在HTML5中，<img>元素和<a>元素默认是可以拖动的，其他元素要想被拖动的话，需要给元素设置一个draggable属性，属性值设置为true, 标签就可以拖拽了。

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

拖拽元素事件

拖拽元素事件： 事件对象为被拖拽元素

- ondragstart，开始拖拽时触发
- ondrag, 拖拽的过程中连续触发
- ondragend, 拖拽结束时触发

```
<div id="box1" draggable="true">box1</div>
<script>
  // 为被拖拽的元素注册事件
  var box1 = document.querySelector('#box1');
      // ondragstart 开始拖拽时触发
      box1.ondragstart = function (evt) {
          console.log(evt.type)
      };
      // ondrag 拖拽的过程中触发
      box1.ondrag = function (evt) {
          console.log(evt.type)
      };
      // ondragend 拖拽结束时触发
      box1.ondragend = function (evt) {
          console.log(evt.type)
      };
</script>
```

目标元素事件

目标元素事件： 事件对象为目标元素

- ondragenter ， 鼠标进入目标元素时触发
- ondragover，鼠标在目标上时连续触发
- ondragleave ， 鼠标离开目标元素时触发
- ondrop，鼠标在目标元素上释放时触发

```
<div id="box2">box2</div>
<script>
  // 目标元素可以注册的事件

  var box2 = document.querySelector('#box2');
      // ondragenter  鼠标进入目标元素时出触发
      box2.ondragenter = function (evt) {
          console.log(evt.type)
      };
      // ondragover   鼠标拖拽到目标元素上时连续触发
      box2.ondragover = function (evt) {
          console.log(evt.type)
      };
      // ondragleave  鼠标离开目标元素时触发
      box2.ondragleave = function (evt) {
          console.log(evt.type)
      };
      // ondrop 鼠标在目标元素上释放时触发
      box2.ondrop = function (evt) {
          console.log(evt.type)
      };
</script>
```

在拖动元素时，显示了一个“禁止”标志，这表明“被拖动元素”拖动到“目标元素”上时，目标元素并不接受被拖动元素。这是因为当被拖动元素被“拖过”document对象时，document对象默认阻止了拖动事件，而其他HTML元素也是位于document对象内的，因此它们也不接受“放”。

为了让 document 可以接受“放”，应该为 document 注册 ondragover 和 ondrop 事件，在事件的处理函数中取消 document 对拖动事件的默认行为。如下面代码所示：

```
document.ondragover = function(e){
  // 取消事件的默认行为
  e.preventDefault();
}
document.ondrop = function(e){
  //取消事件的默认行为
  e.preventDefault();
}
```

携带数据

拖拽事件的事件对象有一个 dataTransfer 对象，通过该对象提供的方法，可以让拖动操作携带数据。

- `setData()` 存放数据
- `getData()` 获取数据
- `clearData()` 清除所有数据，或某项数据

在拖拽开始时，使用 `setData()` 方法存放数据

```
var box1 = document.querySelector('#box1');
    box1.ondragstart = function (evt) {
        // 存放数据
        evt.dataTransfer.setData('name', '张三');
        evt.dataTransfer.setData('age', '20');
    };
```

在拖放结束时，使用 `getData()` 方法获取数据

```
var box2 = document.querySelector('#box2');
    box2.ondrop = function (evt) {
      // 获取数据
        console.log(evt.dataTransfer.getData('name'));
        console.log(evt.dataTransfer.getData('age'));
    };
```

拖拽外部文件

如果拖拽的不是页面上的元素，而是本地的文件，此时，拖拽进来的文件会被放入到 `dataTransfer` 对象的 `files` 属性值中，该值是 `FileList` 对象实例，`FileList` 对象中保存的是 `File` 对象。

```
box1.ondrop = function (evt) {
    evt.preventDefault();

    // event.dataTransfer.files 属性的值是 FileList 对象,
    // FileList 中保存的都是拖拽进来的 File 对象。
    console.log(evt.dataTransfer.files);
};
```

我们可以使用 `FileReader` 对象读取文件中的内容。

```
box1.ondrop = function (evt) {
    evt.preventDefault();

    var fileList = evt.dataTransfer.files;

    // 使用 FileReader 对象读取文件中内容
    var reader = new FileReader();
        reader.onload = function () {
            console.log(reader.result)
        };
      // 将文件中的内容读取成 base64 编码的形式
        reader.readAsDataURL(fileList[0]);
};
```