plupload官网：<https://www.plupload.com/>

文档地址：<https://www.plupload.com/docs/>

下载地址：https://www.plupload.com/download/

# 基本实例方法

1、init()：初始化Plupload实例，只有初始化完成后才能使用

 url : "/examples/upload",

2、setOption(option, [value])：设置特定的配置参数，对象信息，option：参数名称、value：参数值

up.setOption({  
 'url': host,  
 'multipart\_params': new\_multipart\_params  
});

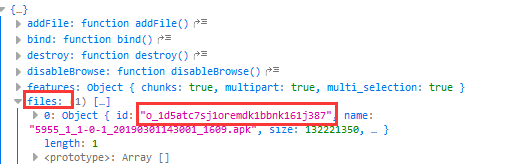
3、start()：点击上传按钮，或直接调用，开始上传上传队列中的内容

up.start();

document.getElementById('postfiles').onclick = function() {  
 up.start();

};

4、getFile(id)：通过id来获取文件对象，这个id不好获取

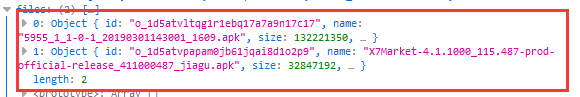


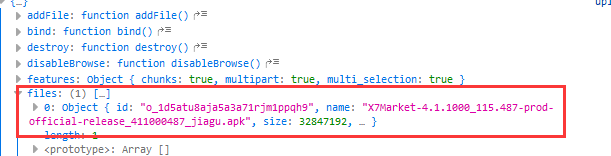
**文件没选择一次，这个队列就会增长一次，实现单文件上传得清除队列中的内容，只保留单条数据**

5、addFile(file, [fileName])：在上传队列中添加文件，添加成功触发**FileAdded**事件，file：**原生文件|plupload对象|input[type=’file’]**，filename：文件名称，队列中存在数据，就可以使用start进行上传

6、removeFile(file)：上传队列中移除文件

7、splice(start, length)：在上传队列中移除文件，实现单文件上传，每次选择文件后，需要使第一个文件出列，需要判断当前队列长度





**不能直接使用files.length判断队列的长度**，这个是每次选择文件的长度，如果一次选择3个文件上传，这个值会是3，而不是队列实际的长度

判断队列实际长度：**up.files.length，必须指定的对象下的files**

单文件上传判断方法：

if(up.files.length > 1){//获取队列中数据的长度  
 up.splice(0,1);//单个文件上传，去除第一个文件即可  
}

8、bind(name, func, scope)：绑定监听函数。name：事件名、func：函数、scope：监听函数作用域，也就是函数中的this指向

up.bind(‘FileAdded’,function(up,files){});

# 基本使用事件

1、Init：当Plupload初始化完成后触发监听函数参数**(uploader)**

2、PostInit：当Init事件发生后触发监听函数参数**(uploader)**

3、UploadFile：当上传队列中某一个文件开始上传后触发监听函数参数：**(uploader,file)，uploader为当前的plupload对象，file触发事件的文件对象**

4、BeforeUpload：当队列中的某一个文件正要开始上传前触发监听函数参数：**(uploader,file)，在这个函数中也可以设置option参数**

**5、**QueueChanged：当上传队列发生变化后触发，即上传队列新增了文件或移除了文件**(uploader)**

6、UploadProgress：会在文件上传过程中不断触发，可以用此事件来显示上传进度监听函数参数：**(uploader,file)。这里的file为单个文件，可以获取到id等信息**

7、FilesAdded：当文件添加到上传队列后触发监听函数参数：**(uploader,files)。files为数组，本次添加的数据个数，在这个函数中可以进行队列操作**

8、Error：当发生错误时触发监听函数参数：**(uploader,errObject)**



9、destroy()：销毁plupload实例

# 基本配置参数

{

'key' : g\_object\_name,

'policy': policyBase64,

'OSSAccessKeyId': accessid,

'success\_action\_status' : '200', //让服务端返回200,不然，默认会返回204

'callback' : callbackbody,

'signature': signature,

};

