### 一、XMLHttpRequest 2.0的家臣们

我大学那会儿，一个称为Ajax的东西对前端行业造成了深远影响，不仅是JS语言，而包括前端地位、职位兴起以及工作分工等。抛开IE6浏览器不谈，其他浏览器的Ajax实际上都是借助XMLHttpRequest实现的。

然后，好多年过去了，XMLHttpRequest带着两位家臣，DOMString和Document数据类型攻城略地，几乎一统天下。

然时代是发展的，人们群众的需求是旺盛的，HTML5犹如冉冉升起的新星开始普照大地，恩泽大众。XMLHttpRequest由于就两个家臣DOMString和Document，且并不是100%听话。因此，其已经开始hold不住HTML5的耀眼光芒了。为了顺应时代的潮流，XMLHttpRequest凹凸曼变身升级到2.0，变化诸多，其中一个很重要的变化就是广招家臣，扩张实力，与HTML5一起完成千秋万载之大业。

这些家臣有：DOMString、Document、FormData、Blob、File、ArrayBuffer这些类型。也就是在XMLHttpRequest Level 2背景下，我们Ajax可以发送任意这些类型的数据。有了诸多忠实可靠的家臣，XMLHttpRequest Level 2犹如织田信长般势不可挡，前途无量！

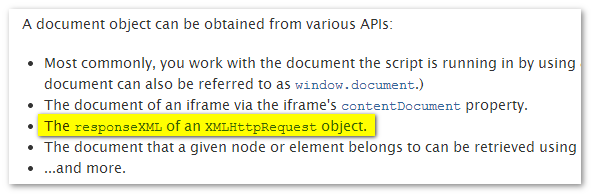
织田信长家臣有：羽柴秀吉、柴田胜家、明智光秀、竹中半兵卫、黑田官兵卫、织田信忠、泷川一益、丹羽长秀、前田利家、池田恒兴、佐久间信盛、森兰丸、九鬼嘉隆

### 二、家臣之DOMString

跟着XMLHttpRequest闯南走北很多年，看名字似乎很嚣张且高深莫测。实际上，在JavaScript中，DOMString就是String。规范解释说DOMString指的是UTF-16字符串，而JavaScript正是使用了这种编码的字符串，因此，在Ajax中，DOMString就等同于JS中的普通字符串。

大家应该都与XMLHttpRequest中数据返回属性之responseText打过交道吧，按照我的理解，这厮就是与DOMString数据类型发生关系的，表明返回的数据是常规字符串。

### 三、家臣之Document数据类型

如果单纯看Document对象，则解释很多，在这里，我们只要关注下图标注的这一个：  


可以看到，实际上就是XMLHttpRequest中数据返回属性之responseXML，也就是可以解析为XML的数据。因此，这里的Document数据类似你就可以近似看成XML数据类型。

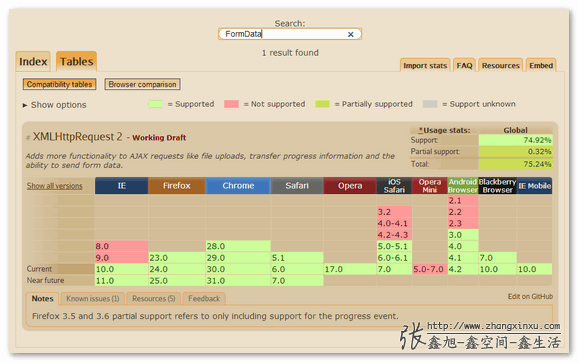
DOMString和Document都是XMLHttpRequest时代就跟随的数据类型，元老级。下面这些数据类型都是XMLHttpRequest 2.0新增的，新招的家臣，各怀绝技哦！

### 四、家臣之FormData对象

XMLHttpRequest Level 2添加了一个新的接口FormData. 利用FormData对象，我们可以通过JavaScript用一些键值对来模拟一系列表单控件，我们还可以使用XMLHttpRequest的send()方法来异步的提交这个”表单”。比起普通的ajax, 使用FormData的最大优点就是我们可以异步上传一个二进制文件。

以上为官方口吻的解释，略抽象。我们应该都用过jQuery，其中有个方法叫做serialize(), 作用就是表单序列化，也就是以查询字符串形式获得类表单post/get的数据给Ajax请求，例如：userid=123&username=zxx.

