Uppy Cheat Sheet

Uppy Core

@uppy/core	required for all plugins

UI

@uppy/dashboard	complete UI
@uppy/drag-drop	plain drag and drop area

Uploaders

@uppy/aws-s3	upload direclty to AWS S3 or compatible services
@uppy/aws-s3-multipart	S3 Multipart upload directly to AWS or compatible services
@uppy/transloadit	upload files directly to Transloadit for all kinds of processing
@uppy/tus	resumable uploads via the open tus standard
@uppy/xhr-upload	regular uploads to a HTTP server

UI Elements

@uppy/drop-target	drag-and-drop files on any element on the page DB DD UU
@uppy/progress-bar	minimal upload progress indicator DD UU
@uppy/status-bar	advanced upload progress status bar ID DD UU
@uppy/informer	send notifications like “smile” before taking a selfie ID DD UU
@uppy/thumbnail-generator	generate preview thumbnails for images to be uploaded ID DD UU

Plugins for Dashboard DB

Sources

@uppy/audio	record audio
@uppy/box	import from Box
@uppy/dropbox	import from Dropbox
@uppy/facebook	import from Facebook
@uppy/google-drive	import from Google Drive
@uppy/instagram	import from Instagram
@uppy/onedrive	import from OneDrive
@uppy/screen-capture	record your screen, and optionally your microphone
@uppy/unsplash	import from Unsplash
@uppy/url	import from any URL
@uppy/webcam	record or make a picture with your webcam
@uppy/zoom	import from Zoom

@uppy/remote-sources

add all available remote sources (Box, Dropbox, Facebook, GoogleDrive, Instagram, OneDrive, Unsplash, Url, and Zoom)

UI

@uppy/image-editor	allow to crop, rotate, zoom, and flip images that are added to Uppy

Misc DB DD UU

@uppy/golden-retriever	restore files after a browser crash or if accidentally the tab is closed
@uppy/compressor	optimize images before upload
@uppy/form	collect metadata from form right before an Uppy upload
@uppy/locales	translate Uppy into your language of choice

Legend

ID	included by default in Dashboard UI
DB	can be used in Dashboard UI
DD	can be used in Drag & Drop UI
UU	can be used in Custom User Interface

Uppy Core Cheat Sheet

Core https://uppy.io/docs/uppy

```
npm install @uppy/core
```

```
import Uppy from '@uppy/core';
import '@uppy/core/dist/style.min.css';

const uppyInstance = new Uppy();
```

Core exports:
- Uppy (default class)
- UIPlugin
- BasePlugin
- debugLogger

Uppy file properties

Uppy keeps files in state with the File browser API, wrapped in an object (Uppy file)
These properties should be mutated through methods

file.source	name of the plugin that was responsible for adding the file
file.id	unique ID for the file
file.name	name of the file
file.meta	object containing file metadata
file.type	MIME type of the file
file.data	• *for local files:* is the actual File or Blob object representing the its contents • *for files imported from remote providers:* file data is not available in the browser
file.progress	upload progress data (bytesUploaded, bytesTotal, uploadStarted, uploadComplete, percentage)
file.size	size in bytes of the file
file.isRemote	true if the file is imported from a remote provider
file.remote	bag of data for remote providers
file.preview	optional URL to a visual thumbnail of the file
file.uploadURL	URL to the uploaded file, when an upload is completed

Options

option	**default**	**description**
id	uppy	site-wide unique ID for Uppy instance
autoProceed	false	upload as soon as files are added
allowMultipleUploadBatches	true	allow several upload batches
debug	false	send debugging and warning logs
logger	justErrorsLogger	logger used for uppy.log
meta	{}	key/value pairs to add to each file’s metadata
onBeforeFileAdded(file, files)	(file) => file	function called before a file is added to Uppy
onBeforeUpload(files)	(files) => files	function called before upload is initiated
locale		override locale strings
store	DefaultStore	store that is used to keep track of internal state
infoTimeout	5000	how long an Informer notification will be visible
restrictions	{}	conditions for restricting an upload

property	**value**	**description**
maxFileSize	number	maximum file size in bytes for each individual file
minFileSize	number	minimum file size in bytes for each individual file
maxTotalFileSize	number	maximum file size in bytes for all selected files
maxNumberOfFiles	number	number of files that can be selected
minNumberOfFiles	number	minimum number of files that must be selected before the upload
allowedFileTypesArray	array	wildcards (image/*), exact mime types (image/jpeg), or extensions (.jpg)
requiredMetaFieldsArray	array<string>	make keys from the meta object in every file required before uploading

