

# Bootstrap File Input

[Source](#)

[Docs](#)

[Demo](#)

[Home](#) / Bootstrap File Input

☆ Star 5,205

🍴 Fork 2,414

👁 Watch 384

👤 Follow 2,160

Thankful to Krajee!

☕ BUY A COFFEE

or

💰 DONATE

to get more out of us.

[Learn](#)

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

BOOTSTRAP-FILEINPUT



— Top —



Features



Installation



Usage



Pre-Requisites



Browser Support



Usage Modes



Translations



Ajax Uploads



Plugin Options



Plugin Events



Plugin Methods



Implementations



License



















Comments & Discussion



— Bottom —



BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

Bootstrap 5 history reader. This can also be 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 [refer this webtip](#) 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)

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

# 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).

















BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

- 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



















BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

- 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 templated 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 example To 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`.

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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.

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

















## 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 [uploadUrl](#) property.
- Enhance plugin to now allow files to be added, appended, removed (based on FEEDBACK from many). Thus one can append files to preview.

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

- 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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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.



— Top —



# Usage

Features



Installation



Usage



Pre-Requisites



Browser Support



Usage Modes



Translations



Ajax Uploads



Plugin Options



Plugin Events



Plugin Methods



Implementations



License



Comments &amp; 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

Load the following assets in your header.

```

1. <!-- bootstrap 5.x or 4.x is supported. You can also use
2. <link rel="stylesheet" href="https://cdn.jsdelivr.net/n
3.
4. <!-- default icons used in the plugin are from Bootstrap
5. <link rel="stylesheet" href="https://cdn.jsdelivr.net/n
6.
7. <!-- alternatively you can use the font awesome icon li
8. <!-- Link rel="stylesheet" href="https://use.fontawesom
9.
10. <!-- the fileinput plugin styling CSS file -->
11. <link href="https://cdn.jsdelivr.net/gh/kartik-v/bootst
12.
13. <!-- if using RTL (Right-To-Left) orientation, Load the
14. <!-- Link href="https://cdn.jsdelivr.net/gh/kartik-v/bo
15.
16. <!-- the jQuery Library -->
17. <script src="https://code.jquery.com/jquery-3.6.0.min.j
18.
19. <!-- buffer.min.js and filetype.min.js are necessary in
20.

```

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

```

















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

```

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 [Piexifjs plugin by hMatoba](#). It is required to be

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— 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 **bootstrap-fileinput** 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.

```

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

## [Usage: Step 2b](#)



BOOTSTRAP-FILEINPUT	
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📤
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

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 [jQuery](#).
- Most modern browsers supporting HTML5 file inputs and FileReader API including CSS3 & JQuery. For details, refer the [browser support section](#).
- For file previews to work, the browser must support the HTML5 FileReaderAPI - else the plugin will auto-degrade 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.



— Top —



Features



Installation



Usage



Pre-Requisites



Browser Support



Usage Modes



Translations



Ajax Uploads



Plugin Options



Plugin Events



Plugin Methods



Implementations



License



Comments &amp; Discussion



















— Bottom —



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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

<b>HTML 5 XHR2 &amp; FormData</b>	Allow using ajax uploads with ability to append / delete files and track using a progress bar.	Browsers
<b>HTML5 Drag Drop</b>	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
<b>HTML5 Canvas</b>	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
<b>HTML5 Download Attribute</b>	The <code>download</code> 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

BOOTSTRAP-FILEINPUT	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	📖
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📁
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

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

In this mode, you do not set the [uploadUrl](#) 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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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 [uploadUrl](#) property to a VALID ajax processing server action/URL. If the [uploadUrl](#) 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	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— 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	 <a href="#">1</a>	 <a href="#">2</a>
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	 <a href="#">3</a>	
Ability to append files to already selected list		
Ability to delete files to already selected list	 <a href="#">1</a>	
Progress bar for uploads		

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

















Upload stats for uploads	✗	✓
Resumable / Chunk uploads	✗	✓
Read additional form data	Directly via form submit	Via <code>uploadExtraData</code>
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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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 [language](#) 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 [uploadAsync](#) 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 [uploadUrl](#) 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



BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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 [uploadUrl](#) must send data back as a json encoded object. For example, the server could return a JSON object like below:

```

1. // example JSON response from server
2. {
3.     error: 'An error exception message if applicable',
4.     initialPreview: [
5.         // initial preview thumbnails for server upload
6.     ],
7.     initialPreviewConfig: [

```

BOOTSTRAP-FILEINPUT	
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📶
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

```

10.     initialPreviewThumbTags: [
11.         // initial preview thumbnail tags configuration
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 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:

```

1.     initialPreview: [
2.         '<img src= '/images/desert.jpg' class='file-pre
3.     ]

```

- **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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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', //
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}': ' ',
6.          '{CUSTOM_TAG_INIT}': '<span class=\'custom
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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

- 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 `uploadAsync` 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 `uploadUrl` 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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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)

In synchronous mode as well, the [uploadUrl](#) must send data back as a json encoded object. For example the server could return a JSON object like below:

```

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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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 [initialPreview](#) option setting. For example:

```

1.  initialPreview: [
2.      '<img src='/images/desert.jpg' class='file-pre
3.      '<img src='/images/jellyfish.jpg' class='file-
4.  ]

```

- **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:

```

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

```

BOOTSTRAP-FILEINPUT	
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	📖
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📤
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	🔧
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

```

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.

```

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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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 capabilities 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 [resumable uploads demos](#) 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



















BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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 [uploadUrl](#) receives the following parameters as data from the plugin for resumable uploads. The plugin by default sends this information as a JSON object via `POST`. You can configure these parameter names via [uploadParamNames](#).

- **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

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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

In addition all the data which you set via [uploadExtraData](#) 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 [uploadUrl](#) 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](#)

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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', //
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}': ' ',
6.          '{CUSTOM_TAG_INIT}': '<span class=\'custom
7.      }
8.  ]
```

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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.


















## [Client Code Example](#) (HTML / JS)

```

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

```

## [Server Code Example](#) (PHP)


















BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

```

3. header('Content-Type: application/json'); // set json r
4. $outData = upload(); // a function to upload the bootst
5. echo json_encode($outData); // return json data
6. exit(); // terminate

7.
8. // main upload function used above
9. // upload the bootstrap-fileinput files
10. // returns associative array
11. function upload() {
12.     $preview = $config = $errors = [];
13.     $targetDir = '/webroot/uploads';
14.     if (!file_exists($targetDir)) {
15.         @mkdir($targetDir);
16.     }
17.     $fileBlob = 'fileBlob'; // the
18.     if (isset($_FILES[$fileBlob]) && isset($_POST['uplo
19.         $token = $_POST['uploadToken']; // get
20.         if (!validateToken($token)) { // you
21.             return [
22.                 'error' => 'Access not allowed' // ret
23.             ];
24.         }
25.         $file = $_FILES[$fileBlob]['tmp_name']; // the
26.         $fileName = $_POST['fileName']; // you
27.         $fileSize = $_POST['fileSize']; // you
28.         $fileId = $_POST['fileId']; // you
29.         $index = $_POST['chunkIndex']; // the
30.         $totalChunks = $_POST['chunkCount']; // the
31.         $targetFile = $targetDir.'/'.$fileName; // you
32.         if ($totalChunks > 1) { // cre
33.             $targetFile .= '_' . str_pad($index, 4, '0'
34.         }
35.         $thumbnail = 'unknown.jpg';
36.         if(move_uploaded_file($file, $targetFile)) {
37.             // get list of all chunks uploaded so far t
38.             $chunks = glob("{$targetDir}/{\$fileName}_*"
39.             // check uploaded chunks so far (do not com
40.             $allChunksUploaded = $totalChunks > 1 && co
41.             if ($allChunksUploaded) { // all
42.                 $outFile = $targetDir.'/'.$fileName;
43.                 // combines all file chunks to one file
44.                 combineChunks($chunks, $outFile);
45.             }
46.             // if you wish to generate a thumbnail imag
47.             $targetUrl = getThumbnailUrl($path, $fileNa
48.             // separate link for the full blown image f

