

# Web前端开发技术与实践

## 第17章：文件

阮晓龙

13938213680 / rxl@hactcm.edu.cn  
<http://web.book.51xueweb.cn>

河南中医学院管理信息工程学科  
河南中医学院网络信息中心

2015.9

# 本章主要内容

- 文件存储
- 处理用户文件
- 文件操作
- 文件内容操作
- 文件API
- 案例：用户本地资源管理



# 1.文件存储

---

- 文件是用户可以方便地与他人分享的信息单位。用户不能分享变量的值，但肯定能创建文件的副本，并用DVD、移动存储器或硬盘，或者Internet等机制发送文件。文件可以存储大量数据，可以移动、复制、传输，与内容的性质无关。
- 对于每个应用程序来说，文件总是不可缺少的一部分，但迄今为止，在Web上还没有处理文件的机制。仅有的文件选项就是下载或上传服务器或用户计算机上先前已有的文件。在HTML5出现之前，在Web上没有文件创建、复制，也没有文件处理。

# 1.文件存储

---

- HTML5规范从一开始就考虑到了Web应用程序构建和操作性的每个方面。从设计到基本的数据结构，每件事都考虑到了，文件也不可能遗漏在外。因此，HTML5规范将文件API整合进来。
- 文件API拥有底层基础设施，可以同步工作，也可以异步工作。之所以开发同步部分，是为了在Web Workers API上工作，这一点与其他API类似，而异步部分针对的是普通Web应用程序。这些特征意味着必须注意处理过程中的每个方面，检测处理成功还是失败，日后在此之上可能会采用更简单的API。

# 1.文件存储

- 目前浏览器的支持情况如表17-01所示。

表 17-01 文件 API 的浏览器支持情况

API	Chrome	Firefox	Opera	IE
File API	13+	×	×	×
FileReader API	6+	4+	12+	10+
Filesystem&FileWriter API	13+	×	×	×
BlobBuilder API	17+	6+	×	10+

## 2.处理用户文件

### 2.1读取文件

- 要从用户的计算机上读取用户的文件，必须使用FileReader接口。FileReader拥有4个方法，其中3个用以读取文件，另一个用来中断读取，如表17-02中列出了这些方法以及参数。需要注意的是，无论读取成功或失败，方法并不会返回读取结果，这一结果存储在**result**属性中。

表 17-02 FileReader 对象方法

方法名	参数
abort	null
readAsArrayBuffer	blob
readAsDataURL	blob
readAsText	blob, [encoding]



现场演示:

- 案例17-01: 读取文件

## 2.处理用户文件

### 2.2读取文件属性

- 在实际应用程序中，文件名、文件大小及文件类型等信息都是必需的，这些信息可以让用户了解所处理文件的情况，甚至可以控制用户的输入。<input>标签发送的文件对象提供了可以用来获得文件信息的多个属性，具体属性如下所示。
  - name：该属性返回文件的全名（文件名和扩展名）。
  - size：该属性返回文件的大小，以字节为单位。
  - type：该属性返回文件的类型，以MIME类型表示。





现场演示:

- 案例17-02: 读取文件属性

## 2.处理用户文件

### 2.3文件分割

- 除了文件外，API还能处理另一个源类型，即blob。blob是代表原始数据的对象。创建blob对象的目的是为了克服JavaScript在处理二进制数据上的限制。blob通常是由文件生成的，但并不是必需的，不将整个文件加载到内存就能处理数据是个很好的做法，这种做法为一小片一小片地处理二进制信息提供了可能性。



现场演示:

- 案例17-03: 处理blob

## 2.处理用户文件

### 2.4处理事件

- 将文件加载进内存需要的时间长短取决于文件的大小。对小文件来说，加载过程仿佛一蹴而就；但大文件可能需要几分钟才能加载完成。除了前边已经提过的load事件，API还提供了几个特殊事件，用来告知处理过程的每个情况。表17-03中归纳了FileReader的事件模型。

表 17-03 FileReader 对象事件

事件	描述
onloadstart	读取开始时触发
onprogress	在读取文件或 blob 的时候，周期性地触发这个事件
onabort	当处理中断时触发
onerror	当读取出错时触发
onload	文件读取成功完成时触发
onloadend	读取完成触发，无论成功或失败