Core https://uppy.io/docs/uppy

`npm install @uppy/core`

Methods

use(plugin, opts)	add a plugin to Uppy
removePlugin(instance)	uninstall and remove a plugin
getPlugin(id)	get a plugin by its id
getID()	get the Uppy instance ID
addFile(file)	add a new file to Uppy's internal state. return the file's generated id
removeFile(fileID)	remove a file from Uppy
getFile(fileID)	get a specific Uppy file by its ID
getFiles()	get an array of all added Uppy files
upload()	start uploading the files added
pauseResume(fileID)	toggle pause/resume on an upload
pauseAll()	pause all uploads
resumeAll()	resume all uploads
retryUpload(fileID)	retry an upload
retryAll()	retry all uploads
cancelAll()	cancel all uploads
setState(patch)	update Uppy's internal state
getState()	returns the current state from the Store
setFileState(fileID, state)	update the state for a single file
setMeta(data)	alters global meta object
setFileMeta(fileID, data)	update metadata for a specific file
setOptions(opts)	change the options Uppy initialized with
reset()	stop all uploads in progress, clear file selection, set progress to 0
close()	uninstall all plugins and close the Uppy instance
logout()	calls provider.logout() on each remote provider plugin (Dropbox, ...)
log(message, type)	log a message
info(message, type, duration)	sets a message in state, that can be shown by notification UI plugins
addPreProcessor(fn)	add a preprocessing function
addUploader(fn)	add an uploader function
addPostProcessor(fn)	add a postprocessing function
removePreProcessor(fn)	remove a processor function that was added before
removeUploader(fn)	remove a uploader function that was added before
removePostProcessor(fn)	remove a processor function that was added before
on('event', action)	subscribe to an uppy-event
once('event', action)	create an event listener that fires once
off('event', action)	unsubscribe to an uppy-event

Core https://uppy.io/docs/uppy

```
npm install @uppy/core
```

Events

You can use on *and* once *to listen to these events*

event	emitted
uppy.on('file-added', (file) => {});	each time a file is added
uppy.on('files-added', (files) => {});	each time when one or more files are added
uppy.on('file-removed', (file, reason) => {});	each time a file is removed
uppy.on('upload', (data) => {});	when the upload starts
uppy.on('preprocess-progress', (progress) => {});	progress of the pre-processors
uppy.on('progress', (progress) => {});	each time the total upload progress is updated
uppy.on('upload-progress', (file, progress) => {});	each time an individual file upload progress is available
uppy.on('postprocess-progress', (progress) => {});	progress of the post-processors
uppy.on('upload-success', (file, response) => {});	each time a single upload is completed
uppy.on('complete', (result) => {});	when all uploads are complete
uppy.on('error', (error) => {});	when Uppy fails to upload/encode the entire upload
uppy.on('upload-error', (file, error, response) => {});	each time a single upload failed
uppy.on('upload-retry', (fileID) => {});	when an upload has been retried
uppy.on('upload-stalled', (error, files) => {});	when an upload has not received any progress in some time
uppy.on('retry-all', (fileIDs) => {});	when all failed uploads are retried
uppy.on('info-visible', () => {});	when "info" message should be visible in UI
uppy.on('info-hidden', () => {});	when "info" message should be hidden in UI
uppy.on('cancel-all', () => {});	when cancelAll() is called, all uploads are canceled, files removed, and progress is reset
uppy.on('restriction-failed', (file, error) => {});	when a file violates certain restrictions when added
uppy.on('reset-progress', () => {});	when resetProgress() is called, each file has its upload progress reset to zero

Dashboard https://uppy.io/docs/dashboard

```
npm install @uppy/dashboard
```

```
import Dashboard from '@uppy/dashboard';
import '@uppy/dashboard/dist/style.min.css';

new Uppy().use(Dashboard, { inline: true, target: '#dashboard' });
```

