第5章 HTML5 拖放 API

本章主要介绍 HTML5 拖放 API 的功能与应用。HTML5 拖放 API 可以用于拖曳网页中的元素并放置到页面上的指定区域，也可以直接将本地计算机上的文件拖放到网页中。HTML5 拖放 API 增强了页面友好度，使用该技术可以开发出用户体验良好的人机交互界面。

本章学习目标

- 了解拖放的概念；
- 熟悉拖放事件 DragEvent；
- 熟悉 DataTransfer 对象；
- 掌握拖放 HTML 元素的方法；
- 掌握拖放本地文件的方法。

5.1 HTML5 新增拖放 API

HTML5 拖放 API 规定了所有元素都可以被拖放。具体来说，HTML5 定义的拖放这一行为指的是用户可以使用鼠标左键点击选中允许拖放的元素或文件，在保持鼠标左键按下的情况下可以移动该元素至页面的任意位置，并且在移动到处于具有允许放置状态的元素上释放鼠标左键放置被拖放的元素。其中从鼠标左键按下选中元素，到保持鼠标左键按下并移动该元素的整个过程称为“拖”；将被拖动的元素放置在许可放置的区域上方并释放鼠标左键的行为称为“放”。整个拖放过程增强了人机交互的功能。

5.2 浏览器支持情况

主流浏览器对 HTML5 拖放 API 的支持情况如表 5-1 所示。

表 5-1 主流浏览器对 HTML5 拖放 API 的支持情况

浏览器	IE	Firefox	Chrome	Safari	Opera
支持情况	9.0 及以上版本	4.0 及以上版本	20 及以上版本	5.0 及以上版本	12 及以上版本

由此可见，目前所有的主流浏览器均支持 HTML5 拖放 API。

5.3 HTML5 拖放 API 的应用

5.3.1 DragEvent 事件

拖放元素时的一系列动作会触发相关元素的拖放事件 DragEvent，该事件继承于鼠标

事件 MouseEvent。DragEvent 包含的常用事件类型如表 5-2 所示。

表 5-2 DragEvent 的常用事件一览表

事件名称	存储模式	事件目标	解释
ondragstart	读写模式	该事件由被拖曳的元素触发	当用户刚开始拖动元素时触发该事件
ondrag	保护模式	该事件由被拖曳的元素触发	当元素处于被拖动状态时触发该事件
ondragenter	保护模式	该事件由被拖曳的元素触发	当被拖动的元素进入到可以被放置下来的有效区域的瞬间触发该事件
ondragleave	保护模式	该事件由被拖曳的元素触发	当被拖动的元素离开了可以被放置下来的有效区域的瞬间触发该事件
ondragover	保护模式	该事件由目标区域元素触发	当被拖动的元素处于可以被放置下来的有效区域内时，该事件会不停地被触发。该事件状态在 dragenter 之后，在 dragleave 之前
ondrop	只读模式	该事件由目标区域元素触发	当被拖动的元素被放置在有效的区域时触发该事件
ondragend	保护模式	该事件由被拖曳的元素触发	当拖动操作结束时激发该事件。例如，在拖动元素的过程中释放鼠标左键或按下键盘上的 Esc 键均可触发该事件。该事件状态在 drop 之后

其中只有 ondragstart 事件为读写模式，ondrop 事件为只读模式，其余所有事件均为保护模式状态。在读写模式下既可以写入数据进行传递也可以读取数据；在只读模式下，只允许将数据读取出来，不可以写入新的数据；在保护模式下，当前传递的数据不可以被修改或读取。

从用户在元素上点击鼠标左键开始拖曳行为，到将该元素放置到指定的目标区域中的整个拖放生命周期触发的事件按照顺序如下：

```
dragstart -> drag -> dragenter -> dragover -> dragleave -> drop -> dragend
```

5.3.2 DataTransfer 对象

HTML5 拖放 API 允许在拖放过程中携带一项或多项自定义数据内容。这些数据内容可以使用拖放事件 DragEvent 中的 datatransfer 属性进行添加和处理，该属性来源于 HTML5 中的 DataTransfer 对象，其中包含的每项数据均可有独立的数据类型。

