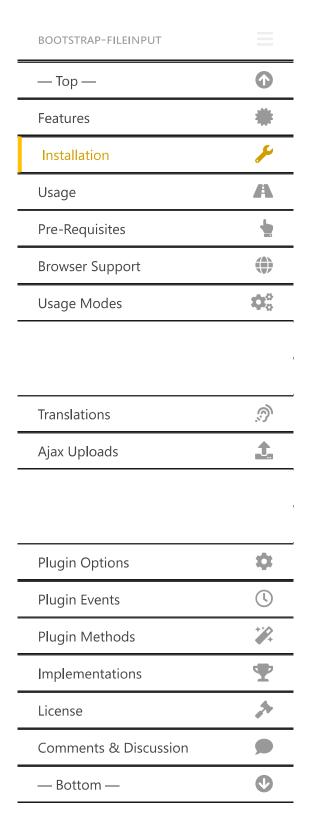
Note

In asynchronous mode, you will ALWAYS receive a single FILE on your server action that processes the ajax upload. Basically the plugin will trigger parallel ajax calls for every file selected for upload. You need to write your server upload logic accordingly so that you always read and upload ONE file. Similarly, in the sending data section below, you must return an initialPreview that reflects data only for the single file received.

Sending Data (from server)

Your server method as set in <u>uploadUrl</u> must send data back as a json encoded object. For example, the server could return a JSON object like below:

```
    // example JSON response from server
    {
    error: 'An error exception message if applicable',
    initialPreview: [
    // initial preview thumbnails for server upload
    ],
    initialPreviewConfig: [
```

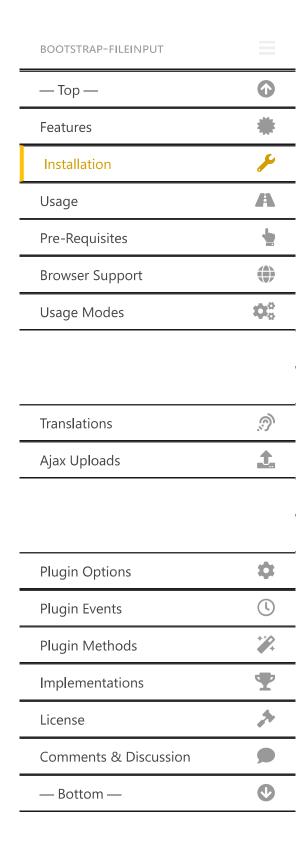


If you have NO DATA to be sent from the server then you MUST at least return an empty JSON object e.g. {}. The JSON object that you return from the server consists of 5 pieces of information. Note that in asynchronous mode, you will ALWAYS receive ONE FILE record from the server - so adjust your code accordingly.

- error: string, which will be the error message for the entire batch upload and will help the plugin to identify error in the file upload. For example the response from server would be sent as {error: 'You are not allowed to upload such a file.'}. Note: The plugin will automatically validate and display ajax exception errors.
- initialPreview: array, the list of image files or any HTML markup to point to your uploaded files. You will always send ONE row in this array because you will always receive ONE file with the upload in asynchronous mode. If this property is set, the plugin will automatically replace the files dynamically in the preview content after upload success. The configuration for this is similar to the initialPreview option setting. For example:

```
    initialPreview: [
    '<img src='/images/desert.jpg' class='file-pre</li>
    ]
```

 initialPreviewConfig: array, the configuration to identify properties for each file markup in initialPreview item (that is setup as part of initialPreview). You will always send ONE row in this



set, the plugin will automatically replace the files dynamically in the preview content after upload success. The configuration for this is similar to the initialPreviewConfig option setting. For example:

```
initialPreviewConfig: [
1.
2.
         {
3.
              caption: 'desert.jpg',
             width: '120px',
4.
              url: 'http://localhost/avatar/delete', //
5.
6.
              key: 100,
7.
              extra: {id: 100}
8.
         }
     ]
9.
```

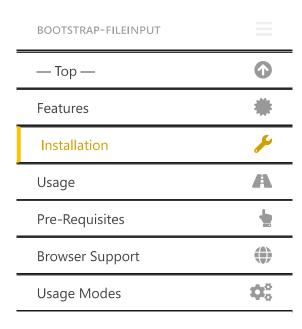
 initialPreviewThumbTags: array, an array of objects corresponding to replacing tags within each initial preview thumbnail. The initial preview thumbnails set via initialPreview will read this configuration for replacing tags.

```
1.
     // change thumbnail footer template
2.
     // set initial preview template tags
3.
     initialPreviewThumbTags:[
         {
4.
5.
              '{CUSTOM_TAG_NEW}': ' ',
              '{CUSTOM_TAG_INIT}': '<span class=\'custom
6.
7.
         }
8.
     ]
```

 append: boolean, whether to append the content to the initialPreview if you already set an initialPreview on INIT. If not set this defaults to true. If set to false, the plugin will overwrite the initialPreview content.

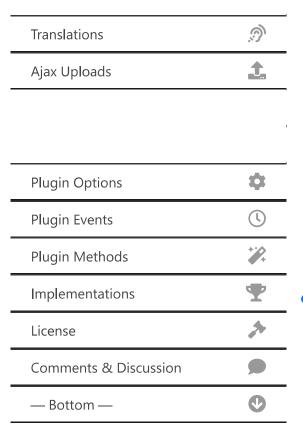
IMPORTANT

 You MUST send a valid JSON response from your server, else the upload process will fail. Even if you do



- To trap and display a validation error, your JSON response data must include the error key, whose value will be the error HTML markup to display. This is to be setup as mentioned above.
- You can also send in additional keys or data with your JSON response, for you to process them for advanced cases using events like fileuploaded.

Synchronous Uploads

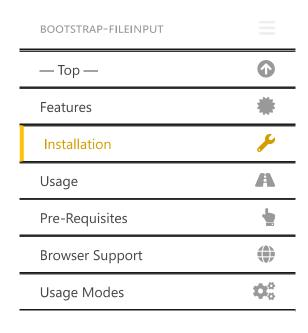


In this mode, the <u>uploadAsync</u> property is set to false. This will trigger just one batch upload call to the server and send files from client to server as an array object. Even in this mode, you can control the maximum number of files allowed at a time to be uploaded by setting the maxFileCount property. However, in synchronous mode, progress will be only at a overall level. Progress of each thumbnail in the preview is not exactly validated and updated. However, the plugin offers you a method of displaying upload errors encountered for each file.