1、browse\_button：点击选择文件，添加队列。可以是对象或id

2、url：服务器端接收和处理上传文件的脚本地址

3、filters：上传限制，过滤



4、multipart：是否可以选择多个文件，true：是，false：否

5、multipart\_params：特殊的附加参数

{

'key' : g\_object\_name,

'policy': policyBase64,

'OSSAccessKeyId': accessid,

'success\_action\_status' : '200', //让服务端返回200,不然，默认会返回204

'callback' : callbackbody,

'signature': signature,

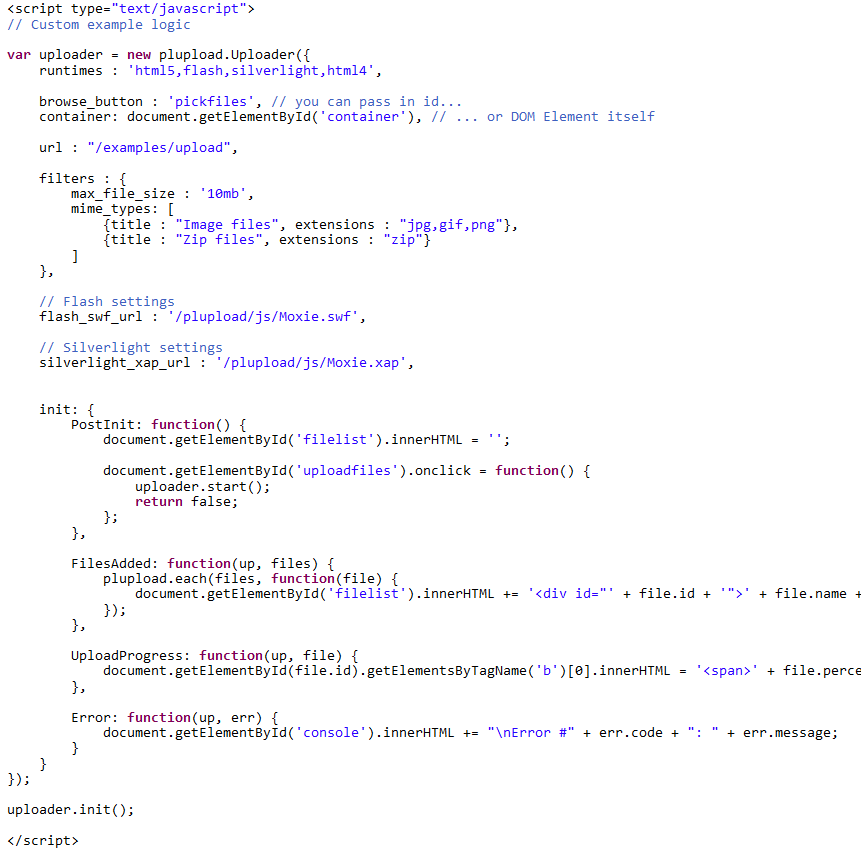
};

6、flash\_swf\_url：flash上传组件的url地址，如果是相对路径，则相对的是调用Plupload的html文档。当使用flash上传方式会用到该参数。lib/plupload-2.1.2/js/Moxie.swf

7、silverlight\_xap\_url：silverlight上传组件的url地址，如果是相对路径，则相对的是调用Plupload的html文档。当使用silverlight上传方式会用到该参数。lib/plupload-2.1.2/js/Moxie.xap

8、runtimes：用来指定上传方式，指定多个上传方式请使用逗号隔开，html5,flash,silverlight,html4

9、init：文档上没有找到，但案例中都有这个配，配置下进行了上传事件监听，不用单独绑定事件



**Plupload上传插件中文帮助文档**

**配置参数**

实例化一个plupload对象时，也就是 new plupload.Uploader()，需要传入一个对象作为配置参数。后面内容中出现的plupload实例均是指new plupload.Uploader()得到的实例对象