DataTransfer 对象的常用属性如表 5-3 所示。

表 5-3 DataTransfer 对象的属性一览表

属性名称	属性值	解释
dropEffect	none copy move link	该属性用于获取或重置当前的拖放类型，共有 4 种取值

续表

属性名称	属性值	解释
effectAllowed	none copy copyLink copyMove link linkMove move all uninitialized	提供所有允许的拖放类型
types	DOMString[]	该属性为只读属性。返回值为字符串数组，包含了所有存入数据的类型
items	DataTransferItemList 对象	该属性为只读属性。返回值为 DataTransferItemList 对象，该对象是以列表的形式保存所有的存入数据
files	FileList 对象	该属性为只读属性。如果拖放的是一个或多个本地文件，则该属性返回值为文件列表对象。如果拖放过程中没有涉及本地文件，则文件列表为空

DataTransfer 对象的常用方法如表 5-4 所示。

表 5-4 DataTransfer 对象的方法一览表

方法名称	解释
getData(format)	获取 DataTransfer 对象中 format 格式的数据。一般在 ondrop 事件中使用，获取传递的数据内容。其中 format 替换成某种数据类型，例如，纯文本类型为 text/plain
setData(format, data)	将数据设置为 format 格式，并保存在 DataTransfer 对象中进行传递。一般在 ondragstart 事件中使用，设置需要传递的数据内容
clearData([format])	清除 DataTransfer 对象中 format 格式的数据。如果省略参数，则表示清除全部数据
setDragImage(image,x,y)	设置拖曳元素时所显示的自定义图标。其中 image 为图片对象，x 和 y 分别指的是图标与鼠标在水平和垂直方向上的距离

5.3.3 拖放元素过程

在 HTML5 页面中实现拖放的主要过程如下：

- 为需要被拖放的元素添加 draggable 属性，使其允许被拖放。
- 在被拖曳元素的 ondragstart 事件中初始化需要传递的数据信息。
- 为作为放置区域的元素设置 ondragover 事件，取消默认操作。
- 为作为放置区域的元素设置 ondrop 事件，接收并处理传递过来的数据内容。

1. 设置元素可拖放状态

在 HTML5 中规定所有元素都支持可拖放属性 draggable，该属性值可以用于定义元素是否为可拖放状态。当 draggable 属性值设置为 true 时表示元素为可拖放状态，设置为 false 时表示元素不可以被拖放。

例如，将一个段落元素变为可拖放状态：

```
<p draggable = "true">
    这是一个可以被拖放的段落元素。
</p>
```

注意：draggable 属性值在声明时不可以被省略。例如，<p draggable>就是错误的写法，必须加上完整的布尔值(true 或者 false)。本示例中的<p draggable="true">为正确的写法。

如果没有为元素设置 draggable 属性，则默认值为 auto 表示元素是否允许拖曳取决于浏览器的默认设置。一般情况下，只有图片元素<img>和带有 href 属性的超链接元素<a>无须设置 draggable 属性即可被拖放，其他元素可以通过设置 draggable 属性值为 true 来实现可拖放状态。

【例 5-1】 设置可拖放元素

使用 draggable 属性为元素设置可拖放状态。


视频讲解

```
1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <meta charset = "utf-8">
5.      <title>HTML5 拖放 API 之设置可拖放元素</title>
6.      <style>
7.      p{
8.          width:100px;                    /* 设置段落元素宽 100 像素 */
9.          height:100px;                   /* 设置段落元素高 100 像素 */
10.         background-color:yellow;        /* 设置段落元素背景色为黄色 */
11.     }
12.     </style>
13.   </head>
14.   <body>
15.   <h3>HTML5 拖放 API 之设置可拖放元素</h3>
16.   <hr />
17.   <p draggable = "true">这是一个可拖放的段落元素.</p>
18.   </body>
19. </html>
```

运行效果如图 5-1 所示。

【代码说明】

本示例包含了一个段落元素<p>，用于演示元素的拖曳效果。将该元素的 draggable 属性值设置为 true 表示允许拖放。为了更清晰地表达显示效果，本示例在<head>首尾标签之间为段落元素<p>添加了 CSS 内部样式表进行样式设置，规定了该段落元素为宽 100 像素、高 100 像素的矩形样式，并且定义了其背景颜色为黄色。