Receiving Data (on server)

Your server method as set in <u>uploadUrl</u> receives the following data from the plugin

• **file data**: The file data is sent to the server in a format very similar to how you would receive data via a native HTML file input (via form submission). For example in PHP you can read this data as \$_FILES['input-name'], where input-name is the name attribute of your file input. Also as in asynchronous mode before, if you do not set a name attribute for your input, the name is



addition to setting multiple property to true. If you do not set your input name as an array format, you would receive only the first file on your server. In PHP you would receive the file data as \$_FILES['input-name'], which will be an array of file objects.

extra data: The plugin can send additional data to your server method. This can be done by setting uploadExtraData as an associative array object in key value pairs. So if you have setup uploadExtraData= {id:'kv-1'}, in PHP you can read this data as \$_POST['id'].



Sending Data (from server)

YIn synchronous mode as well, the <u>uploadUrl</u> must send data back as a json encoded object. For example the server could return a JSON object like below:

```
Plugin Options

Plugin Events

U

Plugin Methods

Implementations

License

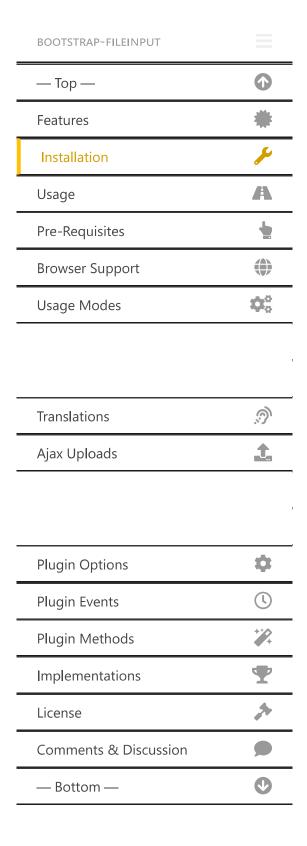
Comments & Discussion

— Bottom—

•
```

```
// example JSON response from server
 1.
 2.
 3.
          error: 'An error exception message if applicable',
          errorkeys: [], // array of thumbnail keys/identifie
 4.
          initialPreview: [
 5.
          ], // initial preview configuration
 6.
          initialPreviewConfig: [
 7.
              // initial preview configuration if you directl
 8.
 9.
          1,
          initialPreviewThumbTags: [
10.
              // initial preview thumbnail tags configuration
11.
12.
13.
          append: true // whether to append content to the in
     }
14.
```

If you have NO DATA to be sent from the server then you MUST at least return an empty JSON object e.g. {}. The JSON object that you return from the server needs to send these 6 pieces of information.



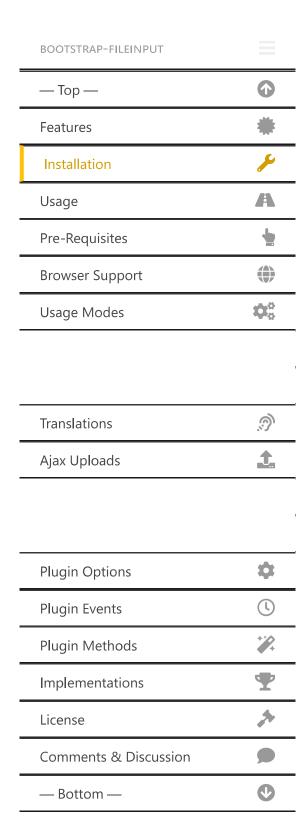
error in the file upload.

- errorkeys: array, the keys (zero-based indexes for the file data received) for the files that have errored out.
 Based on this data, the plugin will automatically set the thumbnails and each individual preview file to error out.
- initialPreview: array, the list of image files or any HTML markup to point to your uploaded files. If this property is set, the plugin will automatically replace the files dynamically in the preview content after upload success. The configuration for this is similar to the <u>initialPreview</u> option setting. For example:

```
    initialPreview: [
    '<img src='/images/desert.jpg' class='file-pre</li>
    '<img src='/images/jellyfish.jpg' class='file-</li>
    ]
```

 initialPreviewConfig: array, the configuration to identify properties for each file markup in initialPreview item (that is setup as part of initialPreview). If this property is set, the plugin will automatically replace the files dynamically in the preview content after upload success. The configuration for this is similar to the initialPreviewConfig option setting. For example:

```
initialPreviewConfig: [
1.
          {
2.
               caption: 'desert.jpg',
3.
              width: '120px',
4.
               url: 'http://localhost/avatar/delete', //
5.
 6.
               key: 100,
7.
               extra: {id: 100}
8.
          },
9.
          {
               caption: 'jellyfish.jpg',
10.
              width: '120px',
11.
```



```
15. return {id: $('#id').val()};
16. },
17. }
18. ]
```

 initialPreviewThumbTags: array, an array of objects corresponding to replacing tags within each initial preview thumbnail. The initial preview thumbnails set via initialPreview will read this configuration for replacing tags.

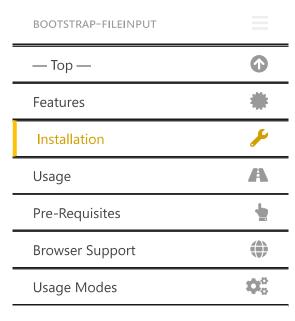
```
// change thumbnail footer template
1.
2.
      // set initial preview template tags
      initialPreviewThumbTags:[
3.
4.
          {
              '{CUSTOM_TAG_NEW}': ' ',
5.
              '{CUSTOM_TAG_INIT}': '<span class=\'custom
6.
7.
          },
8.
          {
9.
              '{CUSTOM_TAG_NEW}': ' ',
               '{CUSTOM_TAG_INIT}': '<span class=\'custom
10.
          }
11.
      ]
12.
```

 append: boolean, whether to append the content to the initialPreview if you already set an initialPreview on INIT. If not set this defaults to true. If set to false, the plugin will overwrite the initialPreview content.

