

Instantiating the Uploader

```
<div id="myUploader"
  style="width:400px; height:210px"></div>
<script>
var myUploader = new YAHOO.widget.Uploader("myUploader");
</script>
```

Instantiates a new Uploader object, `myUploader`, which is bound to a div whose id attribute is `'myUploader'`. The result will be a visual Uploader display — a console-style window that outputs progress messages. To hide the visual display, simply set the size of the placeholder div to `height:0; width:0; overflow:hidden;`.

Constructor

```
YAHOO.widget.LogReader(str html id);
```

Arguments:

- (1) **HTML element (string or object):** A reference to an HTML id string or element object binds the Uploader to an existing page element. This parameter is required.

Limitations

- (1) The Uploader can only send data to servers in the same security sandbox as the `uploader.swf` file. If `uploader.swf` hosted by `yui.yahooapis.com` is used, then the server must contain a cross-domain permissions file allowing `yui.yahooapis.com` to upload files. By default, Uploader looks for `uploader.swf` in the assets directory in the same path as `uploader.js`, so if you're loading Uploader from `yui.yahooapis.com` you will be affected by this issue.
- (2) Because of a known Flash bug, the Uploader running in Firefox in Windows does not send the correct cookies with the upload; instead of sending Firefox cookies, it sends Internet Explorer's cookies for the respective domain. As a workaround, we suggest either using a cookieless upload method or appending `document.cookie` to the upload request.

Dependencies

Uploader requires the Yahoo Global Object, Dom, Event and Element. The Uploader also uses the `uploader.swf` file that by default resides in the `assets` directory in the same path as `uploader.js`. On the client side, Uploader requires that the user have **Flash 9.0.45** or later installed on their browser.

Simple Use Case

```
myUploader.addEventListener("fileSelect", onFileSelect);
myUploader.browse(false, [{description:"Images",
  extensions:"*.jpg, *.gif"}]);
function onFileSelect (event:Object) {
  myUploader.uploadAll("YOUR UPLOAD URL");
}
```

Opens the File Select dialog and allows the user to select one file (parameter `allowMultiple` is set to `false`) with either `jpg` or `gif` extension (the filtering is suggestive, rather than strict – the user can override it by specifying `"*.*"` in the text input of the File Select dialog. When files are selected, they are queued and uploaded to the specified URL.

Solutions

Open the File Select dialog with multiple file selection and filters for different types of images or videos:

```
myUploader.browse(true, [{description:"Images",
  extensions:"*.jpg, *.png, *.gif, *.bmp"},
 {description:"Videos", extensions:"*.avi, *.mpg,
  *.flv, *.mov"}]);
```

Track upload progress and log it in the YUI Logger (must be included on the page):

```
myUploader.addListener("uploadProgress",
  onUploadProgress);
function onUploadProgress (event:Object) {
  YAHOO.log(event.id + ": " + event.bytesLoaded +
  "/" + event.bytesTotal);
}
```

Send custom variables in the same POST request as the file submission:

```
myUploader.upload("file0", "YOUR UPLOAD URL",
  "POST", {var1: "foo", var2: "bar", var3: "baz"});
```

Modify the file form field name from the default "Filedata":

```
myUploader.upload("file0", "YOUR UPLOAD URL", null,
  null, "DifferentFileFieldName");
```

Accept the file upload using PHP on the server side:

```
<?php
foreach ($_FILES as $fieldName => $file) {
  move_uploaded_file($file['tmp_name'],
    "." . $file['name']);
}
echo (" ");
} exit();?>
```

YAHOO.widget.Uploader
Properties:

SWFURL (s)

YAHOO.widget.Uploader
Events:

contentReady
fileSelect
uploadStart
uploadProgress
uploadCancel
uploadComplete
uploadCompleteData
uploadError

YAHOO.widget.Uploader
Methods:

browse()
upload()
uploadAll()
cancel()
clearFileList()
removeFile()