当前使用 draggable 属性实际上只能实现元素的拖曳效果，目前尚不能放置元素到其他指定区域，也不能在拖曳过程中传递有效的数据。这些功能需要配合拖放元素的事

图 5-1 元素的拖曳效果

件 DragEvent 设置自定义回调函数来实现。

2. 为被拖曳元素传递数据

使用 ondragstart 事件监听元素刚被拖动的状态，此时可以为 ondragstart 事件设置自定义名称的回调函数。例如，设置一个自定义函数 drag()来处理 ondragstart 事件：

```
<p draggable = "true" ondragstart = "drag(event)">
    这是一个可以被拖放的段落元素。
</p>
```

当用户开始拖动元素时，元素的 ondragstart 事件会被触发并调用 drag()函数来传递事件参数 event 对象，其中 event.datatransfer 属性用于在拖放过程中传递数据。

DataTransfer 对象的 setData()方法可以用于为拖放事件添加不同类型的数据，包括纯文本、超链接、HTML 代码等。其语法格式如下：

```
dataTransfer.setData(format, data)
```

其中参数 format 用于填写数据类型，参数 data 用于填写需要传递的数据内容。

可用于传递的常用数据类型如下：

- 纯文本类型——text/plain；
- 超链接类型——text/uri-list；
- HTML 代码类型——text/html。

例如，在之前 ondragstart 事件的回调函数 drag()中设置传递的数据：

```
function drag(ev){
    ev.dataTransfer.setData("text/plain","Hello HTML5");                //纯文本数据
    ev.dataTransfer.setData("text/uri - list","http://www.test.com");   //超链接数据
    ev.dataTransfer.setData("text/html","< h3 > Hello HTML5 </h3 >");   //HTML 代码数据
}
```

这些传递数据目前只能在 ondragstart 事件的回调函数中进行设置，从页面上来看没有什么不同。后续需要在放置元素时使用 datatransfer 对象的 getDate()方法进行获取数据才能显示其作用。

其中在触发 ondragstart 事件时可以读写数据。在触发 ondrop 事件时为只读模式，可以读取数据。其余所有事件状态下均为保护模式，不可以读写数据。

在 event 事件中的 target 属性表示被拖曳的元素对象，因此可以利用纯文本类型传递元素对象的 id 名称。仍然以 ondragstart 事件的回调函数 drag()为例：

```
function drag(ev){
    ev.dataTransfer.setData("text/plain",ev.target.id); //纯文本数据，用于传递元素 id 名称
}
```

这样在可放置元素的目标区域就可以使用 getData()方法获取此 id 名称，并且使用 JavaScript 中的 document.getElementById()方法获得并处理被拖曳元素的对象。

3. 定义可放置元素的目标区域

由于被拖动的元素不可以放置在未定义的区域，因此需要将指定的元素定义为可放置区域才能用于放置被拖动的元素。作为可放置区域的元素必须带有 ondragover 事件，用于监听是否有可拖放的元素进入了目标区域。

例如，将另外一个段落元素< p >设置为可放置区域，代码如下：

```
<p ondragover = "allowDrop(event)">
    这是一个可以放置被拖曳元素的段落区域。
</p>
```

其中回调函数 allowDrop()和 drop()名称均可自定义,表示当前事件触发时的处理操作。

默认情况下无法将元素放置在其他元素中,因此需要在放置区域 ondragover 事件的回调函数中使用 event. preventDefault()方法阻止默认处理。

以前面的自定义函数 allowDrop()为例,相关 JavaScript 代码如下:

```
function allowDrop(ev){
    event.preventDefault();                    //阻止默认处理方式
}
```

event. preventDefault()方法可以禁用默认处理,因此在执行该方法后指定的区域允许用于放置被拖曳的元素。

4. 接收被拖曳元素的传递数据

当松开鼠标左键进行放置元素时,放置区域的 ondrop 事件被触发。此时可以使用 datatransfer 对象中的 getData()方法获取传递的数据内容。

例如,为前面的放置区域<p>添加 ondrop 事件用于接收数据,代码如下:

```
<p ondragover = "allowDrop(event)" ondrop = "drop(event)">
    这是一个可以放置被拖曳元素的段落区域。
</p>
```

