

[首页](#) [问答](#) [头条](#) [专栏](#) [讲堂](#) [发现](#) [Q](#)

原生拖拽,拖放事件(drag and drop)

赞 10

已收藏 65

 html5 [qiu_deqing](#) 2015年05月31日发布

17.4k 次浏览

原生拖拽,拖放事件(drag and drop)

拖拽,拖放事件可以通过拖拽实现数据传递,达到良好的交互效果,如:从操作系统拖拽文件实现文件选择,拖拽实现元素布局的修改.

drag and drop事件流程

一个完整的drag and drop流程通常包含以下几个步骤:

1. 设置可拖拽目标.设置属性 `draggable="true"` 实现元素的可拖拽.
2. 监听 `dragstart` 设置拖拽数据
3. 为拖拽操作设置反馈图标(可选)
4. 设置允许的拖放效果,如 `copy` , `move` , `link`
5. 设置拖放目标,默认情况下浏览器阻止所有的拖放操作,所以需要监听 `dragenter` 或者 `dragover` 取消浏览器默认行为使元素可拖放.
6. 监听 `drop` 事件执行所需操作

拖拽事件

以下是拖拽产生的一系列事件,拖拽事件产生时不会产生对应的鼠标事件.

- `dragstart` :拖拽开始时在被拖拽元素上触发此事件,监听器需要设置拖拽所需数据,从操作系统拖拽文件到浏览器时不触发此事件.
- `dragenter` :拖拽鼠标进入元素时在该元素上触发,用于给拖放元素设置视觉反馈,如高亮
- `dragover` :拖拽时鼠标在目标元素上移动时触发.监听器通过阻止浏览器默认行为设置元素为可拖放元素.
- `dragleave` :拖拽时鼠标移出目标元素时在目标元素上触发.此时监听器可以取消掉前面设置的视觉效果.
- `drag` :拖拽期间在被拖拽元素上连续触发
- `drop` :鼠标在拖放目标上释放时,在拖放目标上触发.此时监听器需要收集数据并且执行所需操作.如果是从操作

系统拖放文件到浏览器,需要取消浏览器默认行为.

- `dragend`: 鼠标在拖放目标上释放时,在拖拽元素上触发.将元素从浏览器拖放到操作系统时不会触发此事件.

DataTransfer对象

拖拽事件周期中会初始化一个 `DataTransfer` 对象,用于保存拖拽数据和交互信息.以下是它的属性和方法.

- `dropEffect`: 拖拽交互类型,通常决定浏览器如何显示鼠标光标并控制拖放操作.常见的取值有 `copy`, `move`, `link` 和 `none`
- `effectAllowed`: 指定允许的交互类型,可以取值: `copy`, `move`, `link`, `copyLink`, `copyMove`, `linkMove`, `all`, `none` 默认为 `uninitialized` (允许所有操作)
- `files`: 包含 `File` 对象的 `FileList` 对象.从操作系统向浏览器拖放文件时有用.
- `types`: 保存 `DataTransfer` 对象中设置的所有数据类型.
- `setData(format, data)`: 以键值对设置数据,format通常为数据格式,如 `text`, `text/html`
- `getData(format)`: 获取设置的对应格式数据,format与`setData()`中一致
- `clearData(format)`: 清除指定格式的数据
- `setDragImage(imgElement, x, y)`: 设置自定义图标

`dataTransfer` 对象在传递给监听器的事件对象中可以访问,如下:

```
draggableElement.addEventListener('dragstart', function (event) {  
  event.dataTransfer.setData('text', 'Hello World');  
}, false);
```

推荐的拖拽元素和数据类型

详细参考[MDN recommended drag type](#)

文本

在页面中选择文本并拖拽,无需处理 `dragstart` 设置数据,浏览器自动设置选取的文本.相当于

`event.dataTransfer.setData("text/plain", "this is text to drag")`.只需要在拖放目标上读取对应格式的数据即可.