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **默认值** | **描述** |
| **browse\_button** | String / DOM |  | 触发文件选择对话框的DOM元素，当点击该元素后便后弹出文件选择对话框。该值可以是DOM元素对象本身，也可以是该DOM元素的id |
| **url** | String |  | 服务器端接收和处理上传文件的脚本地址，可以是相对路径(相对于当前调用Plupload的文档)，也可以是绝对路径 |
| **filters** | Object | { } | 可以使用该参数来限制上传文件的类型，大小等，该参数以对象的形式传入，它包括三个属性：**mime\_types：**用来限定上传文件的类型，为一个数组，该数组的每个元素又是一个对象，该对象有title和extensions两个属性，title为该过滤器的名称，extensions为文件扩展名，有多个时用逗号隔开。该属性默认为一个空数组，即不做限制。  **max\_file\_size：**用来限定上传文件的大小，如果文件体积超过了该值，则不能被选取。值可以为一个数字，单位为b,也可以是一个字符串，由数字和单位组成，如’200kb’  **prevent\_duplicates：**是否允许选取重复的文件，为true时表示不允许，为false时表示允许，默认为false。如果两个文件的文件名和大小都相同，则会被认为是重复的文件  filters完整的配置示例如下(当然也可以只配置其中的某一项)：  filters: {  mime\_types : [ //只允许上传图片和zip文件  { title : "Image files", extensions : "jpg,gif,png" },  { title : "Zip files", extensions : "zip" }  ],  max\_file\_size : '400kb', //最大只能上传400kb的文件  prevent\_duplicates : true //不允许选取重复文件  } |
| **headers** | Object |  | 设置上传时的自定义头信息，以键/值对的形式传入，键代表头信息属性名，键代表属性值。html4上传方式不支持设置该属性。 |
| **multipart** | Boolean | true | 为true时将以multipart/form-data的形式来上传文件，为false 时则以二进制的格式来上传文件。html4上传方式不支持以二进制格式来上传文件，在flash上传方式中，二进制上传也有点问题。并且二进制格式上传还 需要在服务器端做特殊的处理。一般我们用multipart/form-data的形式来上传文件就足够了。 |
| **multipart\_params** | Object |  | 上传时的附加参数，以键/值对的形式传入，服务器端可是使用$\_POST来获取这些参数(以php为例)。比如：  multipart\_params: {  one: '1',  two: '2',  three: { //值还可以是一个字面量对象  a: '4',  b: '5'  },  four: ['6', '7', '8'] //也可以是一个数组  } |
| **max\_retries** | Number | 0 | 当发生plupload.HTTP\_ERROR错误时的重试次数，为0时表示不重试 |
| **chunk\_size** | Number/String | 0 | 分片上传文件时，每片文件被切割成的大小，为数字时单位为字节。也可以使用一个带单位的字符串，如"200kb"。当该值为0时表示不使用分片上传功能 |
| **resize** | Object |  | 可以使用该参数对将要上传的图片进行压缩，该参数是一个对象，里面包括5个属性：**width：**指定压缩后图片的宽度，如果没有设置该属性则默认为原始图片的宽度  **height：**指定压缩后图片的高度，如果没有设置该属性则默认为原始图片的高度  **crop：**是否裁剪图片  **quality：**压缩后图片的质量，只对jpg格式的图片有效，默认为90。quality可以跟width和height一起使用，但也可以单独使用，单独使用时，压缩后图片的宽高不会变化，但由于质量降低了，所以体积也会变小  **preserve\_headers：**压缩后是否保留图片的元数据，true为保留，false为不保留,默认为true。删除图片的元数据能使图片的体积减小一点点  resize参数的配置示例如下：  resize: {  width: 100,  height: 100,  crop: true,  quality: 60,  preserve\_headers: false  } |
| **drop\_element** | DOM/String/Array |  | 指定了使用拖拽方式来选择上传文件时的拖拽区域，即可以把文件拖拽到这个区域的方式来选择文件。该参数的值可以为一个DOM元素的id,也可是 DOM元素本身，还可以是一个包括多个DOM元素的数组。如果不设置该参数则拖拽上传功能不可用。目前只有html5上传方式才支持拖拽上传。 |
| **multi\_selection** | Boolean | true | 是否可以在文件浏览对话框中选择多个文件，true为可以，false为不可以。默认true，即可以选择多个文件。需要注意的是，在某些不支持 多选文件的环境中，默认值是false。比如在ios7的safari浏览器中，由于存在bug，造成不能多选文件。当然，在html4上传方式中，也是 无法多选文件的。 |
| **required\_features** | Mix |  | 可以使用该参数来设置你必须需要的一些功能特征，Plupload会根据你的设置来选择合适的上传方式。因为，不同的上传方式，支持的功能是不同 的，比如拖拽上传只有html5上传方式支持，图片压缩则只有html5,flash,silverlight上传方式支持。该参数的值是一个混合类型， 可以是一个以逗号分隔的字符串， |
| **unique\_names** | Boolean | false | 当值为true时会为每个上传的文件生成一个唯一的文件名，并作为额外的参数post到服务器端，参数明为name,值为生成的文件名。 |
| **runtimes** | String | html5,flash, silverlight, html4 | 用来指定上传方式，指定多个上传方式请使用逗号隔开。一般情况下，你不需要配置该参数，因为Plupload默认会根据你的其他的参数配置来选择 最合适的上传方式。如果没有特殊要求的话，Plupload会首先选择html5上传方式，如果浏览器不支持html5，则会使用flash或 silverlight，如果前面两者也都不支持，则会使用最传统的html4上传方式。如果你想指定使用某个上传方式，或改变上传方式的优先顺序，则你 可以配置该参数。 |
| **file\_data\_name** | String | file | 指定文件上传时文件域的名称，默认为file,例如在php中你可以使用$\_FILES['file']来获取上传的文件信息 |
| **container** | DOM/String |  | 用来指定Plupload所创建的html结构的父容器，默认为前面指定的browse\_button的父元素。该参数的值可以是一个元素的id,也可以是DOM元素本身。 |
| **flash\_swf\_url** | String | js/Moxie.swf | flash上传组件的url地址，如果是相对路径，则相对的是调用Plupload的html文档。当使用flash上传方式会用到该参数。 |
| **silverlight\_xap\_url** | String | js/Moxie.xap | silverlight上传组件的url地址，如果是相对路径，则相对的是调用Plupload的html文档。当使用silverlight上传方式会用到该参数。 |