其中回调函数 drop()名称可自定义,表示当前事件触发时的处理操作。

由于 ondrop 事件的默认行为是以超链接的形式打开数据,所以同样首先需要使用 event. preventDefault()方法阻止原先的默认处理方式,然后可以在 ondrop 事件的回调函数中自定义需要处理的内容。例如:

```
function drop(ev){
    event.preventDefault();                                    //阻止默认处理方式
    var data = event.datatransfer.getData("text");             //获取传递的文本类型数据
}
```

这里数据格式如果简写为 text 或 url 会被自动转换为 text/plain 类型或 text/uri-list 类型。如果没有找到指定的数据内容,则返回一个空字符串。

【例 5-2】 拖放元素的简单应用

将段落元素<p>拖放到 id="container"的 元素中。


视频讲解

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <meta charset = "utf-8">
5.          <title>HTML5 拖放 API 的简单应用</title>
6.          <style>
7.              p {
8.                  width: 100px;                  /* 设置段落元素宽 100 像素 */
9.                  height: 100px;                 /* 设置段落元素高 100 像素 */
10.                 background-color: yellow;      /* 设置段落元素背景色为黄色 */
```

```
            }
            div#container {
                border: 1px solid;
                width: 200px;
                height: 200px;
            }
        </style>
    </head>
    <body>
        <h3>HTML5 拖放 API 的简单应用</h3>
        <hr />
        <p id="test" draggable="true" ondragstart="drag(event)">
            这是一个可以被拖放的段落元素
        </p>
        <div id="container" ondragover="allowDrop(event)" ondrop="drop(event)">
            这是一个可以用于放置被拖放元素的区域
        </div>
        <script>
            //ondragstart 事件回调函数
            function drag(ev) {
                //设置传递的内容为被拖曳元素的 id 名称,数据类型为纯文本类型
                ev.dataTransfer.setData("text/plain", ev.target.id);
            }

            //ondragover 事件回调函数
            function allowDrop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
            }

            //ondrop 事件回调函数
            function drop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
                //获取当前被放置的元素 id 名称
                var id = ev.dataTransfer.getData("text");
                //根据 id 名称获取元素对象
                var p = document.getElementById(id);
                //获取文件夹区域并添加该元素对象
                ev.target.appendChild(p);
            }
        </script>
    </body>
</html>
```

运行效果如图 5-2 所示。

【代码说明】

本示例包含了一个段落元素<p>,用于演示元素的拖曳效果。将 id="test"的段落元素<p>的 draggable 属性值设置为 true 表示允许拖放,并且将 id="container"的 元素设置为可放置区域。在拖动元素的过程中传递了段落元素的 id 名称,并且在放置元素时获取该 id 名称并且将被拖曳的段落元素移动到作为可放置区域的 元素中。

(a) 页面初始加载效果　(b) 元素拖动过程　(c) 元素放置效果

图 5-2　段落元素的拖曳效果

5.3.4　自定义拖放图标

使用 DataTransfer 对象中的 setDragImage()方法可以自定义拖曳时显示的图标。其语法格式如下：

```
setDragImage(image, x, y);
```

其中参数 image 表示 Image 对象，代表图标的来源。参数 x 和 y 分别表示图标与鼠标在水平方向和垂直方向上的距离。该方法一般用于 ondragstart 事件的回调函数中，表示从拖动动作开始时更改拖放图标。例如：