Methods

openModal()	show Dashboard modal
closeModal()	hide Dashboard modal
isModalOpen()	return true if Dashboard modal is open, false otherwise

Events

You can use on *and* once *to listen to these events*

event	fired
uppy.on('dashboard:modal-open', () => {});	when Dashboard modal is open
uppy.on('dashboard:modal-closed', () => {});	when Dashboard modal is closed
uppy.on('dashboard:file-edit-start', (file) => {});	when user clicks “edit” icon next to a file
uppy.on('dashboard:file-edit-complete', (file) => {});	when user finished editing the file metadata

Options

option	default	description
id	Dashboard	unique identifier. Plugins added by Dashboard get unique IDs like so: 'Dashboard:StatusBar', 'Dashboard:Informer', ...
target	body	where to render Dashboard (element, class, or id)
inline	false	render Dashboard as a modal or inline
trigger	null	CSS selector that will trigger opening the modal
width	750	width in pixels (used when inline: true)
height	550	height in pixels (used when inline: true)
waitForThumbnailsBeforeUpload	false	wait for all thumbnails to be ready before starting upload
showLinkToFileUploadResult	false	turn file icon and thumbnail into a link to uploaded file
showProgressDetails	false	show/hide progress details in status bar
hideUploadButton	false	show/hide upload button
hideRetryButton	false	hide retry button in status bar and each individual file
hidePauseResumeButton	false	hide pause/resume button
hideCancelButton	false	hide cancel button in status bar and each individual file
hideProgressAfterFinish	false	hide status bar after the upload has finished
doneButtonHandler()		passed to status bar and will render a “Done” button in place of pause/resume/cancel buttons, once upload is done
showSelectedFiles	true	show list of added files with a preview and file info
showRemoveButtonAfterComplete	false	show remove button on every file after a successful upload
singleFileFullScreen	true	when only one file is selected, its preview and meta info will be centered and enlarged
metaFields	null	create text or custom input fields for the user to fill in

Dashboard https://uppy.io/docs/dashboard

```
npm install @uppy/dashboard
```

Options

option	default	description
closeModalOnClickOutside	false	if true, automatically close modal when the user clicks outside of it
closeAfterFinish	false	if true, automatically close modal when all current uploads are complete
disablePageScrollWhenModalOpen	true	disable page scroll when the modal is open
animateOpenClose	true	add animations when modal dialog is opened or closed
fileManagerSelectionType	files	type of selections allowed when browsing your file system via the file manager selection window
proudlyDisplayPoweredByUppy	true	show the Uppy logo with a link
disableStatusBar	false	disable status bar
disableInformer	false	disable informer
disableThumbnailGenerator	false	disable thumbnail generator
theme	light	light or dark theme
autoOpenFileEditor	false	automatically open file editor for the first file
disabled	false	if true, makes Dashboard grayed-out and non-interactive
disableLocalFiles	false	disable local files
onDragOver(event)		callback for ondragover event handler
onDrop(event)		callback for ondrop event handler
onDragLeave(event)		callback for ondragleave event handler

Drag and Drop https://uppy.io/docs/drag-drop/

```
npm install @uppy/drag-drop
```

```
import DragDrop from '@uppy/drag-drop’;
import '@uppy/drag-drop/dist/style.min.css’;

new Uppy().use(DragDrop, { target: '#drag-drop' });
```

Options

option	default	description
id	DragDrop	unique identifier for the plugin
target	null	DOM element, CSS selector, or plugin to place drag and drop
width	100%	drag and drop area width
height	100%	drag and drop area height
note	null	text that explains something about the upload for the user
onDragOver(event)		callback for ondragover event handler
onDragLeave(event)		callback for ondragleave event handler
onDrop(event)		callback for ondrop event handler

Progress Bar

https://uppy.io/docs/progress-bar/

```
npm install @uppy/progress-bar
```

```
import ProgressBar from '@uppy/progress-bar';
import '@uppy/progress-bar/dist/style.min.css';

new Uppy().use(ProgressBar, { target: '#progress-bar' });
```

Options

option	default	description
id	ProgressBar	unique identifier for the plugin
target	null	DOM element, CSS selector, or plugin to mount progress bar
fixed	false	show progress bar at the top of the page with position: fixed
hideAfterFinish	true	hide progress bar after the upload has finished

