拖拽献祭中的黑山羊-DataTransfer对象

这篇文章发布于 2018年09月30日，星期日，22:10，归类于 JS API。 阅读 8476 次, 今日 21 次 12 条评论

一、超位魔法-任意拖拽

伟大的安兹·乌尔·恭，纳萨里克地下大坟墓的主人，四十一位无上至尊的统合者，不死者之王，魔导国魔导王，异世界overlord，在这花好月圆之夜，释放了一个超位魔法——任意拖拽！

凡是不能手写页面元素拖拽效果者，即死，10日后才能复活！顿时，无数前端开发还没意识到发生了什么，眼前一黑，全部都倒下了。

“哈哈，实在是精彩，足足秒了7万零1人，破了我昨天的记录！”恭大人兴奋地大声疾呼。

然后，很快，天空出现巨大的黑球，然后5只巨大的黑山羊幼崽出现了。

大如小山，除了怪物已经无法用其他词形容了。

剩下的开发人员手足无措，惊恐万分，顿时仓皇而逃，但都难逃一死。不死者之王，果然是至高无上的存在！

二、解构魔法-DataTransfer对象

大家好，我是一个小白前端，因为昨天陪女朋友去现充世界赏月，没有参加这场战役，所以幸免于难。但现在还是心有余悸，因为自己也不会手写拖拽功能，如果自己在场，怕是女朋友就是别人的了。

虽庆幸，仍不安。要是下次伟大的安兹·乌尔·恭，不死者之王再释放这个“任意拖拽”超位魔法，那我岂不是死翘翘了？不行，为了活命，我必须对这个超位魔法进行解构，了解之学习之破解之。

皇天不负有心人，经过七七四十九天废寝忘食的研究，终于解构出了这个超位魔法的关键所在——DataTransfer对象，只要弄动DataTransfer对象，什么献祭中的黑山羊的幼崽都只是可爱的小羊而已，不足为惧。

为了让前台前端同僚也可以幸免于难，于是，我冒着被暴露的风险向大家传授DataTransfer对象相关知识。

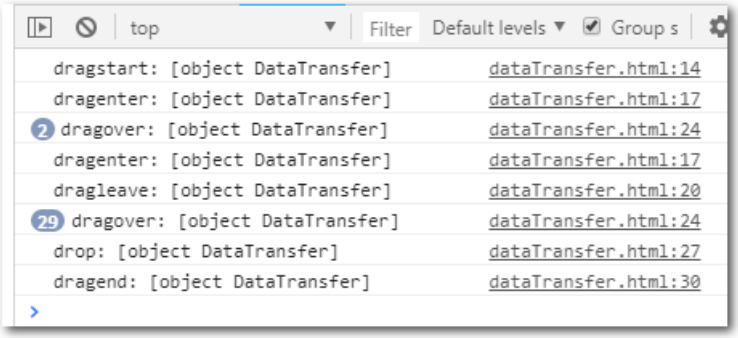
DataTransfer对象的出现

`DataTransfer`对象出现在拖拽事件中，具体包括开始拖拽`dragstart`事件，拖拽进入`dragenter`事件，拖拽离开`dragleave`事件，拖拽经过`dragover`事件，拖拽释放`drop`事件以及拖拽结束`dragend`事件。

示意代码如下：

```
document.addEventListener('dragstart', function (event) {
    console.log('dragstart: ' + event.dataTransfer);
});
document.addEventListener('dragenter', function (event) {
    console.log('dragenter: ' + event.dataTransfer);
});
document.addEventListener('dragleave', function (event) {
    console.log('dragleave: ' + event.dataTransfer);
});
document.addEventListener('dragover', function (event) {
    event.preventDefault();
    console.log('dragover: ' + event.dataTransfer);
});
document.addEventListener('drop', function (event) {
    console.log('drop: ' + event.dataTransfer);
});
document.addEventListener('dragend', function (event) {
    console.log('dragend: ' + event.dataTransfer);
});
```