```
function drag(ev){
    var img = new Image();
    img.src = "image/star.jpg";
    ev.dataTransfer.setDragImage(img, 10, 10);
}
```

其中变量 img 为图片对象，指定的图片素材来源于本地的 image 文件夹中的 star.jpg。

【例 5-3】　定义拖曳的数据和图标


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>HTML5 拖放 API 之设置可拖放元素</title>
    <style>
    p{
        width:100px;                    /*设置段落元素宽 100 像素*/
        height:100px;                   /*设置段落元素高 100 像素*/
        background-color:yellow;        /*设置段落元素背景色为黄色*/
    }
    </style>
  </head>
  <body>
  <h3>HTML5 拖放 API 之设置可拖放元素</h3>
  <hr />
  <p draggable="true" ondragstart="drag(event)">这是一个可拖放的段落元素.</p>
  <script>
  function drag(ev){
```

```
        var img = new Image();
        img.src = "image/star.jpg";
        ev.dataTransfer.setDragImage(img, 5, 5);
    }
    </script>
    </body>
</html>
```

运行效果如图 5-3 所示。

图 5-3 元素的拖曳效果(自定义图标)

【代码说明】

本示例包含了一个段落元素<p>用于演示元素的拖曳效果。将该元素的 draggable 属性值设置为 true 表示允许拖放,并且使用了 setDragImage()方法设置了本地 image 目录中的 star.jpg 图片作为拖曳时显示的图标内容。

5.3.5 自定义拖放行为

DataTransfer 对象具有 effectAllowed 和 dropEffect 属性用于规定拖放行为,当对元素进行拖放时,共有三种常见效果解释如下:

- copy——表示被拖曳的数据将从它的初始位置复制到可放置区域。
- move——表示被拖曳的数据将从它的初始位置移动到可放置区域。
- link——表示被拖曳的数据将从它的初始位置链接一个快捷方式到可放置区域。

这三种效果根据组合又可以形成不同的样式要求,不同的拖放行为对应显示的鼠标图标样式各不相同,具体样式由浏览器和操作系统决定。一般可以在 ondragstart 事件被触发时通过设置 effectAllowed 属性值来规定允许进行何种操作。例如:

```
ev.dataTransfer.effectAllowed = "move";
```

上述代码表示设置允许的操作为移动,effectAllowed 的属性值只能在 ondragstart 事件中进行设置。

effectAllowed 属性共有如下 9 种取值:

- none——不允许任何操作;
- copy——只允许复制操作;
- copyLink——允许复制或者链接;
- copyMove——允许复制或者移动;
- link——只允许链接操作;
- linkMove——允许链接或移动;
- move——只允许移动操作;
- all——允许所有(复制、移动或链接)操作;
- uninitialized——尚未设置 effectAllowed 属性时的默认值,等同于 all。

在拖曳元素的过程中,dropEffect 属性值可以在 dragenter 或 dragover 事件中进行设置。

dropEffect 属性共有如下 4 种取值:

- none——不允许任何操作。
- copy——该状态下被拖曳的元素将复制一个副本放到指定的放置区域。
- move——该状态下被拖曳的元素将移动到指定的放置区域,该属性值为默认值。
- link——该状态下被拖曳的元素与可放置区域之间将创建连接。

dropEffect 属性的取值会受到 effectAllowed 属性取值的约束。例如上面示例中设置 effectAllowed 属性值为 move 时,dropEffect 的属性值也只能设置为 move。effectAllowed 与 dropEffect 属性取值的具体对应关系如表 5-5 所示。

表 5-5 effectAllowed 与 dropEffect 属性取值对照表

effectAllowed 设置的取值	dropEffect 允许的取值
none	none
copy	copy
copyLink	copy 或 link
copyMove	copy 或 move
link	link
linkMove	link 或 move
move	move
all	copy、link 或 move 任选其一
uninitialized 并且被拖曳的元素为文本框中的内容	move 或 copy
uninitialized 并且被拖曳对象为普通元素	copy 或 link
uninitialized 并且被拖曳对象为带 href 属性的超链接元素<a>	link 或 copy

【例 5-4】 自定义拖放行为

在多个可拖曳元素 ondragstart 事件的回调函数中设置不同的 effectAllowed 属性值以查看鼠标指针的显示效果。


视频讲解

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML5 拖放 API 之自定义拖放行为</title>
        <style>
            p {
                width: 100px;                    /*设置段落元素宽 100 像素*/
                height: 100px;                   /*设置段落元素高 100 像素*/
                background-color: lightblue;     /*设置段落元素背景色为浅蓝色*/
                float:left;
                margin:10px;
                text-align:center;
            }
            div#container {
                border: 1px solid;
                width: 340px;
                height: 100px;
                clear:both;
                margin:10px;
                text-align:center;
            }
        </style>
    </head>
```