Status Bar

https://uppy.io/docs/status-bar/

```
npm install @uppy/status-bar
```

```
import StatusBar from '@uppy/status-bar';
import '@uppy/status-bar/dist/style.min.css';

new Uppy().use(StatusBar, { target: '#status-bar' });
```

Options

option	default	description
id	StatusBar	unique identifier for the plugin
target	null	DOM element, CSS selector, or plugin to mount status bar
hideAfterFinish	true	hide Status Bar after the upload is complete
showProgressDetails	false	display remaining upload size and estimated time
hideUploadButton	false	hide upload button
hideRetryButton	false	hide retry button
hidePauseResumeButton	false	hide pause/resume buttons
hideCancelButton	false	hide cancel button
doneButtonHandler	null	will render a “Done” button in place of pause/resume/cancel buttons, once the upload/encoding is done

Drop Target https://uppy.io/docs/drop-target/ `npm install @uppy/drop-target`

```
import DropTarget from '@uppy/drop-target';
import '@uppy/drop-target/dist/style.min.css';

new Uppy().use(DropTarget, { target: document.body });
```

Options

option	default	description
target	null	DOM element, CSS selector, or plugin to place drop target area into
onDragLeave		event listener for dragleave event
onDragOver		event listener for dragover event
onDrop		event listener for drop event

Image Editor https://uppy.io/docs/image-editor/ `npm install @uppy/image-editor`

```
import ImageEditor from '@uppy/image-editor';
import '@uppy/image-editor/dist/style.min.css';

new Uppy().use(ImageEditor, { target: Dashboard });
```

Options

option	default	description
id	ImageEditor	unique identifier for this plugin
quality	0.8	quality of resulting blob that will be saved after editing/cropping
cropperOptions		options that will be directly passed to Cropper.js
actions		show action buttons. To hide all actions pass false

Events

You can use on *and* once *to listen to these events*

event	emitted
uppy.on('file-editor:start', (file) => {});	when selectFile(file) is called
uppy.on('file-editor:complete', (updatedFile) => {});	after save(blob) is called
uppy.on('file-editor:cancel', (file) => {});	when uninstall is called or when the current image editing changes are discarded

Informer https://uppy.io/docs/informer/

```
npm install @uppy/informer
```

```
import Informer from '@uppy/informer';
import '@uppy/informer/dist/style.min.css';

new Uppy().use(Informer, { target: '#informer' });
```

Informer gets its data from `uppy.state.info`, which is updated by various plugins via `uppy.info` method

E.g. in Compressor plugin:

```
const size = prettierBytes(totalCompressedSize);
this.uppy.info(this.i18n('compressedX', { size }), 'info');
```

Options

option	default	description
id	Informer	unique identifier for this plugin
target	null	DOM element, CSS selector, or plugin to mount the Informer into

Thumbnail generator

```
npm install @uppy/thumbnail-generator
```

```
import ThumbnailGenerator from '@uppy/thumbnail-generator';
import '@uppy/thumbnail-generator/dist/style.min.css';

new Uppy().use(ThumbnailGenerator);
```

https://uppy.io/docs/thumbnail-generator

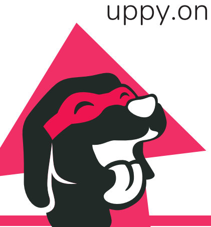
Options

option	default	description
id	ThumbnailGenerator	unique identifier for this plugin
thumbnailWidth	200	width of resulting thumbnail
thumbnailHeight	200	height of resulting thumbnail
thumbnailType	image/jpeg	MIME type of resulting thumbnail
waitForThumbnailsBeforeUpload	false	wait for all thumbnails to be ready before starting the upload

Events

You can use on *and* once *to listen to these events*

event	emitted
uppy.on('thumbnail:generated', (file, preview) => {});	when a thumbnail is generated



Webcam https://uppy.io/docs/webcam/

```
npm install @uppy/webcam
```

```
import Webcam from '@uppy/webcam';
import '@uppy/webcam/dist/style.min.css';

new Uppy().use(Webcam, { target: Dashboard });
```