假设包含上面JavaScript代码的页面有一张图片，拖动这张图片，就可以在控制台控制看到类似下面截图的输出：



若有兴趣亲自看看控制台输出内容，可以狠狠地点击这里：拖拽相关事件中的DataTransfer对象demo

这个DataTransfer对象包含诸多属性和方法，我们可以用来改变拖拽效果，获取和设置拖拽内容，熟练掌握，必定可以抵御献祭中的黑山羊魔法。

三、DataTransfer对象属性和方法详解

DataTransfer对象包含下面5个标准属性和4个标准方法。

标准属性

DataTransfer.dropEffect

获取当前所选拖放操作的类型，或将拖拽操作设置为新类型。值必须为`none`，`copy`，`link`或`move`中的一个。

DataTransfer.effectAllowed

提供可能的所有类型的操作。必须是`none`，`copy`，`copyLink`，`copyMove`，`link`，`linkMove`，`move`，`all`或`uninitialized`中的一个。

DataTransfer.files

拖拽的本地文件列表。如果拖动操作不涉及拖动文件，则此属性为空列表。

DataTransfer.items *（只读）*

提供DataTransferItemList对象，该对象是所有拖动数据的列表。

DataTransfer.types *（只读）*

在`dragstart`事件中设置数据格式，返回的是一个字符串数组。

标准方法

DataTransfer.clearData([format])

删除与给定类型关联的数据。`format`参数是可选的。如果类型为空或未指定，则删除所有关联的数据。如果指定类型的数据不存在，或者数据传输不包含任何数据，则此方法无效。

DataTransfer.getData(format)

返回给定类型的数据，如果该类型的数据不存在或数据传输不包含数据，则返回空字符串。

DataTransfer.setData(format, data)

设置给定类型的数据。如果该类型的数据不存在，则在末尾添加，以使列表中的最后一项成为新格式类型。如果该类型的数据已存在，则在相同位置把现有数据替换掉。

DataTransfer.setDragImage(img, xOffset, yOffset)

设置用于拖动的自定义图像。

下面一个一个详细展开。

四、DataTransfer.dropEffect

`dropEffect`属性顾名思意拖拽效果，在PC web端主要表现在鼠标手形上。不同的`dropEffect`值，鼠标的手形效果是不一样的。

举个例子，假设有个`box`元素，当`dragover`的时候，我们设置其`dropEffect`值分别为`move`，`copy`，`link`和`none`，如下代码示意：

```
// 拖拽元素经过box
box.addEventListener('dragover', function (event) {
    event.preventDefault();
    // 设置dropEffect值为move, copy, link, none
    event.dataTransfer.dropEffect = 'move';
});
```

则4个不同属性值效果如下如下截图示意：




可以看到，设置不同的`dropEffect`属性值，鼠标的手形就会有不一样的表现。

眼见为实，如果您是桌面浏览器浏览，则可以狠狠地点击这里：DataTransfer.dropEffect效果演示demo

选择不同的单选选项，然后拖动图片到该容器上方，即可看到不一样的鼠标手形表现，这就是`dropEffect`目前的表现。

其他信息

`dropEffect`属性的设置主要用在`dragenter`和`dragover`事件中，同时受`effectAllowed`属性影响。

例如，我们在`dragstart`的时候设置`effectAllowed`属性值为`'none'`，则`dropEffect`只能表现为`'none'`，而不会出现其他手形，即使设置了其他手形对应的属性值。

五、DataTransfer.effectAllowed

表示拖拽允许的效果。支持的属性值较多，如下：

uninitialized
: 默认值，表示未初始化。从效果上来讲，和`all`是一样滴。

none
: 不允许拖拽。鼠标保持禁止状态。

copy
: 可以在新位置复制元素。

copyLink
: 允许复制和链接操作。

copyMove
: 允许复制和移动操作。

link
: 可以在新位置建立链接。

linkMove
: 允许链接和移动操作。

move
: 可以把元素移动到新位置。

all
: 什么操作都允许。

如果`effectAllowed`属性值设置为上面列表以外的其它值则没有任何效果。另外在IE浏览器下所有的字都会小写，因此类似`linkMove`会变成`linkmove`。

effectAllowed和dropEffect的关系

`effectAllowed`和`dropEffect`通常应用的事件方法名不一样，`effectAllowed`多用在`dragstart`事件中，而`dropEffect`属性的设置主要用在`dragenter`和`dragover`事件中。