**各种事件说明**

要了解plupload的运行状况，靠的就是在这些事件了

|  |
| --- |
| **Init** |
| 当Plupload初始化完成后触发监听函数参数：**(uploader)**  uploader为当前的plupload实例对象 |
| **PostInit** |
| 当Init事件发生后触发监听函数参数：**(uploader)**  uploader为当前的plupload实例对象 |
| **OptionChanged** |
| 当使用Plupload实例的setOption()方法改变当前配置参数后触发监听函数参数：**(uploader,option\_name,new\_value,old\_value)**  uploader为当前的plupload实例对象，option\_name为发生改变的参数名称，new\_value为改变后的值，old\_value为改变前的值 |
| **Refresh** |
| 当调用plupload实例的refresh()方法后会触发该事件，暂时不清楚还有什么其他动作会触发该事件，但据我测试，把文件添加到上传队列后也会触发该事件。监听函数参数：**(uploader)**  uploader为当前的plupload实例对象 |
| **StateChanged** |
| 当上传队列的状态发生改变时触发监听函数参数：**(uploader)**  uploader为当前的plupload实例对象 |
| **UploadFile** |
| 当上传队列中某一个文件开始上传后触发监听函数参数：**(uploader,file)**  uploader为当前的plupload实例对象，file为触发此事件的文件对象 |
| **BeforeUpload** |
| 当队列中的某一个文件正要开始上传前触发监听函数参数：**(uploader,file)**  uploader为当前的plupload实例对象，file为触发此事件的文件对象 |
| **QueueChanged** |
| 当上传队列发生变化后触发，即上传队列新增了文件或移除了文件。QueueChanged事件会比FilesAdded或FilesRemoved事件先触发监听函数参数：**(uploader)**  uploader为当前的plupload实例对象 |
| **UploadProgress** |
| 会在文件上传过程中不断触发，可以用此事件来显示上传进度监听函数参数：**(uploader,file)**  uploader为当前的plupload实例对象，file为触发此事件的文件对象 |
| **FilesRemoved** |
| 当文件从上传队列移除后触发监听函数参数：**(uploader,files)**  uploader为当前的plupload实例对象，files为一个数组，里面的元素为本次事件所移除的文件对象 |
| **FileFiltered** |
| 暂不清楚该事件的意义，但根据测试得出，该事件会在每一个文件被添加到上传队列前触发监听函数参数：**(uploader,file)**  uploader为当前的plupload实例对象，file为触发此事件的文件对象 |
| **FilesAdded** |
| 当文件添加到上传队列后触发监听函数参数：**(uploader,files)**  uploader为当前的plupload实例对象，files为一个数组，里面的元素为本次添加到上传队列里的文件对象 |
| **FileUploaded** |
| 当队列中的某一个文件上传完成后触发监听函数参数：**(uploader,file,responseObject)**  uploader为当前的plupload实例对象，file为触发此事件的文件对象，responseObject为服务器返回的信息对象，它有以下3个属性：  **response：**服务器返回的文本  **responseHeaders：**服务器返回的头信息  **status：**服务器返回的http状态码，比如200 |
| **ChunkUploaded** |
| 当使用文件小片上传功能时，每一个小片上传完成后触发监听函数参数：**(uploader,file,responseObject)**  uploader为当前的plupload实例对象，file为触发此事件的文件对象，responseObject为服务器返回的信息对象，它有以下5个属性：  **offset：**该文件小片在整体文件中的偏移量  **response：**服务器返回的文本  **responseHeaders：**服务器返回的头信息  **status：**服务器返回的http状态码，比如200  **total：**该文件(指的是被切割成了许多文件小片的那个文件)的总大小，单位为字节 |
| **UploadComplete** |
| 当上传队列中所有文件都上传完成后触发监听函数参数：**(uploader,files)**  uploader为当前的plupload实例对象，files为一个数组，里面的元素为本次已完成上传的所有文件对象 |
| **Error** |
| 当发生错误时触发监听函数参数：**(uploader,errObject)**  uploader为当前的plupload实例对象，errObject为错误对象，它至少包含以下3个属性(因为不同类型的错误，属性可能会不同)：  **code：**错误代码，具体请参考plupload上定义的表示错误代码的常量属性  **file：**与该错误相关的文件对象  **message：**错误信息 |
| **Destroy** |
| 当调用destroy方法时触发监听函数参数：**(uploader)**  uploader为当前的plupload实例对象 |