Options

option	default	description
id	Webcam	unique identifier for this plugin
target	null	DOM element, CSS selector, or plugin to place Webcam into
countdown	false	amount of seconds to wait before taking a snapshot
onBeforeSnapshot	Promise.resolve	hook function to call before a snapshot is take
modes	[] (all modes)	types of recording modes to allow (possible values: video-audio, video-only, audio-only, picture)
mirror	true	mirror preview image from camera
videoConstraints	{}	configure MediaTrackConstraints
showVideoSourceDropdown	false	show a dropdown to choose the video device to use
showRecordingLength	false	show the length of recording while recording is in progress
preferredVideoMimeType	null	preferred mime type for video recordings
preferredImageMimeType	image/jpeg	preferred mime type for images
mobileNativeCamera	isMobile()	replace Uppy's custom camera UI on mobile and tablet with native device camera

Screen capture https://uppy.io/docs/screen-capture/

```
npm install @uppy/screen-capture
```

```
import ScreenCapture from '@uppy/screen-capture';
import '@uppy/screen-capture/dist/style.min.css';

new Uppy().use(ScreenCapture, { target: Dashboard });
```

Options

option	default	description
id	ScreenCapture	unique identifier for this plugin
target	null	DOM element, CSS selector, or plugin to place ScreenCapture into
title	Screen Capture	title/name shown in UI
displayMediaConstraints	null	options passed to MediaDevices.getDisplayMedia()
userMediaConstraints	null	options passed to MediaDevices.getUserMedia()
preferredVideoMimeType	null	preferred mime type for video recordings

Companion Plugins - Remote sources

Box

```
npm install @uppy/box
```

```
import Box from '@uppy/box’;
new Uppy().use(Dashboard, {...}).use(Box, {
    target: Dashboard,
    companionUrl: 'https://your-companion.com',
});
```

Dropbox

```
npm install @uppy/dropbox
```

```
import Dropbox from '@uppy/dropbox’;
new Uppy().use(Dashboard, {...}).use(Dropbox, {
    ...
});
```

Facebook

```
npm install @uppy/facebook
```

```
import Facebook from '@uppy/facebook’;
new Uppy().use(Dashboard, {...}).use(Facebook, {
    ...
});
```

Google Drive

```
npm install @uppy/google-drive
```

```
import GoogleDrive from '@uppy/google-drive’;
new Uppy().use(Dashboard, {...}).use(GoogleDrive, {
    ...
});
```

Instagram

```
npm install @uppy/instagram
```

```
import Instagram from '@uppy/instagram’;
new Uppy().use(Dashboard, {...}).use(Instagram, {
    ...
});
```

OneDrive

```
npm install @uppy/onedrive
```

```
import OneDrive from '@uppy/onedrive’;
new Uppy().use(Dashboard, {...}).use(OneDrive, {
    ...
});
```

Unsplash

```
npm install @uppy/unsplash
```

```
import Unsplash from '@uppy/unsplash’;
new Uppy().use(Dashboard, {...}).use(Unsplash, {
    ...
});
```

Import from URL

```
npm install @uppy/url
```

Companion has SSRF protections build-in so you don’t have to worry about security implications of arbitrary URLs

```
import Url from '@uppy/url’;
new Uppy().use(Dashboard, {...}).use(Url, {
    ...
});
```

Zoom

```
npm install @uppy/zoom
```

```
import Zoom from '@uppy/zoom’;
new Uppy().use(Dashboard, {...}).use(Zoom, {
    ...
});
```

Uppy **Source** **Cheat Sheet**

Options

option	default	description
id	plugin name	unique identifier for the plugin
target	null	DOM element, CSS selector, or plugin to place the plugin into
title	plugin name	title/name shown in UI
companionUrl	null	URL to a Companion instance
companionHeaders	{}	custom headers that should be sent on every request
companionAllowedHosts	companionUrl	valid and authorised URL(s) from which OAuth responses should be accepted
companionCookiesRule	same-origin	correlates to RequestCredentials value

Companion Plugins - All Remote sources

npm install @uppy/remote-sources

```
import RemoteSources from '@uppy/remote-sources';

new Uppy();
  .use(Dashboard).use(RemoteSources, { companionUrl: 'https://your-companion-url' });
```

requires Dashboard

add all available remote sources, such Instagram, Google Drive, Dropbox,... to Uppy Dashboard