FormData对象的作用就类似于这里的serialize()方法，不过FormData是浏览器原生的，且支持二进制文件，是个一眼就会让人喜欢的很赞的东西！

**兼容性**如下：  


IE10+浏览器已经良好支持了，下面要介绍的其他家臣也都是IE10+支持。

实际使用是作为构造函数，如下：

new FormData ([可选]HTMLFormElement)

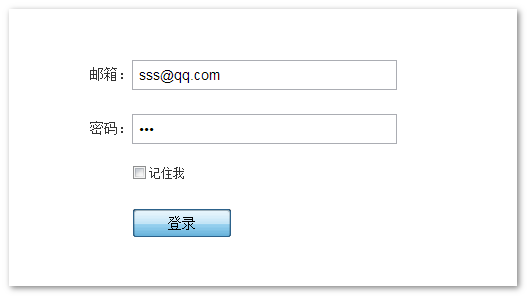
HTMLFormElement这个参数可选，可有可无。表示form表单元素，就是我们要序列化，要提交的那个表单元素。

例如：

var newFormData = new FormData(someFormElement);

newFormData就是someFormElement这个表单元素中所有键值对数据了。

您可以狠狠地点击这里：[FormData对象与表单数据获取demo](http://www.zhangxinxu.com/study/201310/formdata-form-data-get.html" \t "http://www.zhangxinxu.com/wordpress/2013/10/understand-domstring-document-formdata-blob-file-arraybuffer/_blank)

demo页面为一个普通的登录表单，截图如下：  


点击登录执行Ajax登录，不过这里是采用FormData格式发送的。

相关JS代码如下：

document.querySelector("#formData").addEventListener("submit", function(event) {

var myFormData = new FormData(this);

var xhr = new XMLHttpRequest();

xhr.open(this.method, this.action);

xhr.onload = function(e) {

if (xhr.status == 200 && xhr.responseText) {

// 显示：'欢迎你，' + xhr.responseText;

this.reset();

}

}.bind(this);

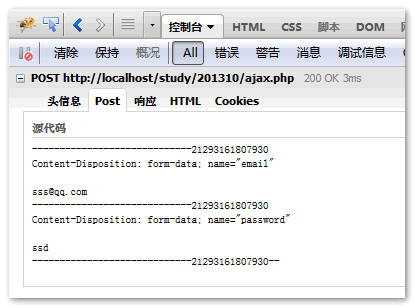
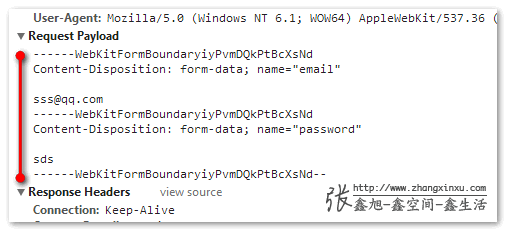
// 发送FormData对象数据

xhr.send(myFormData);

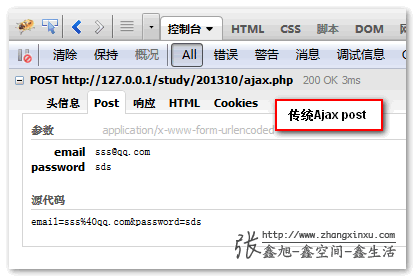
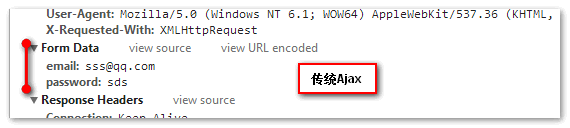
// 阻止默认的表单提交

event.preventDefault();

}, false);

我们打开工具查看下请求：  
  


以上分别是Firebug和Chrome开发者工具查看的结果。

我们再看下传统Ajax请求：  
  


差异还是比较大的。  
FormData提交格式的每个数据分三部分：

* 第一部分也就是第一行，表示“分界线(boundary)”，我尚未深入研究这个分界线，不过，我没估计错的话，二进制大文件分隔传输时候，就是使用这个分界线。在webkit核心中，使用“——WebKitFormBoundary”加16位随机Base64位编码的字符串作为分隔边界。根据Firebug的显示，Firefox中，似乎是使用很多个"-"加时间戳进行边界分隔的。这里的边界的作用比较单纯，可能就是把表单的这两个字段作为两个独立数据流传输。
* 第二部分也就是第二行，表示内容配置，这里都是统一的form-data（因为是FormData对象格式提交的），然后紧跟着name键值。
* 第三部分就是第三行，表示传输的值。

