文件上传的渐进式增强

作者： [阮一峰](http://www.ruanyifeng.com/)

日期： [2012年8月10日](http://www.ruanyifeng.com/blog/2012/08/)

文件上传是最古老的互联网操作之一。

20多年了，它几乎没变，还是原来的样子：操作麻烦、缺乏交互、用户体验不佳。在这个新技术日新月异的时代，显得非常落伍。



网页开发者们想了很多办法，试图提升文件上传的功能和操作体验，在各种Javascript库的基础上，开发了五花八门的[插件](http://superdit.com/2010/06/29/10-jquery-ajax-file-uploader-plugins/" \t "_blank)。可是，由于不同浏览器之间的差异，缺乏统一接口，这些插件要么用起来很麻烦，要么不能普遍适用。

HTML5提供了一系列新的浏览器API，使得文件上传有可能出现革命性变化。英国程序员[Remy Sharp](http://remysharp.com/" \t "_blank)总结了这些[新的接口](http://html5doctor.com/drag-and-drop-to-server/" \t "_blank)，本文在他的文章基础上，讨论如何采用HTML5的API，对文件上传进行渐进式增强，实现以下功能：

　　\* iframe上传

　　\* ajax上传

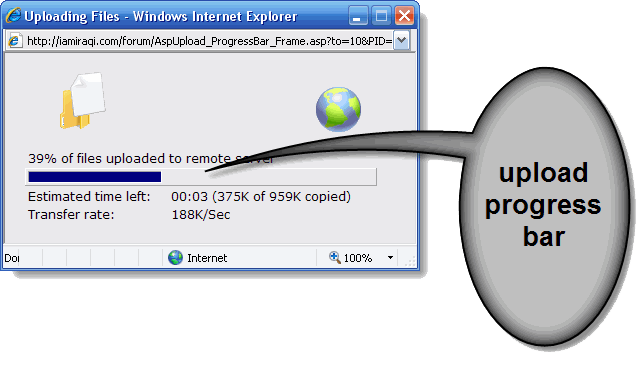
　　\* 进度条

　　\* 文件预览

　　\* 拖放上传

为了对这些功能有一个感性认识，你可以先看看Remy Sharp提供的[范例](http://html5demos.com/dnd-upload" \t "_blank)。

虽然这些API，还没有得到广泛部署，但它们是未来的潮流。有了它们，代码就可以写得非常优雅简洁，上面五个功能都能在20行以内实现。



**一、传统形式**

让我们从最基本的开始。

文件上传的传统形式，是使用表单元素file：

　　<form id="upload-form" action="upload.php" method="post" enctype="multipart/form-data" >

**<input type="file" id="upload" name="upload" /> <br />**

　　　　<input type="submit" value="Upload" />

　　</form>

所有浏览器都支持上面的代码。它在IE浏览器中，显示如下：



用户先选择文件，然后点击"Upload"按钮，文件开始上传。

**二、iframe上传**

传统的表单上传，属于"同步上传"。也就是说，点击上传按钮后，网页"锁死"，用户只能等待上传结束，然后浏览器刷新，跳到表单的action属性指定的网址。

有没有办法"异步上传"，在网页不重载的情况下，完成整个上传过程呢？

在HTML5没有出现之前，只能使用iframe做到这一点。用户点击submit时，动态插入一个iframe元素（以下代码使用了jQuery函数库）。

　　var form = $("#upload-form");

　　form.on('submit',function() {

　　　　// 此处动态插入iframe元素

　　});

插入iframe的代码如下：

　　var seed = Math.floor(Math.random() \* 1000);

　　var id = "uploader-frame-" + seed;

　　var callback = "uploader-cb-" + seed;

　　var iframe = $('<iframe id="'+id+'" name="'+id+'" style="display:none;">');

　　var url = form.attr('action');

**form.attr('target', id).append(iframe).attr('action', url + '?iframe=' + callback);**

最后一行，有两个地方值得注意。首先，它为表单添加target属性，指向动态插入的iframe窗口，这使得上传结束后，服务器将结果返回iframe窗口，所以当前页面就不会跳转了。其次，它在action属性指定的上传网址的后面，添加了一个参数，使得服务器知道回调函数的名称。这样就能将服务器返回的信息，从iframe窗口传到上层页面。

服务器（upload.php）返回的信息，应该是如下形式：

　　<script type="text/javascript">

　　　　window.top.window['callback'](data);