For example the response from server would be sent as {error: 'You have faced errors in 4 files.', errorkeys: [0, 3, 4, 5]}. **Note:** The plugin will automatically validate and display ajax exception errors.

IMPORTANT

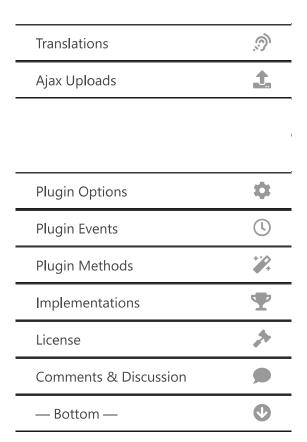
 You MUST send a valid JSON response from your server, else the upload process will fail. Even if you do not encounter any error, you must at least send an empty JSON object {} from your server.



will be the error HTML markup to display. In addition, you must typically also send the errorkeys for synchronous mode to identify the keys for files which faced errors. This is to be setup as mentioned above.

 You can also send in additional keys or data with your JSON response, for you to process them for advanced cases using events like filebatchuploadsuccess.

Resumable Uploads

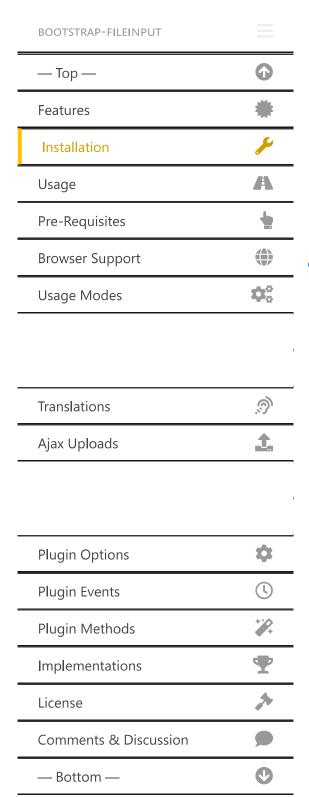


With release v5.0, the plugin supports resumable and chunk uploads. This is a special ajax upload mode that can be activated by setting the enableResumableUpload
property to true along with a valid uploadUrl. When resumable uploads are enabled - the plugin sends files by splitting it into chunks as determined by the properties within resumableUploadOptions. This upload mode is a special variant with additional capabilites than the synchronous and asynchronous uploads. Note that the files selected are sent in a sequence ONE at a time to uploadUrl. You can configure additional data to be sent for upload via uploadExtraData. This typically is useful to send an upload token for security authorizations.

Refer the <u>resumable uploads demos</u> for understanding few scenarios of setting up resumable and chunk uploads using the bootstrap-fileinput plugin.

NOTE

For resumable uploads (when enableResumableUpload property to true), the thumbnail specific upload button will not be displayed. The uploads will be controlled at the

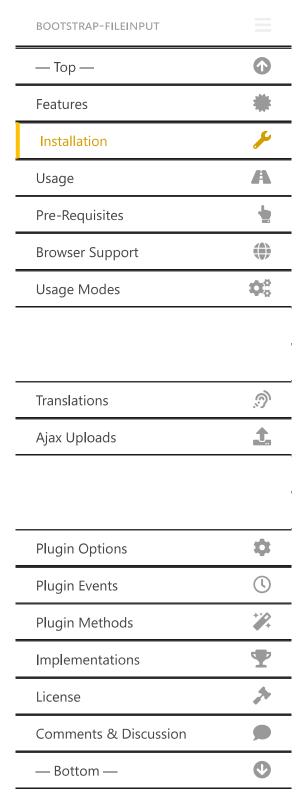


breaking files into necessary chunks. One therefore cannot upload a specific file thumbnail in this mode, when enableResumableUpload property to true. The plugin will automatically upload the batch of files in a serial sequence in this mode (resuming paused / broken uploads where required).

Receiving Data (on server)

Your server method as set in <u>uploadUrl</u> receives the following parameters as data from the plugin for resumable uploads. The plugin by default sends this information as a JSON object via <u>POST</u>. You can configure these parameter names via <u>uploadParamNames</u>.

- fileId: string, unique file identifier for the file being uploaded generated via generateFileId
- fileBlob: Blob, the actual file blob chunk sliced as per the chunk size (for example in PHP you would receive this as \$_FILES['fileBlob']).
- fileName: string, name of the file being uploaded
- fileSize: double, size of the file being uploaded in bytes
- fileRelativePath: string, client relative path of the file being uploaded
- chunkIndex: string, index number of the current chunk
- chunkSize: double, size of each file chunk being uploaded
- chunkSizeStart: *double*, chunk start size for this current blob



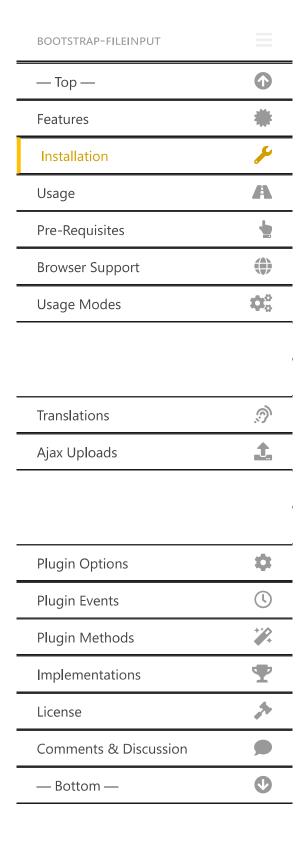
 retryCount: integer, total number of upload retries done so far for this chunk

In addition all the data which you set via <u>uploadExtraData</u> as key value pairs is also received by the server as part of the JSON response. This typically is useful to parse an upload token for security authorizations to detect the right upload user with right accesses.

Sending Data (from server)

Your server method as set in <u>uploadUrl</u> must return back the following response (in JSON FORMAT) to the plugin for resumable uploads. Not unlike async uploads you cannot just send an empty object. Send back the chunkIndex to identify success and/or the error to identify an error.