虽然前台传输差异较大，但是，后台的处理是可以一致的，例如，我这里的PHP代码就非常简单：

<?php

$username = $\_POST['email'];

if (isset($username) == true) {

echo $username;

} else {

echo '';

}?>

FormData对象还有一个方法，为append()方法，可以人为的给当前FormData对象添加一个键/值对。

语法如下：

void append(DOMString 键, Blob 值, [可选] DOMString 文件名);

void append(DOMString 键, DOMString 值);

语法第一行出现了Blob, 这是我们下面要介绍的家臣之一，您可以先记住，这是用来表示二进制文件的，后面的文件名可选，据说，如果缺省，且传输的是Blob对象，则会使用"blob"代替。  
第二行就是比较常规的用法，DOMString这个家臣已经介绍了，在JavaScript中就是普通字符串的意思。因此，比方说我们要额外提交个token值，可能就是：

myFormData.append("token", "ce509193050ab9c2b0c518c9cb7d9556");

于是，后台就可以get token这个值了。

大家自行补脑，我就不再撑篇幅了。

### 五、家臣之Blob数据对象

一个Blob对象就是一个包含有只读原始数据的类文件对象。Blob对象中的数据并不一定得是JavaScript中的原生形式。File接口基于Blob, 继承了Blob的功能，并且扩展支持了用户计算机上的本地文件。

创建Blob对象的方法有几种，可以调用Blob构造函数，还可以使用一个已有Blob对象上的slice()方法切出另一个Blob对象，还可以调用canvas对象上的toBlob方法。

以上为MDN上官方口吻的解释。实际上，Blob是计算机界通用术语之一，全称写作：BLOB (binary large object)，表示二进制大对象。MySql/Oracle数据库中，就有一种Blob类型，专门存放二进制数据。

在实际Web应用中，Blob更多是图片二进制形式的上传与下载，虽然其可以实现几乎任意文件的二进制传输。

举个例子，使用Blob从服务器上GET某mm的图片（只要关心标红的部分）：

var xhr = new XMLHttpRequest();

xhr.open("get", "mm1.jpg", true);

xhr.responseType = "blob";

xhr.onload = function() {

if (this.status == 200) {

var blob = this.response; // this.response也就是请求的返回就是Blob对象

var img = document.createElement("img");

img.onload = function(e) {

window.URL.revokeObjectURL(img.src); // 清除释放

};

img.src = window.URL.createObjectURL(blob);

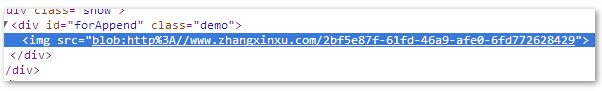
eleAppend.appendChild(img);

}

}

xhr.send();

您可以狠狠地点击这里：[Blob获取图片并二进制显示demo](http://www.zhangxinxu.com/study/201310/blob-get-image-show.html" \t "http://www.zhangxinxu.com/wordpress/2013/10/understand-domstring-document-formdata-blob-file-arraybuffer/_blank)

我们查看demo页面这个mm图片元素，会发现其URL地址既不是传统HTTP，也不是Base64 URL，而是Blob形式~如下截图示意：  
  


这就是Blob在Web开发中非常重要的一个功能——创建Blob网址。上述代码涉及XMLHttpRequest 2一些重要知识点，以及window.URL相关技术，都是可以深入挖掘学习的部分，但，不是本文重点，以后有机会会细致阐述。

但是，并不是所有的图片都能以Blob形式请求，因为，毕竟是Ajax请求嘛，还是有一定的跨域限制。XMLHttpRequest 2虽然支持跨源资源共享(CORS)，但是，还是需要对Access-Control-Allow-Origin的设置，允许来自那个域名的这类请求，例如，允许本人的站点Blob请求你服务器上的图片资源，你可以设置：