　　</script>

然后，在当前网页定义回调函数：

　　window[callback] = function(data){

　　　　console.log('received callback:', data);

　　　　iframe.remove(); //removing iframe

　　　　form.removeAttr('target');

　　　　form.attr('action', url);

　　　　window[callback] = undefined; //removing callback

　　};

**三、ajax上传**

HTML5提出了XMLHttpRequest对象的[第二版](http://www.w3.org/TR/XMLHttpRequest/" \t "_blank)，从此ajax能够上传文件了。这是真正的"异步上传"，是将来的主流。上一节的iframe上传，可以用作老式浏览器的替代方案。

ajax上传代码，放在表单的submit事件回调函数中：

　　form.on('submit',function() {

　　　　// 此处进行ajax上传

　　});

我们主要用的是[FormData对象](https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest/FormData" \t "_blank)，它能够构建类似表单的键值对。

　　// 检查是否支持FormData  
　　if(window.FormData) {

　　　　var formData = new FormData();

　　　　// 建立一个upload表单项，值为上传的文件  
　　　　formData.append('upload', document.getElementById('upload').files[0]);

　　　　var xhr = new XMLHttpRequest();

　　　　xhr.open('POST', $(this).attr('action'));

　　　　// 定义上传完成后的回调函数  
　　　　xhr.onload = function () {

　　　　　　if (xhr.status === 200) {

　　　　　　　　console.log('上传成功');

　　　　　　} else {

　　　　　　　　console.log('出错了');

　　　　　　}

　　　　};

　　　　xhr.send(formData);

　　}

**四、进度条**

XMLHttpRequest第二版还定义了一个[progress事件](https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest/Using_XMLHttpRequest" \l "Monitoring_progress" \t "_blank)，可以用来制作进度条。

首先，在页面中放置一个HTML元素progress。

　　<progress id="uploadprogress" min="0" max="100" value="0">0</progress>

然后，定义progress事件的回调函数。

　　xhr.upload.onprogress = function (event) {

　　　　if (event.lengthComputable) {

　　　　　　var complete = (event.loaded / event.total \* 100 | 0);

　　　　　　var progress = document.getElementById('uploadprogress');

　　　　　　progress.value = progress.innerHTML = complete;

　　　　}

　　};

注意，progress事件不是定义在xhr，而是定义在xhr.upload，因为这里需要区分下载和上传，下载也有一个progress事件。

**五、图片预览**

如果上传的是图片文件，利用[File API](http://dev.w3.org/2006/webapi/FileAPI/" \t "_blank)，我们可以做一个图片文件的预览。这里主要用到[FileReader对象](http://www.w3.org/TR/FileAPI/" \l "FileReader-interface" \t "_blank)。

　　// 检查是否支持FileReader对象  
　　if (typeof FileReader != 'undefined') {

　　　　var acceptedTypes = {  
　　　　　　'image/png': true,  
　　　　　　'image/jpeg': true,  
　　　　　　'image/gif': true  
　　　　};

　　　　if (acceptedTypes[document.getElementById('upload').files[0].type] === true) {

　　　　　　var reader = new FileReader();

　　　　　　reader.onload = function (event) {

　　　　　　　　var image = new Image();

　　　　　　　　image.src = event.target.result;

　　　　　　　　image.width = 100;

　　　　　　　　document.body.appendChild(image);

　　　　　　};

　　　　reader.readAsDataURL(document.getElementById('upload').files[0]);

　　　　}

　　}

**六、拖放上传**

最后，利用HTML5的[拖放功能](http://www.w3.org/TR/html5/dnd.html" \t "_blank)，实现拖放上传。

先在页面中放置一个容器，用来接收拖放的文件。

　　<div id="holder"></div>

对它设置样式：

　　#holder {  
　　　　border: 10px dashed #ccc;  
　　　　width: 300px;  
　　　　min-height: 300px;  
　　　　margin: 20px auto;  
　　}

　　#holder.hover {  
　　　　border: 10px dashed #0c0;  
　　}

拖放文件的代码，主要是定义dragover、dragend和drop这三个事件。

　　// 检查浏览器是否支持拖放上传。  
　　if('draggable' in document.createElement('span')){

　　　　var holder = document.getElementById('holder');

　　　　holder.ondragover = function () { this.className = 'hover'; return false; };

　　　　holder.ondragend = function () { this.className = ''; return false; };

　　　　holder.ondrop = function (event) {

　　　　　　event.preventDefault();

　　　　　　this.className = '';

　　　　　　var files = event.dataTransfer.files;

　　　　　　// do something with files

　　　　};

　　}

完成后的效果和总体代码，请看拖放上传[demo](http://html5demos.com/dnd-upload)。