- chunkIndex: string, index number of the processed chunk (typically you must return back the same chunkIndex you received via POST).
- error: string, which will be the error message for the specific file being uploaded and will help the plugin to identify error in the file upload. For example the response from server would be sent as {error: 'You are not allowed to upload such a file.'}. Note: The plugin will automatically validate and display ajax exception errors.
- initialPreview: array, the list of image files or any HTML markup to point to your uploaded files. You will always send ONE row in this array because you will always receive ONE file with the upload in asynchronous mode. If this property is set, the plugin will automatically replace the files dynamically in the preview content after upload success. The configuration for this is similar to the initialPreview



following for example:

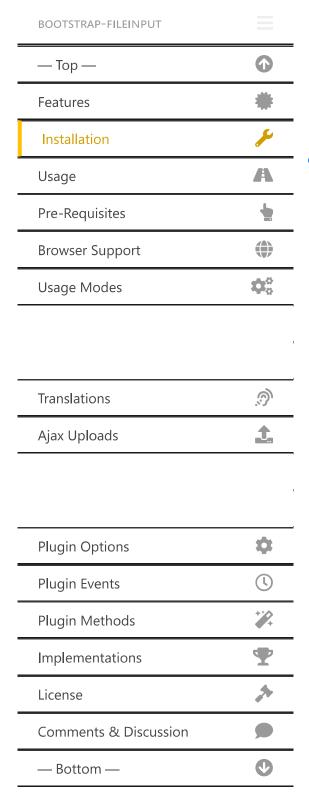
```
1. initialPreview: ['http://localhost/uploads/images/
```

• initialPreviewConfig: array, the configuration to identify properties for each file markup in initialPreview item (that is setup as part of initialPreview). You will always send ONE row in this array - because you will always receive ONE file with the upload in asynchronous mode. If this property is set, the plugin will automatically replace the files dynamically in the preview content after upload success. The configuration for this is similar to the initialPreviewConfig option setting. For example:

```
1.
     initialPreviewConfig: [
2.
3.
              caption: 'desert.jpg',
4.
             width: '120px',
5.
              url: 'http://localhost/avatar/delete', //
              key: 100,
6.
              extra: {id: 100}
7.
         }
8.
9.
     ]
```

 initialPreviewThumbTags: array, an array of objects corresponding to replacing tags within each initial preview thumbnail. The initial preview thumbnails set via initialPreview will read this configuration for replacing tags.

```
// change thumbnail footer template
1.
2.
     // set initial preview template tags
     initialPreviewThumbTags:[
3.
4.
         {
              '{CUSTOM_TAG_NEW}': ' ',
5.
              '{CUSTOM_TAG_INIT}': '<span class=\'custom
6.
7.
         }
8.
     ]
```



initialPreview if you already set an initialPreview on INIT. If not set this defaults to true. If set to false, the plugin will overwrite the initialPreview content.

Oclient Code Example (HTML / JS)

```
<input type="file" id="input-100" name="input-100[]" ac</pre>
 1.
 2.
      <script>
      $('document').on('ready', function() {
 3.
          $("#input-id").fileinput({
4.
              uploadUrl: "http://localhost/file-upload.php",
 5.
              enableResumableUpload: true,
 6.
              resumableUploadOptions: {
 7.
                 // uncomment below if you wish to test the f
 8.
                 // to the server and resume uploads from tha
 9.
                 // testUrl: "http://localhost/test-upload.ph
10.
              },
11.
12.
              uploadExtraData: {
                  'uploadToken': 'SOME-TOKEN', // for access
13.
14.
              },
              maxFileCount: 5,
15.
              allowedFileTypes: ['image'],
                                               // allow only i
16.
17.
              showCancel: true,
              initialPreviewAsData: true,
18.
19.
              overwriteInitial: false,
              // initialPreview: [],
                                               // if you have
20.
              // initialPreviewConfig: [],
21.
                                               // if you have
              theme: 'fa5',
22.
              deleteUrl: "http://localhost/file-delete.php"
23.
          }).on('fileuploaded', function(event, previewId, in
24.
              console.log('File Uploaded', 'ID: ' + fileId +
25.
          }).on('fileuploaderror', function(event, data, msg)
26.
              console.log('File Upload Error', 'ID: ' + data.
27.
          }).on('filebatchuploadcomplete', function(event, pr
28.
              console.log('File Batch Uploaded', preview, con
29.
30.
          });
31.
     });
32.
      </script>
```

Server Code Example (PHP)

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

21.

22.23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

38.

39.

40.

41.

42.

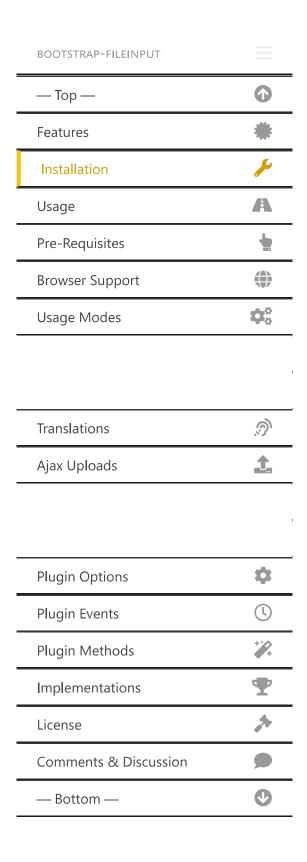
43.

44. 45.

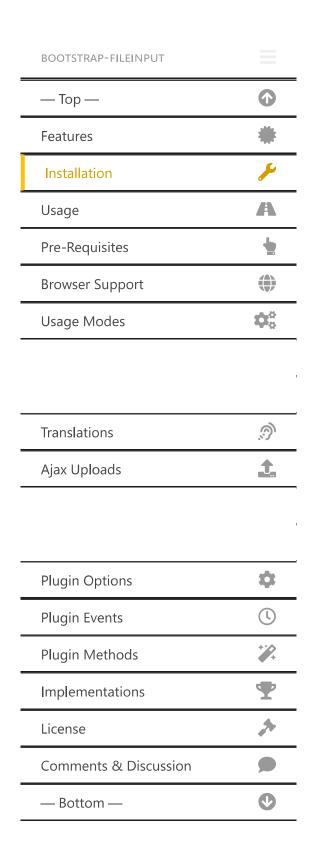
46.