Options

option	default	description
id	RemoteSources	unique identifier for the plugin
sources	['Box','Dropbox','Facebook', 'GoogleDrive','Instagram', 'OneDrive','Unsplash','Url','Zoom']	list of remote sources that will be enabled
companionUrl	null	URL to a Companion instance
companionHeaders	{}	custom headers to be sent on every request
companionAllowedHosts	companionUrl	valid and authorised URL(s) from which OAuth responses should be accepted
companionCookiesRule	same-origin	correlates to the RequestCredentials value
target	Dashboard	DOM element, CSS selector, or plugin

Audio

https://uppy.io/docs/audio/

`npm install @uppy/audio`

```
import Audio from '@uppy/audio';
import '@uppy/audio/dist/style.min.css';

new Uppy().use(Audio, { target: Dashboard });
```

Options

option	default	description
id	audio	unique identifier for this plugin
target	null	DOM element, CSS selector, or plugin to place Audio into
title	Audio	title/name shown in the UI
showAudioSourceDropdown	false	show a dropdown to select the audio device

Tus https://uppy.io/docs/tus/

```
npm install @uppy/tus
```

```
import Tus from '@uppy/tus';

new Uppy().use(Tus, { endpoint: 'https://tusd.tusdemo.net/files/' });
```

Options

option	default	description
id	Tus	unique identifier for this plugin
endpoint	null	URL of the tus server
headers	null	object or function returning an object with HTTP headers to send along requests
chunkSize	Infinity	max size of a PATCH request body in bytes
withCredentials	false	configure the requests to send Cookies using xhr.withCredentials property
retryDelays	[0, 1000, 3000, 5000]	when uploading a chunk fails, automatically try again after the millisecond intervals
onBeforeRequest(req, file)		behaves like the onBeforeRequest function from tus-js-client but with the added file argument
allowedMetaFields	null	array of field names to limit the metadata fields that will be added to uploads as Tus Metadata
limit	20	limit the amount of uploads at the same time
onShouldRetry: (err, retryAttempt, options, next)		when an upload fails onShouldRetry is called with the error and the default retry logic as the last argument

AWS S3 https://uppy.io/docs/aws-s3/

```
npm install @uppy/aws-s3
```

```
import AwsS3 from '@uppy/aws-s3’;

new Uppy().use(AwsS3, { companionUrl: 'http://companion.uppy.io' });
```

Options

option	default	description
id	AwsS3	unique identifier for this plugin
companionUrl	null	companion instance to use for signing S3 uploads
companionHeaders	{}	custom headers that should be senton every request
allowedMetaFields	null	array of field names to limit the metadata fields that will be added to upload as query parameters
getUploadParameters(file)	null	function that returns upload parameters for a file
timeout	30_000	when no upload progress events have been received for this amount of milliseconds, assume connection has an issue and abort upload
limit	5	limit the amount of uploads at the same time (0 means no limit on concurrent uploads) recommended: between 5 and 20
getResponseData(responseText, response)		customize response handling once an upload is completed

AWS S3 Multipart

https://uppy.io/docs/aws-s3/

```
npm install @uppy/aws-s3-multipart
```

for large files (100 MB+).

```
import AwsS3Multipart from '@uppy/aws-s3-multipart';

new Uppy().use(AwsS3Multipart, { companionUrl: 'http://companion.uppy.io' });
```

Options

option	default	description
limit	6	maximum amount of files to upload in parallel
companionUrl	null	URL to a Companion instance
companionHeaders	{}	custom headers that should be sent on every request
companionCookiesRule	same-origin	correlates to the RequestCredentials value
retryDelays	[0, 1000, 3000, 5000]	intervals in milliseconds used to retry a failed chunk

function	description
getChunkSize(file)	function that returns the minimum chunk size to use when uploading the given file
createMultipartUpload(file)	function that calls S3 Multipart API to create a new upload
listParts(file, { uploadId, key })	function that calls S3 Multipart API to list the parts of a file that have already been uploaded
signPart(file, partData)	function that generates a signed URL for the specified part number
abortMultipartUpload(file, { uploadId, key })	function that calls S3 Multipart API to abort a Multipart upload, and removes all parts that have been uploaded
completeMultipartUpload(file, { uploadId, key, parts })	function that calls S3 Multipart API to complete a upload, combining all parts into a single object in bucket
allowedMetaFields: null	array of field names to limit the metadata fields that will be added to upload as query parameters
shouldUseMultipart(file)	boolean, or a function that returns a boolean which is called for each file that is uploaded with the corresponding UppyFile instance as argument