`effectAllowed`和`dropEffect`的彼此间是有制约关系，当我们给`effectAllowed`设置了对应的属性值，则`dropEffect`只能设置为`effectAllowed`允许的值，否则是无效的。

举个例子，如果我们设置`effectAllowed`值为`'copyMove'`，则`dropEffect`只有`'copy'`和`'move'`这两个属性值才有效。

`effectAllowed`看上去很屌，但实际应用的时候相当鸡肋。我们通常的拖拽应用是用不到这个的。只要下面这个场景，那就是当我们有很多个元素需要拖拽，但是需要像垃圾一样分门别类的时候，`effectAllowed`就有用了。

原生拖拽事件有这样一种行为，那就是如果`effectAllowed`值和`dropEffect`值不一致，则无法响应`drop`事件。我们可以想象一下，假设我们在网页中放三个垃圾箱，分别回收move元素，copy元素和link元素，由于DOM元素的转移或者复制我们都是在`drop`事件中完成的，则`effectAllowed`包含`co`

`py` 元素的才能拖拽进入copy垃圾箱（可以触发 `drop` 事件）。

平常开发都是简单的拖拽，哪里需要用到分门别类啊，因此 `effectAllowed` 也就无用武之地了。

//zxx: 这里有个CodePen，可以感受下。

link作为属性值案例一则

这里插播一个案例，实现的是Chrome浏览器下拖拽链接也能新窗口打开的实现。在Firefox和IE浏览器下，链接元素你按住一拖拽，就可以在新的浏览器标签页中打开，Chrome浏览器则需要拖动到地址栏才可以。如果我们想要实现Chrome浏览器下拖拽也能标签页中打开，就可以看看下面这个案例。

您可以狠狠地点击这里：拖拽链接新标签页打开demo

拖拽链接到demo页面一个灰色盒子中，如下图：

释放鼠标，此时这个链接就会在新标签页中打开。

相关代码如下：

```
<a href="#" id="link">拖拽我到下面框框试试</a>
<p id="box" class="box"></p>
```

```
var isOpenLink = null;
link.addEventListener('dragstart', function (event) {
    event.dataTransfer.effectAllowed = 'link';
});
box.addEventListener('dragover', function (event) {
    event.preventDefault();
    // 检测是否需要新窗口打开链接
    if (event.dataTransfer.dropEffect != 'link') {
        isOpenLink = true;
    }
    event.dataTransfer.dropEffect = 'link';
});
box.addEventListener('drop', function (event) {
    event.preventDefault();
    if (isOpenLink) {
        window.open(link.href);
    }
});
```

通过检测 `dataTransfer.dropEffect`，有效避免和原本支持拖拽新标签页打开链接的浏览器冲突。

六、DataTransfer.files

当我们从桌面往网页浏览器中拖文件的时候，DataTransfer.files就派上用场了，其对应的列表只就是我们拖进去的文件列表。

目前在实际开发中应用的比较多的是拖拽上传，具体可参见我11年写的一个ajax图片上传demo。

这里我又写了一个更精简的demo，演示如何使用DataTransfer.files获得桌面拖拽进入的文件信息。

您可以狠狠的点击这里：DataTransfer.files获取桌面文件信息demo

例如，我框选两个文本文件和两个快捷方式，拖到demo页面的框框里面，结果效果如下图：

可以看到，两个txt文件识别为了text/plain，而快捷方式文件没有mime-type，因此，输出内容是空。

相关JS代码如下：

```
box.addEventListener('drop', function (event) {
    event.preventDefault();
    // 遍历文件信息
    var files = event.dataTransfer.files || [];
    var length = files.length;
    if (length == 0) {
        this.innerHTML = '<p>没有文件</p>';
        return;
    }
    var html = '';
    for (var index = 0; index < length; index++) {
        html += '<p>类型: '+ files[index].type +'</p>';
    }
    this.innerHTML = html;
});
```

