# Uploadify v2.0.2


Developed by Ronnie Garcia and Travis Nickels
www.ronniesan.com

## WHAT IS IT?

This plug-in allows you to change any element with an ID attribute on your page into a single or multiple file upload tool. The plug-in uses a mix of JQuery, Flash, and a backend upload script in the language of your choice to send files from your local computer to your website server.

## HOW DO I IMPLEMENT IT?

Implementation of the plug-in is very easy and only relies on one JQuery function call to initiate.

1. Download the latest zip package from http://www.uploadify.com
2. Extract the files and upload them to your server (for the most basic installation, upload all files to the same folder).
3. Link the jQuery script, swfobject script and the Uploadify plug-in to the page you will be using it on. You can do so with the following code in the `<head>` section of your page:

```
<script type="text/javascript" src="/path/to/jquery-1.3.2.min.js"></script>
<script type=”text/javascript” src=”/path/to/sefobject.js"
<script type="text/javascript" src="/path/to/jquery.uploadify.v.2.0.2.min.js"></script>
```

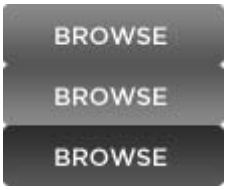
4. Add the call to the plug-in using the $.ready event in the `<head>` section of your page:

```
<script type="text/javascript">
$(document).ready(function() {
	$('#someID').uploadify({
		'uploader':  '/path/to/uploadify.swf',
		'script':    '/path/to/uploadify.php',
		'folder':    '/path/to/uploads-folder',
		'cancelImg': '/path/to/cancel.png'
	});
});
</script>
```

When the document is done loading, the elements that the function was called against will be hidden and replaced by the browse button. The component is now ready to use on your page.

## AVAILABLE OPTIONS

| | |
|---|---|
| **uploader** | The path to the uploader.swf file. For absolute paths prefix the path with either '/' or 'http'<br>Default = 'uploader.swf' |
| **script** | The path to the backend script that will be processing your uploaded files.<br>For absolute paths prefix the path with either '/' or 'http'<br>Default = 'upload.php' |
| **checkScript** | The relative path to the backend script that will check if the file selected already resides on the server. No Default. 'check.php' is provided with core files. |
| **scriptData** | An object containing name/value pairs of additional information you would like sent to the upload script. {'name': 'value'} |
| **fileDataName** | The name of your files array in the upload server script.<br>Default = 'Filedata' |
| **method**<br>new | Set the method for sending scriptData to the backend script. Either 'GET' or 'POST'. Default is set to 'POST'. |
| **scriptAccess** | The access mode for scripts in the flash file. If you are testing locally, set to 'always'.<br>Default = 'sameDomain' |
| **folder** | The path to the folder you would like to save the files to. Do not end the path with a '/'.<br>For absolute paths prefix the path with either '/' or 'http'. Note server security issues with trying to upload to remote destinations. |
| **queueID**<br>new | The ID of the element you want to use as your file queue. By default, one is created on the fly below the 'Browse' button. |
| **queueSizeLimit**<br>new | The limit of the number of items that can be in the queue at one time. Default = 999. |
| **multi** | Set to **true** if you want to allow multiple file uploads. |
| **auto** | Set to **true** if you would like the files to be uploaded when they are selected. |
| **fileDesc** | The text that will appear in the file type drop down at the bottom of the browse dialog box. |
| **fileExt** | A list of file extensions you would like to allow for upload. Format like '*.ext1;*. ext2;*.ext3'.<br>fileDesc is required when using this option. |
| **sizeLimit** | A number representing the limit in bytes for each upload. |
| **simUploadLimit**<br>changed | A limit to the number of simultaneous uploads you would like to allow.<br>Default:1. |
| **buttonText** | The text you would like to appear on the default button. Default = 'BROWSE' |
| **buttonImg** | The path to the image you will be using for the browse button. |
| **hideButton** | Set to **true** if you want to hide the button image. |