Transloadit https://uppy.io/docs/transloadit/

```
npm install @uppy/transloadit
```

```
import Transloadit from '@uppy/transloadit';

new Uppy().use(Transloadit, {
    assemblyOptions: {
        params: {
            auth: {
                key: 'your-transloadit-key' }, template_id: 'your-template-id',
            },
        },
   });
```

Options

option	default	description
id	Transloadit	unique identifier for this plugin
service	https://api2.transloadit.com	Transloadit API URL to use
limit	20	limit the amount of uploads at the same time
assemblyOptions	null	configure the Assembly Instructions, the fields to send along to the assembly, and authentication
waitForEncoding	false	wait for template to finish, rather than only the upload, before marking the upload complete
waitForMetadata	false	wait for Transloadit's backend to catch early errors, not the entire Assembly to complete
importFromUploadURLs	false	allow another plugin to upload files, and then import those files into Transloadit Assembly
alwaysRunAssembly	false	always create and run an Assembly when uppy.upload() is called, even if no files were selected

Events

You can use on *and* once *to listen to these events*

event	emitted
uppy.on('transloadit:assembly-created', (assembly, fileIDs) => {});	when an Assembly is created
uppy.on('transloadit:upload', (file, assembly) => {});	when Transloadit has received an upload (requires waitForMetadata to be set)
uppy.on('transloadit:assembly-executing', (assembly) => {});	when Transloadit has received all uploads, and is executing the Assembly
uppy.on('transloadit:result', (stepName, result, assembly) => {});	when a result came in from an Assembly (requires waitForEncoding to be set)
uppy.on('transloadit:complete', (assembly) => {});	when an Assembly completed (requires waitForEncoding to be set)

XHR https://uppy.io/docs/xhr-upload/

```
npm install @uppy/xhr-upload
```

```
import XHR from '@uppy/xhr-upload';

new Uppy().use(XHR, { endpoint: 'https://your-domain.com/upload’ });
```

Options

option	default	description
id	XHRUpload	unique identifier for this plugin
endpoint	null	URL of your server
method	post	which HTTP method to use for upload
formData	true	use a multipart form upload, using FormData
fieldName	‘files[]' if bundle option is true, otherwise it defaults to 'file'	when formData is true, this is used as the form field name for the file to be uploaded
allowedMetaFields	null	array of field names to limit the metadata fields added to upload • *empty array []* - not send any fields, • *['name']* - only send the ‘name’ field, • *null* - send all metadata fields
headers		HTTP headers to use for the upload request
bundle	false	send all files in a single multipart request
validateStatus	(status, responseText, response) => bool	check if the response was successful
getResponseData	(responseText, response) => void	extract response data from successful upload
getResponseError	(responseText, response) => void	extract error from failed upload
responseUrlFieldName	url	field name with the location of the uploaded file
timeout	30_000	abort the connection if no upload progress events have been received for this milliseconds amount
limit	5	maximum amount of files to upload in parallel
responseType	text	response type expected from server, determining how xhr.response property should be filled
withCredentials	false	whether cross-site Access-Control requests should be made using credentials

Golden Retriever https://uppy.io/docs/golden-retriever/ `npm install @uppy/golden-retriever`

```
import GoldenRetriever from '@uppy/golden-retriever';

new Uppy().use(GoldenRetriever);
```

by default, only IndexedDB storage will be used

Options

option	default	description
id	GoldenRetriever	unique identifier for this plugin
expires	24 hours	how long to store metadata and files, used for LocalStorage and IndexedDB
serviceWorker	false	enable Service Worker storage