**Plupload实例的属性**

|  |  |
| --- | --- |
| **属性** | **描述** |
| **id** | Plupload实例的唯一标识id |
| **state** | 当前的上传状态，可能的值为plupload.STARTED或plupload.STOPPED，该值由Plupload实例的stop()或statr()方法控制。默认为plupload.STOPPED |
| **runtime** | 当前使用的上传方式 |
| **files** | 当前的上传队列，是一个由上传队列中的文件对象组成的数组 |
| **settings** | 当前的配置参数对象 |
| **total** | 表示总体进度信息的QueueProgress对象 |

**四、Plupload实例的方法**

|  |  |
| --- | --- |
| **方法** | **描述** |
| **init()** | 初始化Plupload实例 |
| **setOption(option, [value])** | 设置某个特定的配置参数,option为参数名称，value为要设置的参数值。option也可以为一个由参数名和参数值键/值对组成的对象，这样就可以一次设定多个参数，此时该方法的第二个参数value会被忽略。 |
| **getOption([option])** | 获取当前的配置参数，参数option为需要获取的配置参数名称，如果没有指定option，则会获取所有的配置参数 |
| **refresh()** | 刷新当前的plupload实例，暂时还不明白什么时候需要使用 |
| **start()** | 开始上传队列中的文件 |
| **stop()** | 停止队列中的文件上传 |
| **disableBrowse(disable)** | 禁用或启用plupload的文件浏览按钮,参数disable为true时为禁用，为false时为启用。默认为true |
| **getFile(id)** | 通过id来获取文件对象 |
| **addFile(file, [fileName])** | 向上传队列中添加文件，如果成功添加了文件，会触发FilesAdded事件。参数file为要添加的文件,可以是一个原生的文件,或者一个plupload文件对象,或者一个input[type="file"]元素,还可以是一个包括前面那几种东西的数组；fileName为给该文件指定的名称 |
| **removeFile(file)** | 从上传队列中移除文件，参数file为plupload文件对象或先前指定的文件名称 |
| **splice(start, length)** | 从上传队列中移除一部分文件，start为开始移除文件在队列中的索引，length为要移除的文件的数量，该方法的返回值为被移除的文件。该方法会触发FilesRemoved 和QueueChanged事件 |
| **trigger(name, Multiple)** | 触发某个事件。name为要触发的事件名称，Multiple为传给该事件监听函数的参数，是一个对象 |
| **hasEventListener(name)** | 用来判断某个事件是否有监听函数，name为事件名称 |
| **bind(name, func, scope)** | 给某个事件绑定监听函数，name为事件名，func为监听函数，scope为监听函数的作用域，也就是监听函数中this的指向 |
| **unbind(name, func)** | 移除事件的监听函数，name为事件名称，func为要移除的监听函数 |
| **unbindAll()** | 移除所有事件的所有监听函数 |
| **destroy()** | 销毁plupload实例 |