现场演示:

- 案例17-04: 用事件来控制流程

## 3.文件操作

### 3.1本地磁盘操作

- 应用程序保留的空间就像一个沙盒，是一块有根目录和配置的小硬盘，要使用这个硬盘，必须请求为应用程序初始化一个FileSystem。
- “文件API：目录和系统”包含了两个不同的版本，分别为异步API和同步API。
  - 异步API：对于一般的应用来说非常有用，可以防止阻塞。
  - 同步API：特别为Web Workers设计。

## 3.文件操作

### 3.1本地磁盘操作

- 考虑到安全性，API接口设计时做了一些限制，具体如下所述。
  - 存储配额限制(quota limitations)。
  - 同源限制，如只能读写同域内的cookie和localStorage。
  - 文件类型限制，限制可执行文件的创建或者重命名为可执行文件。

## 3.文件操作

### 3.1本地磁盘操作

- 首先需要通过请求一个LocalFileSystem对象来得到HTML 5文件系统的访问，使用window.requestFileSystem全局方法的具体代码如下所示。
  - `window.requestFileSystem(type, size, successCallback, opt_errorCallback)`



## 3.文件操作

### 3.1本地磁盘操作

- 在使用window.requestFileSystem时需要注意以下几个方面的事项。
  - Google Chrome和Opera是目前仅有的实现了这部分API的浏览器。
  - 因为该实现还属于实验性质，所以必须用特定方法webkitRequestFileSystem()替代requestFileSystem()方法。换用这个方法后，才能在浏览器里测试上面的代码及后面的示例，且参数type只能选用TEMPORARY，否则会显示QUOTA\_EXCEEDED\_ERROR错误。
  - 需要将HTML文件发布至一个Web站点下，通过浏览器访问以查看效果。

## 3.文件操作

### 3.1本地磁盘操作

- 要在本地通过“file://...”方式运行以下示例，则建议使用Chrome浏览器，且必须使用以下标签打开Chrome:--allow-file-access-from-files。要在Windows上给Chrome加上这个标签，请在桌面的Chrome图标上单击鼠标右键，选择【属性】选项。在打开的窗口中，可以看到【目标】域，里面是Chrome执行文件的路径和文件名，并添加相应标签，完成之后为：  
C\Users\...\Chrome\Application\chrome.exe --allow-file-access-from-files，此方法建议用于开发测试的环境。

## 3.文件操作

### 3.1本地磁盘操作

- 表17-04列出了requestFileSystem的方法。

表 17-04 requestFileSystem 方法

方法	参数
requestFileSystem	type, size, success function, error function



现场演示：

- 案例17-05：设置自己的文件系统

## 3.文件操作

### 3.2创建文件

- getFile()方法是API的DirectoryEntry接口的一部分。这个接口共提供了4种方法用来创建和处理文件及目录，具体如表17-05所示。

表 17-05 DirectoryEntry 对象方法

方法	参数
getFile	path,options,success function,error function
getDirectory	path,options,success function,error function
createReader	null
removeRecursively	null

## 3.文件操作

### 3.3创建目录

- getFile()方法（针对文件）和getDirectory()方法（针对目录）的用法完全相同。只需要将getFile()换成getDirectory()即可，具体代码如下所示。

```
function create(){
    var name=document.getElementById("myentry").value;
    if(name!=""){
        hd.getDirectory(name, {create:true, exclusive:false}, show, showerror);
    }
}
```

## 3.文件操作

### 3.4列出文件

- 如前所述，`createReader()`方法可以得到指定路径中的项（文件和目录）列表。这个方法返回的`DirectoryReader`对象的`readEntries()`方法可以读取指定目录中的项。`DirectoryReader`对象的方法如表17-06所示。

表 17-06 DirectoryReader 对象方法

方法	参数
<code>readEntries</code>	<code>success function, error function</code>



现场演示:

- 案例17-06: 列出文件



## 3.文件操作

### 3.6处理文件

- 如前所述，`createReader()`方法可以得到指定路径中的项（文件和目录）列表。这个方法返回的`DirectoryReader`对象的`readEntries()`方法可以读取指定目录中的项。`DirectoryReader`对象的方法如表17-06所示。

表 17-06 DirectoryReader 对象方法

方法	参数
<code>readEntries</code>	<code>success function, error function</code>

## 3.文件操作

### 3.6处理文件

- 对于执行常规的文件和目录操作来说，还有几个有用的方法。使用这些方法可以移动、复制或删除项，就像桌面应用程序一样，具体的对象方法如表17-08所示。

表 17-08 Entry 对象方法

方法	参数
moveTo	parent, new name, success function, error function
copyTo	parent, new name, success function, error function
remove	null



现场演示:

- 案例17-07: 移动文件

## 3.文件操作

3.7复制

- moveTo()方法和copyTo()方法唯一的区别就是后者保留原始文件。要使用copyTo()方法，只需要修改示例17-07代码中方法的名称。modify()函数修改完后具体代码如下所示。

```
function modify(){
    var origin=document.getElementById('origin').value;
    var destination=document.getElementById('destination').value;
    hd.getFile(origin, null, function(file){
        hd.getDirectory(destination, null, function(dir){
            file.copyTo(dir, null, success, showerror);
        },showerror);
    }, showerror);
}
```

## 3.文件操作

### 3.7删除

- 删除文件或目录相对于移动或复制文件更为简单，需要完成的操作就是获得将要删除的文档或目录的Entry对象，然后在这个对象上应用remove()方法。具体代码如下所示。

```
function remove(){
    var origin=document.getElementById("origin").value;
    var origin=path+origin;
    hd.getFile(origin, null, function(entry){
        entry.remove(success,showerror)
    }, showerror);
}
```

## 3.文件操作

### 3.7删除

- 如果要删除的是目录而不是文件，则必须使用 `getDirectory()` 方法创建目录的 `Entry` 对象，然后 `remove()` 方法的用法不变。但对目录来说，有一种情况必须考虑：如果目录不为空，则 `remove()` 方法会返回错误。如果要删除目录及其内容，必须使用另一个方法 `removeRecursively()`。具体代码如下所示。

```
function removeDirectory(){
    var destination=document.getElementById('destination').value;
    hd.getDirectory(destination, null, function(entry){
        entry.removeRecursively(success,showerror)
    }, showerror);
}
```

## 4.文件内容操作

### 4.1写入内容

- 要向文件写入内容，必须创建FileWriter对象。该对象是由FileEntry接口的createWriter()方法返回的。本接口是Entry接口的扩展，提供了操作文件的两个方法，FileWriter对象方法如表17-09所示。

表 17-09 FileEntry 对象方法

方法	参数
createWriter	success function,error function
file	success function,error function



现场演示:

- 案例17-08: 写入内容



## 4.文件内容操作

### 4.2追加内容

- 因为没有指定在哪个位置插入内容，所以前面的代码从文件开始写入blob，要选择在现有文件特定位置或末尾追加内容，必须使用seek()方法。
- 以下代码函数改进了前面的writecontent()函数，加入seek()方法将写入位置移动到文件末尾。这样write()方法写入的内容就不会覆盖文件现有的内容。

```
function writecontent(fileWriter){
    var text=document.getElementById('mytext').value;
    fileWriter.seek(fileWriter.length);
    fileWriter.onwriteend=success;
    var blob=new Blob([text],[type:"text/plain; charset=UTF-8"]);
    fileWriter.write(blob);
}
```

## 4.文件内容操作

---

### 4.3读取文件内容

- 读取过程要使用FileReader()构造函数和readAsText()等读取方法读取并获得文件的内容。



现场演示:

- 案例17-09: 从文件系统读取文件

## 5.案例：用户本地资源管理

---

- 本案例通过表单的形式可将用户选择的文件保存至本地，也能够使用户自定义添加文件，并以列表的方式可视化的展示本地资源信息，同时在信息列表中即可实现对用户本地资源的查看、修改和删除等管理操作。



现场演示：

- 案例17-10：用户本地资源管理

Thanks.