HTML5新特性 -- 拖放

1.拖放

1.1 概述

拖放是HTML5的新特性，即将对象抓取的从一个位置拖动另外一个位置

在HTML5中任何元素可以拖放

为解决浏览器的兼容性，建议在被拖放的HTML文本类型的元素上添加draggable="true"，如：

```
<p draggable="true">段落</p>
```

1.2 DragEvent接口

DragEvent表示拖、放事件接口，该接口继承MouseEvent和Event接口

DragEvent接口的事件包括：

- dragstart
- drag
- dragend

  dragend

- dragenter
- dragover
- dragleave
- drop

1.3 源对象事件

• dragstart

dragstart事件在源对象开始被拖动时触发，其语法结构是：

```
HTMLElement.ondragstart = function(){
    ....
}
```

• drag

`drag`事件在源对象被拖动过程中触发，其语法结构是：

```
HTMLElement.ondrag = function(){
    ....
}
```

- `dragend`

事件在源对象被拖动结束后触发（也可能在目标区域内，也可能在目标区域外），其语法结构是

```
HTMLElement.ondragend = function(){
    ....
}
```

1.4 目标对象事件

- `dragenter`

`dragenter`事件在源对象进入目标对象时触发，其语法结构是：

```
HTMLElement.ondragenter = function(){
    ...
}
```

- `dragover`

`dragover`事件在源对象在目标对象上悬停时触发，其语法结构是：

```
HTMLElement.ondragover = function(event){
    event.preventDefault();
    ...
}
```

> `dragover`事件中必须要执行`event.preventDefault()`，否则不会触发`drop`事件

- `dragleave`

`dragleave`事件在源对象拖动离开目标对象时触发，其语法结构是：

```
HTMLElement.ondragleave = function(){
    ...
}
```

• drop

drop事件在源对象在目标对象上释放时触发，其语法结构是：

```
HTMLElement.ondrop = function(){
    ...
}
```

> 在drop事件中"必须"也要执行event.preventDefault()和event.stopPropagation()，否则源对象的行为可能会触发（如拖动时图片的话，在Firefox浏览器会新打开标签页）

按正常的触发顺序来理解：

dragstart(1) -> drag(>=1) -> dragenter(>=1) -> dragover(>=1) ->drop(1) -> dragend(1)

1.5 dataTransfer属性

dragEvent接口的dataTransfer属性将返回DataTransfer对象，用于保存拖动数据，其语法结构是：

```
DataTransfer DragEvent.dataTransfer
```

2.DataTransfer对象

在进行拖放操作时，DataTransfer对象用来保存拖动的数据，它可以保存一项或多项数据。

2.1 方法

• setData()方法

setData()方法用于为一个给定的类型设置数据，如果类型不存在，则自动添加到末尾，如果存在的话，则替换数据，其语法结构是：

```
DataTransfer.setData(类型,数据)
```

> 所谓的类型就是一个键名

• getData()方法

getData()方法用于返回指定类型的数据，如果类型不存在或者没有数据，将返回空字符串，其语法结构是：

```
string DataTransfer.getData(类型)
```

3.Multer中间件

3.1 概述

对于存在<input type="file">元素的表单有以下要求：

> 表单的提交方式只能为POST
>
> 必须设置表单的enctype="multipart/form-data"属性

Multer是一个Node.js中间件，用于处理multipart/form-data类型的表单数据

Multer中间件会添加一个body对象及files或file对象到Request对象

为了避免冲突，Multer会自动修改上传文件名称（但是没有扩展名）

如果要修改文件名就需要通过diskStorage()方法设置本地存储规则

3.2 安装

```
npm install --save multer
```

3.3 使用Multer

```
var multer = require('multer');

var upload = multer({
        dest:'目标位置'
});
```

3.4 请求参数

- single(filename)

接收以filename（其实就是<input type="file">的name）上传的文件， 上传文件信息存储在req.file对象内，其语法结构如：

```
var multer = require('multer');
var upload = multer({dest:'upload/'});

app.post('/post',upload.single('浏览框的名称'),(req,res)=>{

});
```

- array(filename)

接收以`filename`（其实就是`<input type="file">`的`name`）上传的文件，所有上传文件的信息存储在`res.files`对象内，其语法结构如：

```
var multer = require('multer');
var upload = multer({dest:'upload/'});

app.post('/post',upload.array('浏览框的名称'),(req,res)=>{

});
```

• `diskStorage()`方法

```
var path = '/112233';
var storage = multer.diskStorage({
    //定义上传到目录的相关规则
    //req代表HTTP Request对象(如果在上传目录的相关规则中要使用HTTP请求对象)
    //file代表的上传文件的对象
    //cb(callback,回调函数)
    destination:function(req,file,cb){
        cb(null,path);
    },
    //定义上传文件的相关规则
    filename:function(req,file,cb){
        ...
        cb(null,filename);
    }
});
```

使用规则

```
var upload = multer({storage:自定义规则});
```

• 文件信息

每个上传文件包括以下信息：

- `originalname`，上传文件的原始名称（包含扩展名）
- `size`，上传文件字节数（以字节为单位）