```

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

```

51.         chunkIndex => $index, // the
52.         'initialPreview' => $targetUrl, // the
53.         'initialPreviewConfig' => [
54.             [
55.                 'type' => 'image', // check
56.                 'caption' => $fileName, // caption
57.                 'key' => $fileId, // keys
58.                 'fileId' => $fileId, // file
59.                 'size' => $fileSize, // file
60.                 'zoomData' => $zoomUrl, // separate
61.             ]
62.         ],
63.         'append' => true
64.     ];
65.     } else {
66.         return [
67.             'error' => 'Error uploading chunk ' . $
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 to
78. function combineChunks($chunks, $targetFile) {
79.     // open target file handle
80.     $handle = fopen($targetFile, 'a+');
81.
82.     foreach ($chunks as $file) {
83.         fwrite($handle, file_get_contents($file));
84.     }
85.
86.     // you may need to do some checks to see if file
87.     // is matching the original (e.g. by comparing file
88.
89.     // after all are done delete the chunks
90.     foreach ($chunks as $file) {
91.         @unlink($file);
92.     }
93.
94.     // close the file handle
95.     fclose($handle);
96. }
97.

```

BOOTSTRAP-FILEINPUT	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📁
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	🔧
Implementations	🏆
License	🔑
Comments & Discussion	💬
— Bottom —	↓

```

100. // assuming this is an image file or video file
101. // generate a compressed smaller version of the file
102. // here and return the status
103. $sourceFile = $path . '/' . $fileName;
104. $targetFile = $path . '/thumbs/' . $fileName;
105. //
106. // generateThumbnail: method to generate thumbnail
107. // using $sourceFile and $targetFile
108. //
109. if (generateThumbnail($sourceFile, $targetFile) ==
110.     return 'http://localhost/uploads/thumbs/' . $fileName;
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.

```


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

```


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

[View Plugin Options](#)


## [Events](#)

BOOTSTRAP-FILEINPUT 


---

— Top — 


---

Features 


---

**Installation** 


---

Usage 


---

Pre-Requisites 


---

Browser Support 


---

Usage Modes 


---

Translations 


---

Ajax Uploads 


---

Plugin Options 


---

Plugin Events 


---

Plugin Methods 


---

Implementations 


---

License 

---

Comments & Discussion 

---

— Bottom — 

## Methods

[View Plugin Methods](#)

## Implementations

The plugin has been implemented as extensions in the following frameworks


Sl.	Framework	Extension Source	Developed By	Extension Demo
1.	<a href="#">Yii Framework 2.0</a>	<a href="#">Yii2 Widgets</a>	<a href="#">Krajee</a>	<a href="#">Yii2 File Input</a>

Do you know any other framework extension using this plugin which is not listed here? [Tell us](#) so that we can consider its inclusion in this list.


## License

**bootstrap-fileinput** is released under the BSD 3-Clause License. See the bundled [LICENSE.md](#) for details.




BOOTSTRAP-FILEINPUT 


---

— Top — 


---

Features 


---

**Installation** 


---

Usage 


---

Pre-Requisites 


---

Browser Support 


---

Usage Modes 


---

Translations 


---

Ajax Uploads 


---

Plugin Options 


---

Plugin Events 


---

Plugin Methods 


---

Implementations 


---

License 

---

Comments & Discussion 

---


— Bottom — 




# Comments & Discussion

## Note

You can now visit the [Krajee Webtips Q & A forum](#) for searching OR asking questions OR helping programmers with answers on these extensions and plugins. For asking a question [click here](#). 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.

[Comments](#)   [Community](#)      1 [Login](#)


 Favorite 74    Tweet    Share   Sort by Newest

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

 **Stephen Persson** • 3 months ago • edited

BOOTSTRAP-FILEINPUT	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📤
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

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](#)

^ | v • Reply • Share ›



**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

^ | v • Reply • Share ›



**Witter** • 8 months ago

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

^ | v • Reply • Share ›



**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	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📤
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

[see more](#)

^ | v • 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](#)

^ | v • 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.....

BOOTSTRAP-FILEINPUT	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	📖
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📤
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

[see more](#)

^ | v • 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](#)

^ | v • Reply • Share ›

**Ballu Ikka** ↗ daniel 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.....

BOOTSTRAP-FILEINPUT	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📁
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

[see more](#)

^ | v • 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](#)

^ | v • 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.....

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
<hr/>	
Translations	
Ajax Uploads	
<hr/>	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

^ | v • Reply • Share ›



**Mark Bagnall** • a year ago

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

^ | v • Reply • Share ›



**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[]="";
```