**文件对象的属性和方法**

在很多事件监听函数中，都会提供文件对象给你

|  |  |
| --- | --- |
| **属性/方法** | **描述** |
| **id** | 文件id |
| **name** | 文件名，例如”myfile.gif” |
| **type** | 文件类型，例如”image/jpeg” |
| **size** | 文件大小，单位为字节，当启用了客户端压缩功能后，该值可能会改变 |
| **origSize** | 文件的原始大小，单位为字节 |
| **loaded** | 文件已上传部分的大小，单位为字节 |
| **percent** | 文件已上传部分所占的百分比，如50就代表已上传了50% |
| **status** | 文件的状态，可能为以下几个值之一：plupload.QUEUED, plupload.UPLOADING, plupload.FAILED, plupload.DONE |
| **lastModifiedDate** | 文件最后修改的时间 |
| **getNative()** | 获取原生的文件对象 |
| **getSource()** | 获取mOxie.File 对象，想了解mOxie是什么东西，可以看下https://github.com/moxiecode/moxie/wiki/API |
| **destroy()** | 销毁文件对象 |

**QueueProgress 对象的属性**

plupload实例的total属性是一个QueueProgress对象

|  |  |
| --- | --- |
| **属性** | **描述** |
| **size** | 上传队列中所有文件加起来的总大小，单位为字节 |
| **loaded** | 队列中当前已上传文件加起来的总大小,单位为字节 |
| **uploaded** | 已完成上传的文件的数量 |
| **failed** | 上传失败的文件数量 |
| **queued** | 队列中剩下的(也就是除开已经完成上传的文件)需要上传的文件数量 |
| **percent** | 整个队列的已上传百分比，如50就代表50% |
| **bytesPerSec** | 上传速率，单位为 byte/s，也就是 字节/秒 |

**Plupload命名空间上的属性**

Plupload的命名空间上有一些属性，用来表示一些常量。记住，不是plupload实例的属性，而是plupload的属性

|  |  |
| --- | --- |
| **属性名称** | **描述** |
| **VERSION** | 当前plupload的版本号 |
| **STOPPED** | 值为1，代表上传队列还未开始上传或者上传队列中的文件已经上传完毕时plupload实例的state属性值 |
| **STARTED** | 值为2，代表队列中的文件正在上传时plupload实例的state属性值 |
| **QUEUED** | 值为1，代表某个文件已经被添加进队列等待上传时该文件对象的status属性值 |
| **UPLOADING** | 值为2，代表某个文件正在上传时该文件对象的status属性值 |
| **FAILED** | 值为4，代表某个文件上传失败后该文件对象的status属性值 |
| **DONE** | 值为5，代表某个文件上传成功后该文件对象的status属性值 |
| **GENERIC\_ERROR** | 值为-100，发生通用错误时的错误代码 |
| **HTTP\_ERROR** | 值为-200，发生http网络错误时的错误代码，例如服务气端返回的状态码不是200 |
| **IO\_ERROR** | 值为-300，发生磁盘读写错误时的错误代码，例如本地上某个文件不可读 |
| **SECURITY\_ERROR** | 值为-400，发生因为安全问题而产生的错误时的错误代码 |
| **INIT\_ERROR** | 值为-500，初始化时发生错误的错误代码 |
| **FILE\_SIZE\_ERROR** | 值为-600，当选择的文件太大时的错误代码 |
| **FILE\_EXTENSION\_ERROR** | 值为-601，当选择的文件类型不符合要求时的错误代码 |
| **FILE\_DUPLICATE\_ERROR** | 值为-602，当选取了重复的文件而配置中又不允许有重复文件时的错误代码 |
| **IMAGE\_FORMAT\_ERROR** | 值为-700，发生图片格式错误时的错误代码 |
| **IMAGE\_MEMORY\_ERROR** | 当发生内存错误时的错误代码 |
| **IMAGE\_DIMENSIONS\_ERROR** | 值为-702，当文件大小超过了plupload所能处理的最大值时的错误代码 |