链接

实际案例

前面介绍了最基本的理论知识,下面进行实际操作

元素拖拽

目标: 拖拽元素到达目的区域,改变在DOM中的位置,同时设置反馈视觉效果[在线demo](#)

```
<div id="demo1">
  <ul class="panel-list">
    <li class="panel-item"></li>
    <li class="panel-item"></li>
    <li class="panel-item"></li>
    <li class="panel-item"></li>
    <li class="panel-item"></li>
  </ul>
  <h2>拖拽下面的方块到上面任意容器中</h2>

  <!-- 设置draggable使元素成为可拖拽元素 -->
  <span class="movable" id="demo1-src" draggable="true"></span>

  <style>
    #demo1 {
      margin: 20px;
    }
    #demo1 .panel-list {
      overflow: hidden;
      list-style: none;
      margin: 0;
      padding: 0;
    }
    #demo1 .panel-item {
      float: left;
      margin-right: 30px;
      width: 100px;
      height: 100px;
      background: #ddd;
      border: 1px solid #ddd;
    }
    #demo1-src {
      display: inline-block;
      width: 50px;
      height: 50px;
      background: purple;
    }
    #demo1 .over {
      border: 1px dashed #000;
      -webkit-transform: scale(0.8, 0.8);
    }
  </style>
</div>
```

```
}
</style>
<script>
(function () {

    var dnd = {
        // 初始化
        init: function () {
            var me = this;
            me.src = document.querySelector('#demo1-src');
            me.panelList = document.querySelector('.panel-list');

            // 为拖拽源监听dragstart,设置关联数据
            me.src.addEventListener('dragstart', me.onDragStart, false);

            // 拖拽鼠标移入元素,在拖放目标上设置视觉反馈
            me.panelList.addEventListener('dragenter', me.onDragEnter, false);

            // 取消元素dragover默认行为,使其可拖放
            me.panelList.addEventListener('dragover', me.onDragOver, false);

            // 拖拽移出元素,清除视觉反馈
            me.panelList.addEventListener('dragleave', me.onDragLeave, false);

            // 鼠标释放,在拖放目标上接收数据并处理
            me.panelList.addEventListener('drop', me.onDrop, false);
        },
        onDragStart: function (e) {
            e.dataTransfer.setData('text/plain', 'demo1-src');
        },
        onDragEnter: function (e) {
            if (e.target.classList.contains('panel-item')) {
                e.target.classList.add('over');
            }
        },
        onDragLeave: function (e) {
            if (e.target.classList.contains('panel-item')) {
                e.target.classList.remove('over');
            }
        },
        onDragOver: function (e) {
            e.preventDefault();
        },
        onDrop: function (e) {
            var id = e.dataTransfer.getData('text/plain');
            var src = document.getElementById(id);
            var target = e.target;
            if (target.classList.contains('panel-item')) {
                target.appendChild(src);
                target.classList.remove('over');
            }
        }
    };
})();
```

```

    }

    };

    dnd.init();
  }());
</script>
</div>

```

从操作系统拖拽图片到指定区域进行预览

从操作系统拖拽文件到浏览器中.不会触发 `dragstart` , `dragend` ,只需取消拖放区域的默认行为,设置反馈,并在拖放发生时取消浏览器默认行为,通过 `e.dataTransfer.files` 获取文件信息进行操作.在线[demo](#)

```

<div id="demo2">
  <h3>从文件夹中拖拽图片到下面的区域进行预览</h3>
  <ul class="preview"></ul>
  <style>
    #demo2 {
      margin: 20px;
    }
    #demo2 .preview {
      height: 300px;
      background: #ddd;
    }
    #demo2 li {
      float: left;
      margin-left: 40px;
    }
    #demo2 img {
      max-height: 150px;
      width: auto;
    }
  </style>

  <script>
    (function (w) {
      var doc = w.document;

      var dnd = {
        init: function () {
          var me = this;
          var preview = doc.querySelector('#demo2 .preview');

          preview.addEventListener('dragover', function (e) {
            e.preventDefault();
          }, false);

```

```
preview.addEventListener('drop', function (e) {
    // 操作系统拖放文件到浏览器需要取消默认行为
    e.preventDefault();

    [].forEach.call(e.dataTransfer.files, function (file) {
        if (file && file.type.match('image.*')) {
            var reader = new FileReader();

            reader.onload = function (e) {
                var img = doc.createElement('img');
                img.src = e.target.result;
                var li = doc.createElement('li');
                li.appendChild(img);
                preview.appendChild(li);
            };

            reader.readAsDataURL(file);
        }
    });
}, false);
};
```

参考资料

- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Drag_and_drop
- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Drag_operation...
- <http://blog.teamtreehouse.com/implementing-native-drag-and-drop>
- <http://www.html5rocks.com/en/tutorials/dnd/basics/>

2015年05月31日发布 ...

赞 | 10

已收藏 | 65

你可能感兴趣的文章

[HTML5 DragEvent学习+制作一个可以拖动的DIV](#) 8 收藏, 425 浏览

[Qunee for HTML5 V2.5新版本发布](#) 1 收藏, 921 浏览

HTML5 实时监听输入框值变化 2 收藏, 347 浏览




本作品采用 署名-非商业性使用-禁止演绎 4.0 国际许可协议 进行许可。


1 条评论

默认排序

时间排序

 **phping** · 2017年02月22日


指定区域内不能拖拽移动呀。。。

 赞 +1 回复



文明社会，理性评论

发表评论

 **qiu_deqing**

820 声望

关注作者


发布于专栏

笔记

笔记,方便查找

4 人关注

关注专栏

产品	资源	商务	关于	关注	条款
热门问答	每周精选	人才服务	关于我们	产品技术日志	服务条款
热门专栏	用户排行榜	企业培训	加入我们	社区运营日志	内容许可
热门讲堂	徽章	活动策划	联系我们	市场运营日志	

最新活动
技术圈
找工作
移动客户端

帮助中心
声望与权限
社区服务中心
开发手册

广告投放
合作联系

团队日志
社区访谈



扫一扫下载 App

Copyright © 2011-2018 SegmentFault. 当前呈现版本 17.06.16
浙ICP备 15005796号-2 浙公网安备 33010602002000号 杭州堆栈科技有限公司版权所有

CDN 存储服务由 又拍云 赞助提供