什么时候会出现“没有文件”的提示呢？比方说demo页面你框选几个文字，拖拽到方框框里面，就会提示“没有文件”了，因为你拖拽进去的本来就不是文件内容。不过，这个拖拽内容我们是可以使用DataTransfer.items获取的。

七、DataTransfer.items

DataTransfer.items可以用来获取拖拽的数据信息，只读。

DataTransfer.items为DataTransferItem类型的数据列表集合，而DataTransferItem又包含多个属性和方法，属性包括 `kind` 和 `type` ，方法包括 `getAsString()` 和 `getAsFile()` ，这个和剪切板item对象方法是一致的。

我们可以通过一个小案例快速了解一下这些属性含义和作用。

您可以狠狠地点击这里：DataTransferItem属性和方法信息输出demo

log信息打印测试代码如下：

```
console.log('kind: ' + DataTransferItem.kind + ', type: ' + DataTransferItem.type + '\n');
DataTransferItem.getAsString(function (str) {
    console.log('\ngetAsString: ' + str);
});
```

主要测试了 `kind` 和 `type` 属性和 `getAsString()` 方法。

拖动图片，此时就可以得到DataTransfer.items具体都是些什么信息，如下截图：

其中，包含3个item， `kind` 全部都是 `string` ， `type` 指的是mimeType类型，包含3中不同类型，

为：`text/plain`，`text/html`和`text/uri-list`。

如果我们拖动页面上文字信息到输入框上显示的就只有2中类型项：

如果从浏览器地址栏拖文字到输入框，就会只有`text/plain`这1种类型。

//zxx: 浏览器不同的输入容器会自动甄别显示内容，例如：，对于文本框，显示内容为text/plain纯文本（如本demo），如果是富文本输入框（如HTML元素设置contenteditable属性），则显示内容为text/html富文本。

实际上，子item的`kind`还有可能是`'file'`，例如把微博个人主页头像拖进来：

此时，可以调用`getAsFile()`方法将其转换成二进制文件对象，然后可以ajax上传等处理。

实际开发的处理模板

有个处理模板，无论何种数据类型，大家可以找到对应位置，进行相应的处理，省掉不少自己写逻辑判断的麻烦。

```
handleDataTransferItems = function (items) {
  for (var i = 0; i < items.length; i += 1) {
    var kind = items[i].kind;
    var type = items[i].type;
    // 逻辑开始
    if (kind == 'string') {
      if (type.indexOf('text/plain') != -1) {
        items[i].getAsString(function (str) {
          // str是纯文本，该怎么处理，就在这里处理
        });
      } else if (type.indexOf('text/html') != -1) {
        items[i].getAsString(function (str) {
          // str是富文本，就在这里处理
        });
      } else if (type.indexOf('text/uri-list') != -1) {
        items[i].getAsString(function (str) {
          // str是uri地址，在这里进行处理
        });
      }
    } else if (kind == 'file') {
      // 如果是图片
      if (type.indexOf('image/') != -1) {
        var file = items[i].getAsFile();
        // file就是图片文件对象，可以上传，或者其他处理
      }
    }
  }
};
```

使用示例：

```
document.addEventListener('drop', function (event) {
    var items = event.dataTransfer.items || [];
    handleDataTransferItems(items);
```

```
});
```