Access-Control-Allow-Origin: http://zhangxinxu.com

要允许任何域向您提交请求，可以设置：

Access-Control-Allow-Origin: \*

我们都知道CSS3的font-face属性，在Firefox浏览器下，如果字体文件跨域（包括跨子域），是显示不出来的，也是通过

Access-Control-Allow-Origin: \*

设置解决。其实，本质是一样的。

由于权限原因，我的个人站点无法配置Access-Control-Allow-Origin，我测试了下，新浪微博的图片是无法二进制请求的，不过我的前东家，[xiaomishu.com](http://www.xiaomishu.com/)的图片都是可以Ajax请求并Blob显示的，悄悄告诉大家，是我当初动的手脚，(\*^\_\_^\*) 嘻嘻……

**属性**  
Blob对象有两个属性，参见下表：

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **描述** |
| **size** | unsigned long long(表示可以很大的数值) | Blob对象中所包含数据的大小。字节为单位。 **只读。** |
| **type** | DOMString | 一个字符串，表明该Blob对象所包含数据的MIME类型。例如，上demo图片MIME类似就是”image/jpeg“. 如果类型未知，则该值为空字符串。 **只读。** |

今天在微博上看到一个表单提交之前判断文件大小并作阻止的tip，实际上，就是使用的Blob对象的size属性。

**构造函数**  
与FormData对象类似，Blob也有一个构造函数用法。语法如下：

Blob Blob(

[可选] Array parts,

[可选] BlobPropertyBag properties

);

例如：

var myBlob= new Blob(arrayBuffer);

其中，两个参数的含义是：

**parts**

一个数组，包含了将要添加到Blob对象中的数据。数组元素可以是任意多个的ArrayBuffer, ArrayBufferView(typed array), Blob, 或者DOMString对象。

**properties**

一个对象，设置Blob对象的一些属性。目前仅支持一个type属性，表示Blob的类型。

**方法**  
Blob对象有个很重要的方法-slice()，作用是，可以实现文件的分割！

这个slice()有一段不堪回首的历史，不过现在大家不要关心。目前的slice()方法已经跟JS中数组啊，字符串的slice方法用法一致了。如下：

Blob slice(

[可选] long long start,

[可选] long long end,

[可选] DOMString contentType

};

参数释义：

**start**

开始索引，可以为负数，语法类似于数组的slice方法。默认值为0.

**end**

结束索引，可以为负数，语法类似于数组的slice方法。默认值为最后一个索引。

**contentType**

新的Blob对象的MIME类型，这个值将会成为新的Blob对象的type属性的值，默认为一个空字符串。

显然，此方法返回的数据格式还是Blob对象，不过是指定范围复制的新的Blob对象。注意，如果start参数的值比源Blob对象的size属性值还大，则返回的Blob对象的size值为0，也就是不包含任何数据。

### 六、家臣之File对象

File顾名思意就是“文件”，通常而言，表示我们使用file控件(<input type="file">)选择的[FileList](https://developer.mozilla.org/en-US/docs/Web/API/FileList)对象，或者是使用拖拽操作搞出的[DataTransfer](https://developer.mozilla.org/en-US/docs/DragDrop/DataTransfer)对象。

这里的File对象也是二进制对象，因此，从属于Blob对象，Blob对象的一些属性与方法，File对象同样适合，且推荐使用Blob对象的属性与方法。

File对象自身也有一些属性与方法，但是，有些已经过时——不推荐使用，因此，当前很多HTML5 Ajax文件上传下载的教程中出现是属性和方法都是过时的，不要盲目Copy，请大家明辨！

**属性**

**File.lastModifiedDate**[只读]

文件对象最后修改的日期

**File.name**[只读]

文件对象的名称

**File.fileName**[只读] [过时不推荐使用]

文件对象的名称（请使用File.name代替）

**File.fileSize**[只读] [过时不推荐使用]

文件对象的大小（请使用Blob.size代替）

**Blob.size**[只读]

Blob对象包含数据的字节大小

**Blob.type**[只读]

一个字符串，表明该Blob对象所包含数据的MIME类型

**方法**

**File.getAsBinary()**[过时不推荐使用]

二进制形式返回文件数据（请使用FileReader对象的FileReader.readAsBinaryString()方法代替）

**File.getAsDataURL()**[过时不推荐使用]

返回文件data:URL编码字符串数据（请使用FileReader对象的FileReader.readAsDataURL()方法代替）

**File.getAsText(string encoding)**[过时不推荐使用]

以给定的字符串编码返回文件数据解释后的文本（请使用FileReader对象的FileReader.readAsText()方法代替）

**Blob.size**[只读]

Blob对象包含数据的字节大小

**Blob.type**[只读]

一个字符串，表明该Blob对象所包含数据的MIME类型。

上面有提到FileReader对象，这货是相当的有货，之前有人曾问我，如何将图片转换成Data base64 url格式，其中一个方法就是FileReader.readAsDataURL()方法（还有就是canvas元素的toDataURL()和toDataURLHD()方法），然与本文主旨无关，暂不赘述；如您有兴趣，页面底部有其相关知识点链接，可自行概览。