| | |
|---|---|
| rollover | Set to **true** if you would like to activate rollover states for your browse button. To prepare your browse button for rollover states, simple add the 'over' and 'press' states below the normal state in a single file. See below for an example:<br><br>  <br><br><br>*Mouseover events are inconsistent in Flash 9 so you may see a short lag when using the rollover option. |
| width | The width of the button image / flash file. Default = 30 |
| height | The height of the button image / flash file. If **rollover** is set to **true**, this should be 1/3 the height of the actual file. Default = 110 |
| wmode | Set to **transparent** to make the background of the flash file transparent. If set to **transparent**, the flash file will be at the top-most layer of the page. By omitting the **buttonImg** option and setting **wmode** to **transparent**, the entire flash file will be transparent, allowing you layer content below it or style the button using CSS. Default = 'opaque' |
| cancelImg | The path to the default cancel image. Default = 'cancel.png' |
| onInit | A function that triggers when the script is loaded. The default event handler hides the targeted element on the page and replaces it with the flash file, then creates a queue container after it. The default function will not trigger if the value of your custom function returns **false**. For custom functions, you can access the html for the flash file using the variable **flashElement**. |
| onSelect | A function that triggers for each element selected. The default event handler generates a 6 character random string as the unique identifier for the file item and creates a file queue item for the file. The default event handler will not trigger if the value of your custom function returns **false**.<br><br>Three arguments are passed to the function:<br>**event:** The event object.<br>**queueID:** The unique identifier of the file that was selected.<br>**fileObj:** An object containing details about the file that was selected.<br>• **name** – The name of the file<br>• **size** – The size in bytes of the file<br>• **creationDate** – The date the file was created<br>• **modificationDate** – The last date the file was modified<br>• **type** – The file extension beginning with a '.' |

| | |
|---|---|
| onSelectOnce | A function that triggers once for each select operation. There is no default event handler.<br><br>Two arguments are sent to the function:<br>**event:** The event object.<br>**data:** An object containing details about the select operation.<br>• **fileCount** – The total number of files in the queue<br>• **filesSelected** – The number of files selected in the select operation<br>• **filesReplaced** – The number of files that were replaced in the queue<br>• **allBytesTotal** – The total number of bytes for all files in the queue |
| onCancel | A function that triggers when a file upload is cancelled or removed from the queue. The default event handler removes the file from the upload queue. The default event handler will not trigger if the value of your custom function returns **false**.<br><br>Four arguments are sent to the function:<br>**event:** The event object.<br>**queueID:** The unique identifier of the file that was cancelled.<br>**fileObj:** An object containing details about the file that was selected.<br>• **name** – The name of the file<br>• **size** – The size in bytes of the file<br>• **creationDate** – The date the file was created<br>• **modificationDate** – The last date the file was modified<br>• **type** – The file extension beginning with a '.'<br>**data:** Details about the file queue.<br>• **fileCount** – The total number of files left in the queue<br>• **allBytesTotal** – The total number of bytes left for all files in the queue |
| onClearQueue | A function that triggers when the **fileUploadClearQueue** function is called. The default event handler removes all queue items from the upload queue. The default event handler will not trigger if the value of your custom function returns **false**.<br><br>Two arguments are sent to the function:<br>**event:** The event object.<br>**data:** An object containing details about the file queue.<br>• **fileCount** – The number of files remaining in the upload queue<br>• **allBytesTotal** – The number of bytes remaining in the upload queue |
| onQueueFull<br>new | A function that triggers when the file queue has reached maximum capacity. The default event alerts the user of the queue size.<br><br>Two arguments are sent to the function:<br>• **event** - The event object.•<br>• **queueSizeLimit** - The maximum size of the queue. |