Compressor https://uppy.io/docs/compressor/ `npm install @uppy/compressor`

```
import Compressor from '@uppy/compressor';

new Uppy().use(Compressor);
```

uses Compressor.js library

Options

option	default	description
id	Compressor	unique identifier for this plugin
quality	0.6	output image quality (for image/jpeg and image/webp)
limit	10	number of images that will be compressed in parallel

Events

You can use on *and* once *to listen to these events*

event	emitted
uppy.on('compressor:complete', (file, preview) => {});	when all files are compressed

Form https://uppy.io/docs/form/ `npm install @uppy/form`

```
import Form from '@uppy/form';

new Uppy().use(Form, { target: '#my-form' });
```

Options

option	default	description
id	Form	unique identifier for this plugin
target	null	DOM element or CSS selector for the form element
resultName	uppyResult	name attribute for the <input type="hidden"> where result be added
getMetaFromForm	true	extract metadata from the form
addResultToForm	true	add upload/encoding results back to the form in an <input name="uppyResult" type="hidden"> element
triggerUploadOnSubmit	false	start upload when the form is submitted
submitOnSuccess	false	submit the form after Uppy finishes uploading/encoding

Internationalization

https://uppy.io/docs/locales/

```
npm install @uppy/locales
```

```
import German from '@uppy/locales/lib/de_DE';

const uppy = new Uppy({ locale: German });
```

Overriding specific strings for a plugin

```
import Russian from '@uppy/locales/lib/ru_RU';

const uppy = new Uppy({ locale: Russian });
uppy.use(DragDrop, {
    target: '.UppyDragDrop',
    // override specific strings:
    locale: {
        strings: {
            browse: ‘custom string here’,
        },
    },
});
```

Locales

Arabic - Saudi Arabia	@uppy/locales/lib/ar_SA	Japanese - Japan	@uppy/locales/lib/ja_JP
Bulgarian - Bulgaria	@uppy/locales/lib/bg_BG	Korean - South Korea	@uppy/locales/lib/ko_KR
Chinese - China	@uppy/locales/lib/zh_CN	Norwegian Bokmål - Norway	@uppy/locales/lib/nb_NO
Chinese - Taiwan	@uppy/locales/lib/zh_TW	Persian - Iran	@uppy/locales/lib/fa_IR
Croatian - Croatia	@uppy/locales/lib/hr_HR	Polish - Poland	@uppy/locales/lib/pl_PL
Czech - Czechia	@uppy/locales/lib/cs_CZ	Portuguese - Brazil	@uppy/locales/lib/pt_BR
Danish - Denmark	@uppy/locales/lib/da_DK	Portuguese - Portugal	@uppy/locales/lib/pt_PT
Dutch - Netherlands	@uppy/locales/lib/nl_NL	Romanian - Romania	@uppy/locales/lib/ro_RO
English - United States	@uppy/locales/lib/en_US	Russian - Russia	@uppy/locales/lib/ru_RU
Finnish - Finland	@uppy/locales/lib/fi_FI	Serbian - Serbia(Cyrillic)	@uppy/locales/lib/sr_RS_Cyrillic
French - France	@uppy/locales/lib/fr_FR	Serbian - Serbia(Latin)	@uppy/locales/lib/sr_RS_Latin
Galician - Spain	@uppy/locales/lib/gl_ES	Slovak - Slovakia	@uppy/locales/lib/sk_SK
German - Germany	@uppy/locales/lib/de_DE	Spanish - Spain	@uppy/locales/lib/es_ES
Greek - Greece	@uppy/locales/lib/el_GR	Swedish - Sweden	@uppy/locales/lib/sv_SE
Hebrew - Israel	@uppy/locales/lib/he_IL	Thai - Thailand	@uppy/locales/lib/th_TH
Hungarian - Hungary	@uppy/locales/lib/hu_HU	Turkish - Turkey	@uppy/locales/lib/tr_TR
Icelandic - Iceland	@uppy/locales/lib/is_IS	Ukrainian - Ukraine	@uppy/locales/lib/uk_UA
Indonesian - Indonesia	@uppy/locales/lib/id_ID	Uzbek - Uzbekistan	@uppy/locales/lib/uz_UZ
Italian - Italy	@uppy/locales/lib/it_IT	Vietnamese - Vietnam	@uppy/locales/lib/vi_VN