47.

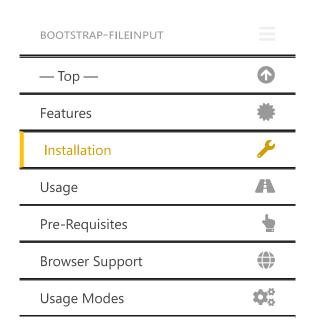
48.



```
header('Content-Type: application/json'); // set json r
$outData = upload(); // a function to upload the bootst
echo json_encode($outData); // return json data
exit(); // terminate
// main upload function used above
// upload the bootstrap-fileinput files
// returns associative array
function upload() {
    $preview = $config = $errors = [];}
    $targetDir = '/webroot/uploads';
    if (!file_exists($targetDir)) {
        @mkdir($targetDir);
    }
    $fileBlob = 'fileBlob';
                                                  // the
    if (isset($_FILES[$fileBlob]) && isset($_POST['uplo
        $token = $_POST['uploadToken'];
                                                  // get
        if (!validateToken($token)) {
                                                  // you
            return [
                'error' => 'Access not allowed'
                                                  // ret
            ];
        }
        $file = $_FILES[$fileBlob]['tmp_name'];
                                                 // the
        $fileName = $_POST['fileName'];
                                                  // you
        $fileSize = $ POST['fileSize'];
                                                  // you
        $fileId = $ POST['fileId'];
                                                  // you
        $index = $_POST['chunkIndex'];
                                                  // the
        $totalChunks = $ POST['chunkCount'];
                                                  // the
        $targetFile = $targetDir.'/'.$fileName;
                                                  // you
        if ($totalChunks > 1) {
                                                  // cre
            $targetFile .= '_' . str_pad($index, 4, '0'
        }
        $thumbnail = 'unknown.jpg';
        if(move_uploaded_file($file, $targetFile)) {
            // get list of all chunks uploaded so far t
            $chunks = glob("{$targetDir}/{$fileName}_*"
            // check uploaded chunks so far (do not com
            $allChunksUploaded = $totalChunks > 1 && co
            if ($allChunksUploaded) {
                                                 // all
                $outFile = $targetDir.'/'.$fileName;
                // combines all file chunks to one file
                combineChunks($chunks, $outFile);
            // if you wish to generate a thumbnail imag
            $targetUrl = getThumbnailUrl($path, $fileNa
            // separate link for the full blown image f
```



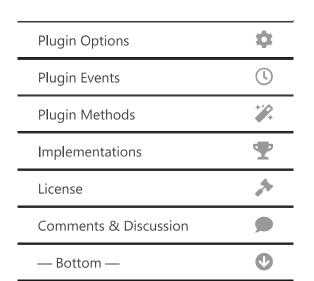
```
CHURKTHUEX -/ PIHUEX,
э⊥.
52.
                       'initialPreview' => $targetUrl, // the
53.
                       'initialPreviewConfig' => [
                           [
54.
                                'type' => 'image',
                                                         // chec
55.
56.
                                'caption' => $fileName, // capt
                                'key' => $fileId,
57.
                                                         // keys
                                'fileId' => $fileId,
                                                         // file
58.
59.
                               'size' => $fileSize,
                                                         // file
                                'zoomData' => $zoomUrl, // sepa
60.
                           ]
61.
62.
                       ],
                       'append' => true
63.
                  ];
64.
              } else {
65.
66.
                       'error' => 'Error uploading chunk ' . $
67.
68.
                  ];
              }
69.
70.
71.
          return [
72.
              'error' => 'No file found'
73.
          ];
74.
     }
75.
76.
      // combine all chunks
77.
      // no exception handling included here - you may wish t
      function combineChunks($chunks, $targetFile) {
78.
79.
          // open target file handle
          $handle = fopen($targetFile, 'a+');
80.
81.
          foreach ($chunks as $file) {
82.
              fwrite($handle, file_get_contents($file));
83.
84.
          }
85.
          // you may need to do some checks to see if file
86.
          // is matching the original (e.g. by comparing file
87.
88.
          // after all are done delete the chunks
89.
90.
          foreach ($chunks as $file) {
              @unlink($file);
91.
          }
92.
93.
94.
          // close the file handle
          fclose($handle);
95.
      }
96.
97.
```



```
// ussuming this is an image file or viaeo file
101.
           // generate a compressed smaller version of the fil
102.
           // here and return the status
           $sourceFile = $path . '/' . $fileName;
103.
104.
           $targetFile = $path . '/thumbs/' . $fileName;
105.
           // generateThumbnail: method to generate thumbnail
106.
107.
           // using $sourceFile and $targetFile
108.
109.
           if (generateThumbnail($sourceFile, $targetFile) ===
110.
               return 'http://localhost/uploads/thumbs/' . $fi
111.
           } else {
112.
               return 'http://localhost/uploads/' . $fileName;
113.
           }
       }
114.
```



Options



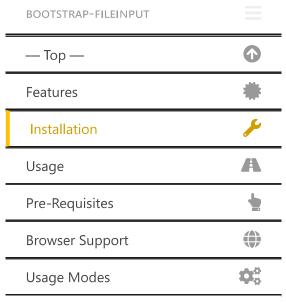
Most of the plugin options can be retrieved at runtime by the following method.

```
    var plugin = $('#file-input).data('fileinput');
    console.log(plugin.initialPreview); // get initialPreview)
```

The plugin supports the options as described in the link below.