| | |
|---|---|
| onError<br>changed | A function that triggers when an error occurs during the upload process. The default event handler attaches an error message to the queue item returning the error and changes it's queue item container to red.<br><br>Four arguments are sent to the function:<br>**event:** The event object.<br>**queueID:** The unique identifier of the file that was cancelled.<br>**fileObj:** An object containing details about the file that was selected.<br>• **name** – The name of the file<br>• **size** – The size in bytes of the file<br>• **creationDate** – The date the file was created<br>• **modificationDate** – The last date the file was modified<br>• **type** – The file extension beginning with a '.'<br>**errorObj:** An object containing details about the error returned.<br>• **type** – Either 'HTTP', 'IO', or 'Security'<br>• **info** – An error message describing the type of error returned |
| onProgress | A function that fires each time the progress of a file upload changes. The default function updates the progress bar in the file queue item. The default function will not trigger if the value of your custom function returns **false**.<br><br>Four arguments are sent to function:<br>**event:** The event object.<br>**queueID:** The unique identifier of the file that was cancelled.<br>**fileObj:** An object containing details about the file that was selected.<br>• **name** – The name of the file<br>• **size** – The size in bytes of the file<br>• **creationDate** – The date the file was created<br>• **modificationDate** – The last date the file was modified<br>• **type** – The file extension beginning with a '.'<br>**data:** An object containing details about the upload and queue.<br>• **percentage** – The current percentage completed for the upload<br>• **bytesLoaded** – The current amount of bytes uploaded<br>• **allBytesLoaded** – The current amount of bytes loaded for all files in the queue<br>• **speed** – The current upload speed in KB/s |
| onComplete | A function that triggers when a file upload has completed. The default function removes the file queue item from the upload queue. The default function will not trigger if the value of your custom function returns **false**.<br><br>Four arguments are sent to the function:<br>**event:** The event object.<br>**queueID:** The unique identifier of the file that was cancelled.<br>**fileObj:** An object containing details about the file that was selected.<br>• **name** – The name of the file<br>• **filepath** – The path on the server to the uploaded file<br>• **size** – The size in bytes of the file<br>• **creationDate** – The date the file was created<br>• **modificationDate** – The last date the file was modified<br>• **type** – The file extension beginning with a '.'<br>**response:** The data sent back from the server.<br>**data:** Details about the file queue.<br>• **fileCount** – The total number of files left in the queue |

| | |
|---|---|
| | • **speed** – The average speed of the file upload in KB/s |
| onAllComplete | A function that triggers when all file uploads have completed. There is no default event handler.<br><br>Two arguments are sent to the function:<br>**event:** The event object.<br>**data:** An object containing details about the upload process.<br>• **filesUploaded** – The total number of files uploaded<br>• **errors** – The total number of errors while uploading<br>• **allbytesLoaded** – The total number of bytes uploaded<br>• **speed** – The average speed of all uploaded files |
| onCheck | A function that triggers when an existing file is detected on the server. The default event handler opens a confirmation box.<br><br>Five arguments are sent to the function:<br>**event:** The event object.<br>**checkScript:** The path to the file checking script.<br>**fileQueue:** A file queue object consisting of key/value pairs with the queue ID as the key and the filename as the value.<br>**folder:** The path to the upload folder.<br>**single:** True if only one file is being uploaded from the queue. |

## RELATED FUNCTIONS

**uploadifySettings**
changed

A function used to change options for a fileUpload object.

Two arguments are available, one is required:•
**Setting** - The option to change.•
**Value** - The value to change the option to. If left blank, this function will return the current value of the setting.

The following example will change the upload folder to '/uploads':

```
$('#someID').uploadifySettings('folder','/uploads');
```

The following example will return the simUploadLimit:

```
$(' #someID').uploadifySettings('simUploadLimit');
```

The following options can be changed:•

buttonImg, buttonText, cancelImg, fileDesc, fileExt, width, height, folder, script, scriptData, simUploadLimit, sizeLimit, hideButton

The following will return the current size of the queue

```
#('#someID').uploadifySettings('queueSize');
```

When using scriptData enter it in {key1 : value1, key2 : value2, …} format.

```
#('#someID').uploadifySettings('scriptData', {'name' : some.val()});
```

If the key already exists it will update it, if it doesn't exist it will add it. You no longer need to re-write every key/value pair. When requesting scriptData it will return the key/value pairs as an object.

**uploadifyUpload**
changed

A function used to begin an upload of a single file or all files in the queue.

One argument is optional:
**queueID:** The unqiue queue identifier for the file to upload.

The following example will upload all files in the queue:

```
$('#someID').uploadifyUpload();
```

**uploadifyCancel**
changed

A function used to remove a file from the queue or stop an upload in progress.

One argument is required.
**queueID:** The unique queue identifier of the file you wish to cancel.

The following example will remove a file from the upload queue:

```
$('#someID').uploadifyCancel('NFJSHS');
```

**uploadifyClearQueue**
changed

A function used to clear all files from the upload queue.

The following example clears the file upload queue:

```
$('#someID').uploadifyClearQueue();
```