你知道 HTML5 中的文件操作么？

HTML5 中提供的文件 API 在前端中有着丰富的应用，上传、下载、读取内容等在日常的交互中很常见。而且在各个浏览器的兼容也比较好，包括移动端，除了 IE 只支持 IE10 以上的版本。想要更好地掌握好操作文件的功能，先要熟悉每个 API。

FileList 对象和 file 对象

HTML 中的 input[type="file"] 标签有个 multiple 属性，允许用户选择多个文件，FileList 对象则就是表示用户选择的文件列表。这个列表中的每一个文件，就是一个 file 对象。

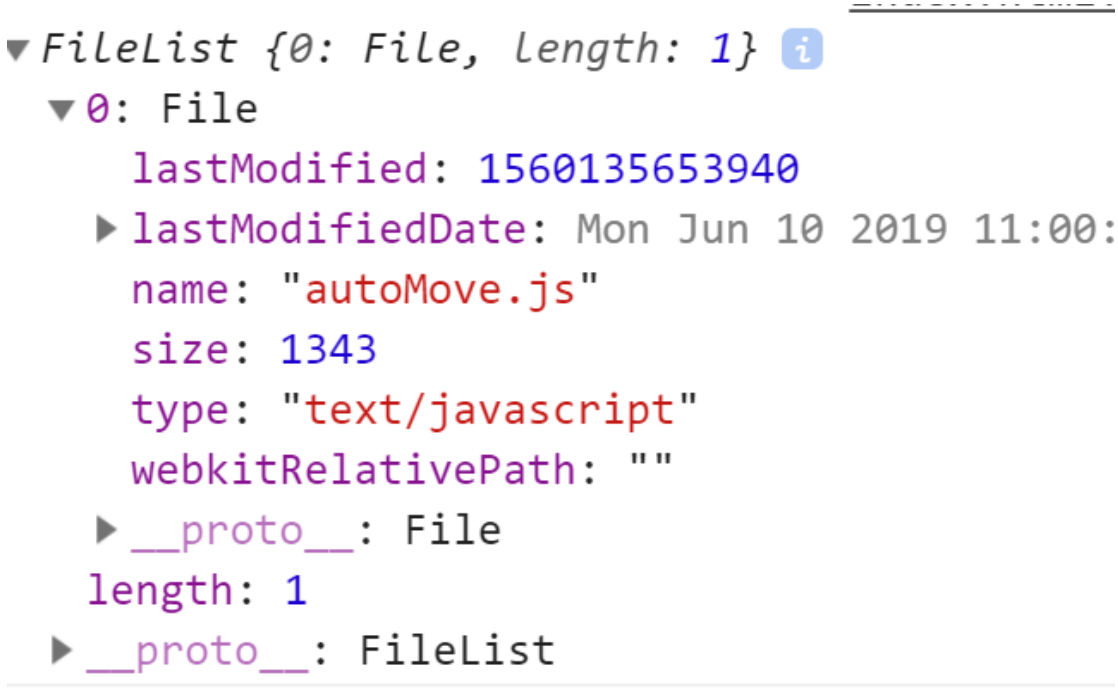
file 对象的属性：

- name ：文件名，不包含路径。
- type ：文件类型。图片类型的文件都会以 image/ 开头，可以由此来限制只允许上传图片。
- size ：文件大小。可以根据文件大小来进行其他操作。
- lastModified ：文件最后修改的时间。

代码：

```
<input type="file" id="files" multiple>
<script>
    var elem = document.getElementById('files');
    elem.onchange = function (event) {
      console.log(this.files)
    }
</script>
```

我们看一下 file 对象：



我们可以看到这个文件的相关信息，当我们传多个文件时，这个数组中的每一位都是一个文件对象。
input 中有个 accept 属性，可以用来规定能够通过文件上传进行提交的文件类型。

accept="image/*" 可以用来限制只允许上传图像格式。但是在 Webkit 浏览器下却出现了响应滞慢的问题，要等上好几秒才弹出文件选择框。

解决方法就是将 * 通配符改为指定的 MIME 类型。
Like this:

```
<input type="file" accept="image/gif,image/jpeg,image/jpg,image/png">
```

Blob 对象

Blob 对象相当于一个容器，可以用于存放二进制数据。它有两个属性，size 属性表示字节长度，type 属性表示 MIME 类型。

如何创建

Blob 对象可以使用 Blob() 构造函数来创建。

```
var blob = new Blob(['hello'], {type:"text/plain"});
```

Blob 构造函数中的第一个参数是一个数组，可以存放 ArrayBuffer 对象、ArrayBufferView 对象、Blob 对象和字符串。

Blob 对象可以通过 slice() 方法来返回一个新的 Blob 对象。

```
var newblob = blob.slice(0,5, {type:"text/plain"});
```

slice() 方法使用三个参数，均为可选。第一个参数代表要从 Blob 对象中的二进制数据的起始位置开始复制，第二个参数代表复制的结束位置，第三个参数为 Blob 对象的 MIME 类型。

canvas.toBlob() 也可以创建 Blob 对象。toBlob() 使用三个参数，第一个为回调函数，第二个为图片类型，默认为 image/png，第三个为图片质量，值在 0 到 1 之间。

```
var canvas = document.getElementById('canvas');
canvas.toBlob(function(blob){ console.log(blob); }, "image/jpeg", 0.5);
```

下载文件

Blod 对象可以通过 window.URL 对象生成一个网络地址，结合 a 标签的 download 属性来实现下载文件功能。

比如把 canvas 下载为一个图片文件。

```
var canvas = document.getElementById('canvas');
canvas.toBlob(function(blob){
    // 使用 createObjectURL 生成地址，格式为 blob:null/fd95b806-db11-4f98-b2ce-5eb16b38ba36
    var url = URL.createObjectURL(blob);
    var a = document.createElement('a');
    a.download = 'canvas';
    a.href = url;
    // 模拟 a 标签点击进行下载
    a.click();
    // 下载后告诉浏览器不再需要保持这个文件的引用了
    URL.revokeObjectURL(url);
});
```

也可以将字符串保存为一个文本文件，方法类似。

FileReader 对象

FileReader 对象主要用来把文件读入内存，并且读取文件中的数据。通过构造函数创建一个 FileReader 对象

```
var reader = new FileReader();
```

该对象有以下方法：

- abort：中断读取操作。
- readAsArrayBuffer：读取文件内容到 ArrayBuffer 对象中。
- readAsBinaryString：将文件读取为二进制数据。
- readAsDataURL：将文件读取为 data: URL 格式的字符串。
- readAsText：将文件读取为文本。

上传图片预览

在常见的应用就是在客户端上传图片之后通过 readAsDataURL() 来显示图片。

```
<input type="file" id="files" accept="image/jpeg,image/jpg,image/png">
<img src="blank.gif" id="preview">
<script>
    var elem = document.getElementById('files'),
        img = document.getElementById('preview');
    elem.onchange = function () {
        var files = elem.files,
            reader = new FileReader();
        if(files && files[0]){
            reader.onload = function (ev) {
                img.src = ev.target.result;
            }
            reader.readAsDataURL(files[0]);
        }
    }
</script>
```

通常涉及文件操作就是这些了。至于上传和下载都需要配合着服务器进行使用，在这里就不过多描述了。