View Plugin Options

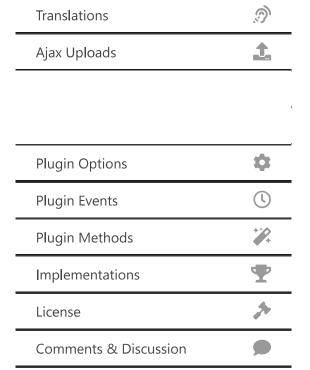






View Plugin Methods

Implementations



The plugin has been implemented as extensions in the following frameworks

SI.	Framework	Extension Source	Developed By	Extension Demo
1.	<u>Yii</u> <u>Framework</u> <u>2.0</u>	<u>Yii2</u> <u>Widgets</u>	<u>Krajee</u>	<u>Yii2 File</u> <u>Input</u>

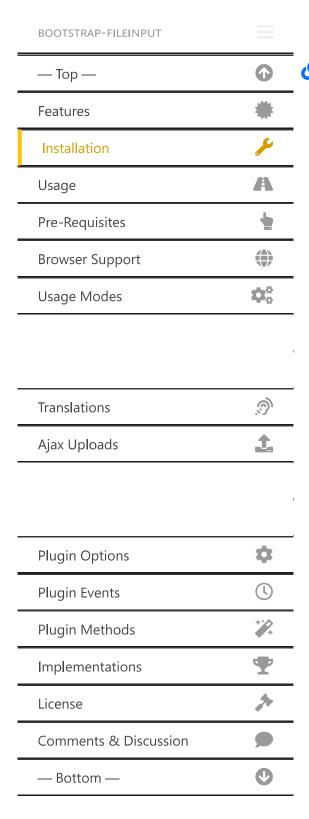
Do you know any other framework extension using this plugin which is not listed here? <u>Tell us</u> so that we can consider its inclusion in this list.



•

bootstrap-fileinput is released under the BSD 3-Clause License. See the bundled <u>LICENSE.md</u> for details.

- Bottom -

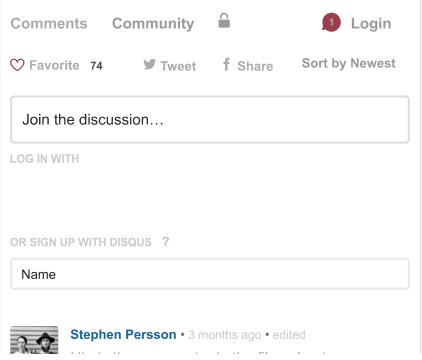


Comments & Discussion

Note

You can now visit the <u>Krajee Webtips Q & A forum</u> for searching OR asking questions OR helping programmers with answers on these extensions and plugins. For asking a question <u>click here</u>. Select the appropriate **question category** (i.e. *Krajee Plugins*) and choose this current page plugin in the **question related to** field.

The comments and discussion section below are intended for generic discussions or feedback for this plugin. Developers may not be able to search or lookup here specific questions or tips on usage for this plugin.



BOOTSTRAP-FILEINPUT	=
— Тор —	•
Features	*
Installation	م
Usage	A
Pre-Requisites	•
Browser Support	(
Usage Modes	₽ °
Ajax Uploads	1
Plugin Options	\$
Plugin Events	(L)
Plugin Methods	***
Implementations	T
License	A
Comments & Discussion	9
— Bottom —	•

1 1 5

files then think its done, but never actually click the upload button. Can we make it do the transfer as soon as they have browsed or dropped files?

Update, sorry - found an example in the AJAX scenarios....

see more



startx25 • 7 months ago

Hi all, is there any option to show "date" attribute in preview mode? i can see size and name but not date info of files



Witter • 8 months ago

Is there an easy way to edit captions for each file?



Alejandro Ormazabal • 8 months ago • edited

Hey, I have a PROBLEM, i upload a photo selected from gallery of my phone, and the exifdata (Lat and Lng) is deleted from photo, only from mobile browsers, not in dektop browsers, i need help!!!!

BOOTSTRAP-FILEINPUT	=
— Тор —	•
Features	*
Installation	₽
Usage	A
Pre-Requisites	_
Browser Support	*
Usage Modes	Φ°
Translations	<u></u>
Ajax Uploads	1
Plugin Options	•
Plugin Events	U
Plugin Methods	***
Implementations	7
License	*
Comments & Discussion	9
— Bottom —	•



Reply • Share >



Karthik L • 8 months ago

how to click the plugin upload button from another button? That means when i click the submit button in the form that wants to trigger the plugin inbuild upload button. Please anyone help me.

see more

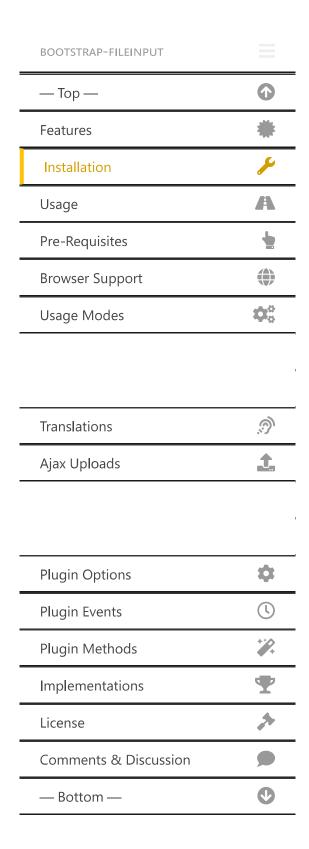
Reply • Share >



Ballu Ikka • 10 months ago

Hey can anyone please help me, in using this plugin (file input) through java.

Like i want to store the input taken from this to the mysql database through java jdbc ,and i don't know javascript to get the input from the input code.....



see more

Reply • Share >



daniel abadi ghadim • 10 months ago

I have problem to use initialPreview from upload Ajax. I return two items from the upload request, but just the first item will be shown.

twig:

```
<div class="file-loading">
<input id="input-id" name="input-res-1" type="file"
class="file" multiple="" data-browse-on-zone-
click="true">
</div>
{% set uploadUrl =
   path('mytest_property_test_upload') %}

<script>
let inputEl = $("#input-id");
inputEl.fileinput(
{
```

see more

Reply • Share >

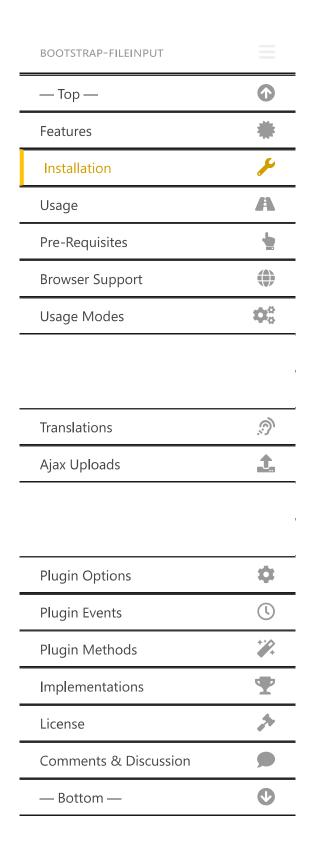


Ballu Ikka Adaniel abadi ghadim

• 10 months ago

Hey can anyone you help me, in using this plugin (file input) through java.