### 七、家臣之ArrayBuffer对象

//zxx:ArrayBuffer对象牵扯知识点非常多，这里仅接触肌肤，深入接触下次会专门再说下。

很术语的解释有：

ArrayBuffer表示二进制数据的原始缓冲区，该缓冲区用于存储各种类型化数组的数据。

ArrayBuffer是二进制数据通用的固定长度容器。

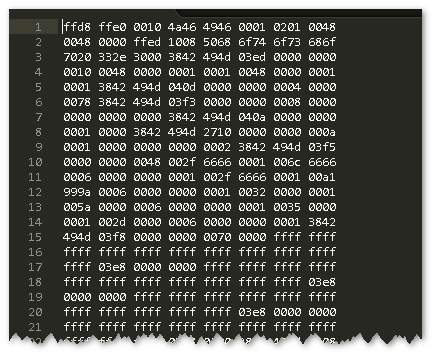
所谓术语，就是小白看不懂的解释语。我再用通俗语解释下，希望大家可以有点感性的认识：

术语中，提到“二进制”，我们脑中应该会出现01010111之类；提到“缓冲”，会联想到在线视频提前加载一部分视频的那个缓冲。但是，两个合起来，“二进制数据缓冲区”，脑补就不连贯了，焦虑产生~~

现在，听我的，上面概念全部扔掉。所谓ArrayBuffer就是个装着2进制数据的对象。或者想象成带了个名叫“缓冲”帽子的二进制数据。然后直接关联：ArrayBuffer ＝ 2进制。

上面＝表示关联，不是相等，诸位。

例如，我们设置Ajax请求的responseType为”arraybuffer“，我们去请求某mm图片，返回的response就是ArrayBuffer，就是个二进制对象。什么缓冲不缓冲的，千万别补脑这个。

如果还觉得概念抽象，可以看下面的具体认知：  
大家可能玩过神器编辑器Sublime Text, 我们随便找张图片拖进去，会发现是类似下面这样子的代码：  


Sublime Text以16进制的形式显示图片资源，ArrayBuffer的差别在于是二进制，因此，我们可以把ArrayBuffer的形体脑补成——上图的数字全是的0101 1000 1101之类的。Get it否？

上面提到的Blob对象也是二进制，那Blob和ArrayBuffer有啥区别呢？

Blob可以append ArrayBuffer数据，也就是Blob是个更高一级的大分类，类似领导的感觉。ArrayBuffer则是具有某种恶魔果实的尖兵。

ArrayBuffer存在的意义就是作为数据源提前写入在内存中，就是提前钉死在某个区域，长度也固定，万年不变。于是，当我们要处理这个ArrayBuffer中的二进制数据，例如，分别8位，16位，32位转换一遍，这个数据都不会变化，3种转换共享数据。

So，ArrayBuffer就是缓冲出来的打死不动的二进制对象。

注意，ArrayBuffer本身是不能读写的，需要借助类型化数组或DataView对象来解释原始缓冲区（宰割原始二进制数据）。

**类型化数组**  
类型化数组(Typed Arrays)是JavaScript中新出现的一个概念，专为访问原始的二进制数据而生。

类型数组的类型有：

|  |  |  |
| --- | --- | --- |
| **名称** | **大小 (以字节为单位)** | **说明** |
| **Int8Array** | 1 | 8位有符号整数 |
| **Uint8Array** | 1 | 8位无符号整数 |
| **Int16Array** | 2 | 16位有符号整数 |
| **Uint16Array** | 2 | 16位无符号整数 |
| **Int32Array** | 4 | 32位有符号整数 |
| **Uint32Array** | 4 | 32位无符号整数 |
| **Float32Array** | 4 | 32位浮点数 |
| **Float64Array** | 8 | 64位浮点数 |