[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	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

[see more](#)[^](#) | [v](#) • [Reply](#) • [Share](#) >**Ballu Ikka** [↗](#) 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](#)[^](#) | [v](#) • [Reply](#) • [Share](#) >**Sooraj EK** • a year ago • edited

Hello,

















```
<form action="process.php" method="post"
  enctype="multipart/form-data">
  <input id="krajee" type="file" name="file[]"
    multiple="true">
  <input type="submit" name="submit" id="submit">
</form>
```

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

```
  theme: 'fa',
```

```
  uploadUrl: "#",
```

```
  allowedFileExtensions: ['.jpg', '.jpeg', '.png', '.gif'],
```

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

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

[see more](#)

^ | v • 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](#)

^ | v • 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** to the plugin. You need to handle the case



BOOTSTRAP-FILEINPUT	☰
— Top —	↑
Features	⚙️
<b>Installation</b>	🔧
Usage	🏠
Pre-Requisites	👉
Browser Support	🌐
Usage Modes	⚙️
Translations	🗣️
Ajax Uploads	📶
Plugin Options	⚙️
Plugin Events	🕒
Plugin Methods	✏️
Implementations	🏆
License	🔨
Comments & Discussion	💬
— Bottom —	↓

based on your app specific confirmation inputs OR deleting it.

[see more](#)

^ | v • 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](#)

^ | v • Reply • Share ›



















**Stig Kølbaek** • a year ago • edited

I need to use Bootstrap Input File with FreeASPUUpload (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 FreeASPUUpload page trigger a callback to Bootstrap Input File on error/success?

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

## SyntaxError: Unexpected end of JSON innit

[see more](#)

^ | v • 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

^ | v • 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.

BOOTSTRAP-FILEINPUT	
— Top —	
Features	
<b>Installation</b>	
Usage	
Pre-Requisites	
Browser Support	
Usage Modes	
Translations	
Ajax Uploads	
Plugin Options	
Plugin Events	
Plugin Methods	
Implementations	
License	
Comments & Discussion	
— Bottom —	

[see more](#)[^](#) | [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


```
$("#input-b6").fileinput({
```

[see more](#)[^](#) | [v](#) • [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 use the combine\_chunks method, please help me

BOOTSTRAP-FILEINPUT 


---

— Top — 


---

Features 


---

**Installation** 


---

Usage 


---

Pre-Requisites 


---

Browser Support 


---

Usage Modes 


---

Translations 


---

Ajax Uploads 


---

Plugin Options 


---

Plugin Events 


---

Plugin Methods 


---

Implementations 


---

License 

---

Comments & Discussion 

---

— Bottom — 

4 years ago • 1 comment

**Select2 Widget**

a year ago • 5 comments

**Yii2 Editors**

visitors to Krajee JQuery Plugins since 22-May-2017



© Kartik Visweswaran 2022

Powered by [Yii Framework](#) | [Privacy Policy](#)