拖放

拖放是一种常见的特性，即抓取对象以后拖到另一个位置。

实例

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function allowDrop(ev)
{
ev.preventDefault();
}

function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}

function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)"
ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69" />

</body>
</html>
```

设置元素为可拖放

把 draggable 属性设置为 true ：

```
<img draggable="true" />
```

拖动什么 - ondragstart 和 setData()

然后，规定当元素被拖动时，会发生什么。

在上面的例子中，ondragstart 属性调用了一个函数，drag(*event*)，它规定了被拖动的数据。

dataTransfer.setData() 方法设置被拖数据的数据类型和值：

```
function drag(ev)
{
ev.dataTransfer.setData("Text",ev.target.id);
}
```

在这个例子中，数据类型是 "Text"，值是可拖动元素的 id ("drag1")。

放到何处 - ondragover

ondragover 事件规定在何处放置被拖动的数据。

默认地，无法将数据/元素放置到其他元素中。如果需要设置允许放置，我们必须阻止对元素的默认处理方式。

这要通过调用 ondragover 事件的 *event*.preventDefault() 方法：

```
event.preventDefault()
```

进行放置 - ondrop

当放置被拖数据时，会发生 drop 事件。

在上面的例子中，ondrop 属性调用了一个函数，drop(*event*)：

```
function drop(ev)
{
ev.preventDefault();
var data=ev.dataTransfer.getData("Text");
ev.target.appendChild(document.getElementById(data));
}
```

代码解释：

- 调用 preventDefault() 来避免浏览器对数据的默认处理（drop 事件的默认行为是以链接形式打开）
- 通过 dataTransfer.getData("Text") 方法获得被拖的数据。该方法将返回在 setData() 方法中设置为相同类型的任何数据。
- 被拖数据是被拖元素的 id ("drag1")
- 把被拖元素追加到放置元素（目标元素）中