Like i want to store the input taken from this to the mysql database through java jdbc ,and i don't know javascript to get the input from the input code.....



see more

Reply • Share >



Mike Spadaro • 10 months ago

Hello,

I am using your resumable uploads functionality. I have implemented the front end code which seems to be working. I copied your example file-upload.php file, and when it hits the function call move_uploaded_file() it returns false. So on the front end I get the error, "Error uploading chunk 0". I am not sure what has to be changed in the two parameters for this to work. Where is this function located and are the default parameters your set up in the example not actually working?

Thanks.

see more

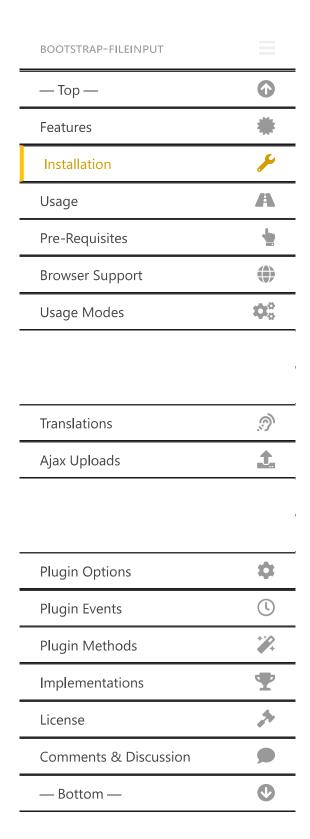
Reply • Share >



Ballu Ikka Mike Spadaro • 10 months ago

Hey can you please help me, in using this plugin (file input) through java.

Like i want to store the input taken from this to the mysql database through java jdbc ,and i don't know javascript to get the input from the input code.....





Mark Bagnall • a year ago

Is it possible to limit how many are processed simultaneously without limiting the number of files uploaded



Adem Natural • a year ago • edited

anybody help me?
in my case, a want set data dinamic initialPreview
im used framework CI
in my controller

```
$data = array(
'nama_user'=>$this->session->userdata('nama_user'),
'ambil_gambar'=>$this->M_setting->getSlides1(),
);
```

in my view

```
$(document).ready(function () {
filename);
//$filenameurl=str_replace("'\'","", $filenameurl);
$a[]="<img src="".$filenameurl."" class="file-
preview-image" alt="Desert" title="Desert">";
```

see more

```
2 ^ V • Reply • Share >
```



Umer Majeed • a year ago

Hi,

I am using he file-input plugin in my theme. I am working on the Edit form where I have displayed the image preview from my server and I want to upload same uploaded image again with form data, Please help me to finish this or let me know if this is possible in file-input plugin?

BOOTSTRAP-FILEINPUT	
— Тор —	•
Features	*
Installation	₽
Usage	A
Pre-Requisites	•
Browser Support	(
Usage Modes	O.
Translations	
Ajax Uploads	1
Plugin Options	*
Plugin Events	(L)
Plugin Methods	%
Implementations	T
License	*
Comments & Discussion	9
— Bottom —	•

see more

Reply • Share >



Ballu lkka → Umer Majeed • 10 months ago

Hey can anyone please help me, in using this plugin (file input) through java. Like i want to store the input taken from this to the mysql database through java jdbc, and i don't know javascript to get the input from the input code.....

see more

Reply • Share >



Sooraj EK • a year ago • edited

Hello,

<form action="process.php" method="post"</pre>

enctype="multipart/form-data">

<input id="krajee" type="file" name="file[]" multiple="true">

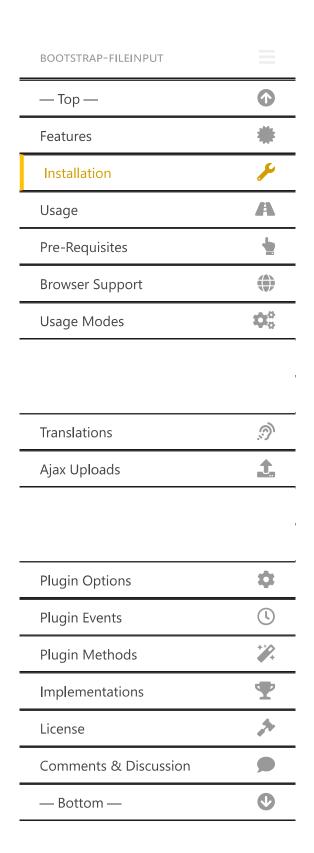
<input type="submit" name="submit" id="submit">

\$("#detailimages").fileinput({

theme: 'fa',

uploadUrl: "#",

allowedFileExtensions: ['ipg'.'ipeg'. 'png'. 'gif'].



maxFileCount: 10, maxFileSize:5000, showUpload: false,

see more

Reply • Share >



Prashant Channe • a year ago • edited Hello,

I am able to upload and download docx file, but unable to preview docx file in thumbnail. I want my application user to be able to preview docx file. Also need support for DWG file download

see more

Reply • Share



Kartik V Mod → Prashant Channe • a year ago • edited

This can be achieved with a specific configuration but you need some understanding on why this is not that straightforward. The docx preview require an external viewer and not supported by native HTML. The plugin uses the free Office 365 viewer to view all MS Office files - but for that you need to upload the file to a internet cloud accessible location first. So to achieve your use case - you would need to immediately upload the file to the server on selection (maybe to temporary location) and return that file URL as initialPreview