```
    <body>
        <h3>HTML5 拖放 API 之自定义拖放行为</h3>
        <hr />
        <p id="test1" draggable="true" ondragstart="drag1(event)">
            拖曳效果<br>move
        </p>
        <p id="test2" draggable="true" ondragstart="drag2(event)">
            拖曳效果<br>copy
        </p>
        <p id="test3" draggable="true" ondragstart="drag3(event)">
            拖曳效果<br>link
        </p>
        <div id="container" ondragover="allowDrop(event)" ondrop="drop(event)">
            可放置区域
        </div>
        <script>
            //ondragstart 事件回调函数
            function drag1(ev) {
                var dataTrans = ev.dataTransfer;
                dataTrans.effectAllowed = "move";
                //设置传递的内容为被拖曳元素的 id 名称,数据类型为纯文本类型
                dataTrans.setData("text/plain", "ev.target.id");
            }
            function drag2(ev) {
                var dataTrans = ev.dataTransfer;
                dataTrans.effectAllowed = "copy";
                //设置传递的内容为被拖曳元素的 id 名称,数据类型为纯文本类型
                dataTrans.setData("text/plain", ev.target.id);
            }
            function drag3(ev) {
                var dataTrans = ev.dataTransfer;
                dataTrans.effectAllowed = "link";
                //设置传递的内容为被拖曳元素的 id 名称,数据类型为纯文本类型
                dataTrans.setData("text/plain", ev.target.id);
            }

            //ondragover 事件回调函数
            function allowDrop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
            }

            //ondrop 事件回调函数
            function drop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
            }
        </script>
    </body>
</html>
```

运行效果如图 5-4 所示。

【代码说明】

本示例包含了三个可拖曳段落元素<p>,用于对比不同拖放行为导致鼠标指针显示效果的区别,其 id 名称分别为 test1、test2 和 test3。为这三个段落元素添加 ondragstart 事件,并定义回调函数名称分别为 drag1(event)、drag2(event)和 drag3(event),在其中分别设

(a) move行为的显示效果　(b) copy行为的显示效果　(c) link行为的显示效果

图 5-4　自定义拖放行为的不同显示效果

置 effectAllowed 属性值为 move、copy 和 link。在页面上设置一个 id="container"的可放置区域 ,用于放置本示例的三个测试段落元素。

由图 5-4 可见,这三种拖放行为均显示正常的鼠标指针,以及指针下方会显示一个空心矩形框。不同的拖放行为会导致鼠标指针显示的样式稍有区别:move 行为没有显示其他特殊内容;copy 行为会在空心矩形的右下角显示一个加号符号的小图标;link 行为会在空心矩形的右下角显示带一个箭头符号的小图标。

5.3.6　本地文件的拖放

除了页面上自带的 HTML 元素外,本地文件也可以使用 HTML5 拖放 API 进行拖曳并放置到页面的指定区域中。传递本地文件时无须设置传递的数据内容,直接在放置文件时使用 DataTransfer 对象的 files 属性即可获取文件列表,里面包含了所有文件。

【例 5-5】　HTML5 拖放 API 之本地文件拖放

将本地文件拖放至页面的指定区域,使用 DataTransfer 对象的 files 属性将文件相关信息(例如,文件名称、修改时间、文件大小等内容)显示在页面上。


视频讲解

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>HTML5 拖放 API 之本地文件拖放</title>
        <style>
            #fileCheck {
                width: 300px;
                height: 100px;
                border: 1px dashed;
                margin: 20px;
            }
            li {
                margin: 10px;
            }
        </style>
    </head>
    <body>
        <h3>HTML5 拖放 API 之本地文件拖放</h3>
        <hr />
        <div id="fileCheck" ondragover="allowDrop(event)" ondrop="drop
        (event)">
            请将文件拖放至此处。
```

```
        </div>
        <div id="status"></div>
        <script>
            //ondragover 事件回调函数
            function allowDrop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
            }

            //ondrop 事件回调函数
            function drop(ev) {
                //解禁当前元素为可放置被拖曳元素的区域
                ev.preventDefault();
                //获取拖曳的文件列表
                var files = ev.dataTransfer.files;
                //用于记录文件的状态,包括文件名、文件大小、修改时间等
                var fileStatus;
                //用于获取单个文件对象
                var f;
                //使用 for 循环遍历所有文件
                for (var i = 0; i < files.length; i++) {
                    //获取当前文件对象
                    f = files[i];
                    //获取最近修改文件的日期对象
                    var lastModified = f.lastModifiedDate;
                    //将日期时间显示为纯文本形式
                    var lastModifiedStr = lastModified.toLocaleString();

                    //组合文件相关信息
                    fileStatus += '<li>文件名称:' + f.name + '<br>文件类型:'
                     + f.type + '<br>文件大小:' + f.size + '字节<br>修改时间:'
                     + lastModifiedStr + '</li>';
                }
                //获取文件状态显示栏对象
                var status = document.getElementById("status");
                //更新文件信息至显示栏中
                status.innerHTML = '<ul>' + fileStatus + '</ul>';
            }
        </script>
    </body>
</html>
```