本质上，类型化数组和ArrayBuffer是一样的。不过一个可读写（脱掉buffer限制），一个当数据源的命。

举一些代码例子，看看本质一致在何处：

// 创建一个8字节的ArrayBuffer

var b = new ArrayBuffer(8);

// 创建一个指向b的视图v1，采用Int32类型，开始于默认的字节索引0，直到缓冲区的末尾

var v1 = new Int32Array(b);

// 创建一个指向b的视图v2，采用Uint8类型，开始于字节索引2，直到缓冲区的末尾

var v2 = new Uint8Array(b, 2);

// 创建一个指向b的视图v3，采用Int16类型，开始于字节索引2，长度为2

var v3 = new Int16Array(b, 2, 2);

上面代码里变量的数据结构如下表所示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **变量** | **索引** | | | | | | | |
|  | 字节（不可索引） | | | | | | | |
| b= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 类型数组 | | | | | | | |
| v1= | 0 | | | | 1 | | | |
| v2= |  |  | 0 | 1 | 2 | 3 | 4 | 5 |
| v3= |  |  | 0 | | 1 | |  |  |

由于类型化数组直接访问固定内存，因此，速度很赞，比传统数组要快！因为普通Javascript数组使用的是Hash查找方式。同时，类型化数组天生处理二进制数据，这对于XMLHttpRequest 2、canvas、webGL等技术有着先天的优势。

**DataView对象**  
DataView对象在可以在ArrayBuffer中的任何位置读取和写入不同类型的二进制数据。

用法语法如下：

var dataView = new DataView(DataView(buffer, byteOffset[可选], byteLength[可选]);

其中，buffer表示ArrayBuffer；byteOffset指缓冲区开始处的偏移量（以字节为单位）；byteLength指缓冲区部分的长度（以字节为单位）。

**属性**

**buffer**

表示ArrayBuffer

**byteOffset**

指缓冲区开始处的偏移量

**byteLength**

指缓冲区部分的长度

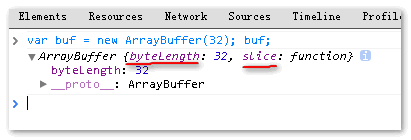
方法有很多，实际上，是有规律的，篇幅原因，也不是重点，就单纯露个脸：  
getInt8, getUint8, getInt16, getUint16, getInt32, getUint32, getFloat32, getFloat64, setInt8, setUint8, setInt16, setUint16, setInt32, setUint32, setFloat32, setFloat64.

下面回到**ArrayBuffer对象**，ArrayBuffer对象自身也可以构造，跟上面的FormData, Blob对象类似，例如：

var buf = new ArrayBuffer(32);

语法为：

ArrayBuffer ArrayBuffer(length[可以很大数值]);

我们在控制台运行下new ArrayBuffer(32)，看看结果：  


可以看到，其有一个byteLength属性，表示ArrayBuffer的长度，也可以说是大小；还有一个slice方法，语法如下：

ArrayBuffer slice(

begin

end[可选]

);

begin表示起始，end表示结束点。据说，Internet Explorer 10 以及iOS6-是没有该方法的。

综上，举个ArrayBuffer的实例吧，发送使用XMLhttpRequest发送ArrayBuffer数据：

function sendArrayBuffer() {

var xhr = new XMLHttpRequest();

xhr.open('POST', '/server', true);

xhr.onload = function(e) { ... };

var uInt8Array = new Uint8Array([1, 2, 3]);

xhr.send(uInt8Array.buffer);

}

使用了类型化数组，发送的是类型化数组(uInt8Array)的buffer属性，也就是ArrayBuffer对象。

over~

### 八、结束语

新技术层出不穷，我觉得吧，以后，行业的分支可能要更细了。比方说JS开发吧，可能就有JS UI交互开发工程师；JS Web开发工程师。因为，一个人想要完全hold住这么多的知识点，还真不是一般人能做到的。

刚开始写的时候，还想最后举个文件分割上传的例子，只可惜内容实在太多，加上去也会被湮没，于是作罢，决定有机会，专门讲下这个。还有FileReader可以独立讲一下，还有类型化数组也可以专门讲一下等。

学路漫漫，任重道远。文中若有致命的结论错误或疏忽的文字书写错误，都欢迎指正，不甚感谢。欢迎讨论，欢迎交流！