BOOTSTRAP-FILEINPUT	=
— Тор —	•
Features	*
Installation	<i>></i>
Usage	A
Pre-Requisites	<u> </u>
Browser Support	•
Usage Modes	₽ °
Translations	,A)
Ajax Uploads	1
Plugin Options	\$
Plugin Events	(L)
Plugin Methods	%
Implementations	7
License	*
Comments & Discussion	
— Bottom —	•

based on your app specific confirmation inputs OR deleting it.

see more

Reply • Share >



Alex • a year ago

Hello everyone. I am using fileinput to load pdf files into a database. This is done in a form where there is still other data. It is necessary to edit in the same way as I can from the byte [] database cause the file to be displayed in fileinput ?? And if necessary, then its replacement is in progress, and the form is preserved completely.

see more

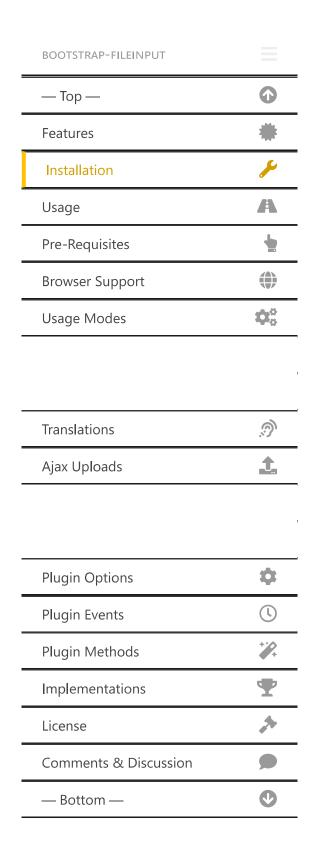
Reply • Share >



Stig Kølbæk • a year ago • edited

I need to use Bootstrap Input File with FreeASPUpload (Yes, classic ASP) and my idea is to trigger the Bootstrap Input File form with action="freeaspupload.asp" and target an IFRAME and upload the file.. but I have 3 questions:

- 1 : Is this actually the best way, or is there another?
- 2 : if it is the best way, how do I from Bootstrap File Input trigger a submit when upload button(s) are clicked?
- 3 : How can I from the FreeASPUpload page trigger a callback to Bootstrap Input File on error/success?



SyntaxFrror: Unexpected end of JSON input

see more

Reply • Share >



Eligio Dzul Zavala • a year ago

i have a problem when deleting, I get this SyntaxError: JSON.parse: unexpected end of data at line 1 column 1 of the JSON data

this is my code:

see more

2 ^ V • Reply • Share >



Kartik V Mod → Eligio Dzul Zavala • a year ago • edited

Check your plugin code and refer the docs on usage refer this webtip Q&A for details on debugging your specific error

Reply • Share >

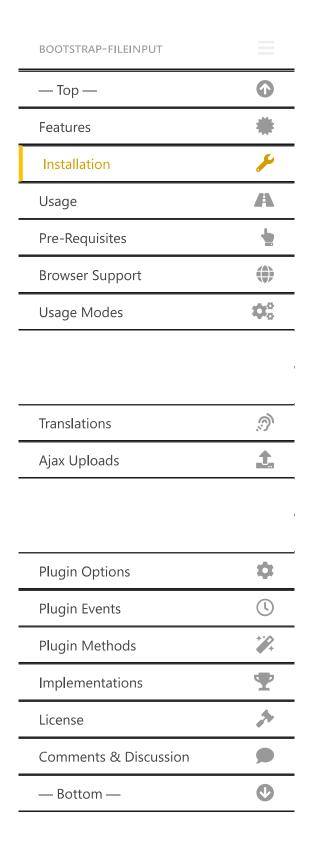


Hemadri Chavada • a year ago • edited

i want to change exist fileinput's option maxFileCount dynamically after upload files.

Can any one help me to sort it out?

Actually i want to get all images which are uploaded every time, which is not replaced by last uploaded file without ajax.





∧ V • Reply • Share >



Cristhofer Alencar • a year ago

Hello, first of all, great plugin. So great that sometimes I feel the need for some kind of course that covers every bit of this excellent plugin. While that's not happening, I could use some help.

I'm trying to just change the Preview Container to a separate div, I've tried "elPreviewContainer" in a hundred different ways and so far no success, oddly enough the "elPreviewImage" does change the structure, but that's not what I'm looking for, I only want to display the preview container on a sub-level div that's all and per the docs, I think the "elPreviewContainer" is the way to go but I simply can't make it happen.

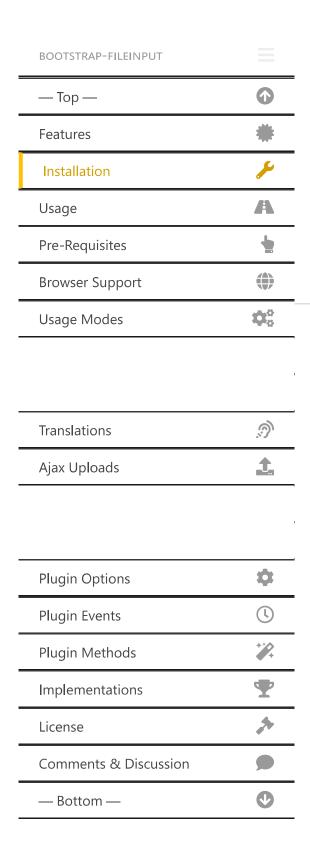
This is my code

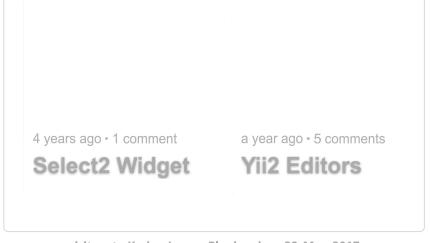
Reply • Share >



Amar • a year ago • edited

I need help with resumable uploads, i am able to upload my video in chunks but is there any plugin that can play my chunked videos, i don't want to





visitors to Krajee Jquery Plugins since 22-May-2017



© Kartik Visweswaran 2022

Powered by Yii Framework | Privacy Policy