运行效果如图 5-5 所示。

(a) 页面初始加载效果　(b) 拖曳本地文件的过程　(c) 放置文件后的效果

图 5-5　本地文件的拖放效果

【代码说明】

本示例包含了一个 id="fileCheck"的区域元素 作为本地文件的放置区域，并在CSS 内部样式表中为其设置样式：宽 300 像素、高 100 像素，带有 1 像素宽的虚线边框，且各边的外边距为 20 像素。其下方还有一个 id="status"的 元素用于显示被拖曳放置的本地文件信息，由于初始状态尚无文件被放置，因此该元素内部为空。显示效果如图 5-5(a)所示。

为该元素添加 ondragover 与 ondrop 事件，分别用于监听是否有元素进入该区域，以及是否有元素放置在该区域。这两个事件的回调函数分别为 allowDrop(event)和 drop(event)，函数名称均可自定义。在这两个函数中均使用 ev.preventDefault()表示允许元素放置在当前区域。本地文件的拖曳过程由图 5-5(b)所示。

在 drop(event)函数中使用 dataTransfer 对象的 files 属性获取拖曳文件列表，由于拖曳规则是允许每次拖曳一个或同时拖曳多个文件进行放置，因此该对象返回值为数组的形式。使用 for 循环遍历本次拖放的所有本地文件，并将每个文件的相关信息累加到 fileStatus 变量中去。最后将 fileStatus 变量的全部内容更新到 id="status"的 元素中，从而实时显示在页面上。放置本地文件后的页面效果如图 5-5(c)所示。

5.4 本章小结

HTML5 新增拖放 API 可以用于拖曳和放置所有指定的 HTML 元素。所有 HTML 元素均可以被设置为可拖放状态，并且可以将其放置到指定区域中。HTML5 拖放 API 中包含了 DragEvent 事件与 DataTransfer 对象。其中 DragEvent 事件包含了从开始拖曳到放置完成的一系列拖放事件。DataTransfer 对象中 setData()方法可用于在拖曳过程中设置传递的数据，而 getData()方法可用于在放置过程中获取传递的数据。

HTML5 拖放 API 可以自定义拖放时显示的图标与不同拖放行为，不同的拖放行为(例如复制、移动、链接等动作)所显示的鼠标指针样式也有所区别。HTML5 拖放 API 还允许拖放本地文件，并且获取文件的名称、修改时间、大小等相关信息。

习题 5

1. 如何将元素设置为允许拖放的状态？
2. 元素被拖曳直到放置在指定区域的完整过程中依次触发了哪些拖放事件？
3. 可在拖放过程中被传递的常见数据类型有哪些？
4. 使用 DataTransfer 对象中的何种方法可以自定义拖放图标？
5. 如何将指定元素设置为允许放置元素的目标区域？
6. 在进行本地文件的拖放时，DataTransfer 对象中的哪个属性可以用于获取文件列表？