近亲clipboardData.items

剪切板对象也有items属性，数据类型以及item子项的属性和方法和DataTransfer对象是一样的，学会了一个自然也就学会了另外一个，不展开。

八、DataTransfer.types

DataTransfer.types用在`dragstart`事件中，包含拖拽内容的包含的mimeType类型们，可以遍历出具体的`type`类型。假设页面有如下全局JS代码：

```
document.addEventListener('dragstart', function (event) {
    // 遍历并输出拖拽内容的类型
    var types = event.dataTransfer.types || [];
    [].slice.call(types).forEach(function (type) {
        console.log(type);
    });
});
```

则拖动纯图片元素输出内容如下：

```
text/uri-list
text/html
Files
```

拖动文字如下：

```
text/html
text/plain
```

眼见为实，您可以狠狠地点击这里：DataTransfer.types使用测试demo

DataTransfer.types有什么用呢？如果我们希望页面某些元素可以拖拽，有些不允许，则可以使用types属性进行检测，例如，如果是包含Files，则执行`event.preventDefault()`，拖拽行为就会被禁止。

九、DataTransfer.getData()

DataTransfer.getData(format)可以快捷获取拖拽的内容。

`format`可以理解为就是DataTransfer.items中的type值，例如，我们想要获取拖拽内容的富文本格式，可以：

```
document.addEventListener('drop', function (event) {
    // 获取拖拽富文本内容
    var html = event.dataTransfer.getData('text/html');
});
```

如果是获取纯文本，可以`text/plain`，或者有时候`text/uri-list`，实际开发时候，我们会直接使用`'text'`，例如：

```
document.addEventListener('drop', function (event) {
    // 获取拖拽纯文本内容
    var html = event.dataTransfer.getData('text');
});
```

优化输入框拖拽输入体验

本案例类似于“利用剪切板JS API优化输入框的粘贴体验”这篇文章，当我们拖拽本文进入输入框的时候，文本信息可能包含不需要的信息，例如手机号中的334分隔符，借助`getData()`方法我们可以进行过滤和优化。

您可以狠狠地点击这里：DataTransfer.getData()优化手机号拖拽输入体验demo

可以看到（如下GIF），手机号明明是132-0803-3621，但拖到输入框后自动变成了13208033621，省掉了再编辑的烦恼：

相关JavaScript代码如下：

```
input.addEventListener('drop', function (event) {
    // 获取拖拽文本内容
    var text = event.dataTransfer.getData('text');
    if (this.value == '' && text.match(/\d/g) && text.match(/\d/g).length == 11) {
        event.preventDefault();
        input.value = text.replace(/\D/g, '');
        input.select();
    }
});
```

十、DataTransfer.setData()

DataTransfer.setData(format, data)可以自定义拖拽的内容信息。可以重置原生的拖拽内容，或者用来参数传递。

例如如果我们页面运行了如下JS代码：

```
document.addEventListener('dragstart', function (event) {
    // 重置拖拽文本内容
    event.dataTransfer.setData('text', '张鑫旭-鑫空间-鑫生活');
});
```

则所有内容拖拽进入输入框内容都是“张鑫旭-鑫空间-鑫生活”，如下GIF示意：

您可以狠狠地点击这里：DataTransfer.setData()重置拖拽文本内容demo

我们实际开发基本上都是拖拽具体模块元素，此时`setData()`方法多用来传递拖拽元素的`id`。

十一、DataTransfer.clearData()

DataTransfer.clearData()只能用在`dragstart`事件中，会清除所有的数据。也就是执行了此方法后，DataTransfer.getData()方法只能获得空字符串。

使用场景不是很多。

当我们需要使用 `setData()` 方法自定义拖拽数据的时候，为了避免原生拖拽数据干扰，可以先执行一次 `clearData()` 方法。这样可以避免 `text/html` 类型数据干扰（如果我们自定义的是纯文本数据）。

十二、DataTransfer.setDragImage()

DataTransfer.setDragImage(img, offsetX, offsetY)用在 `dragstart` 事件中，可以设置拖拽时候有个图片跟在后面。

其中， `img` 表示图片DOM元素对象， `offsetX` 表示距离鼠标的水平偏移距离， `offsetY` 表示距离鼠标的垂直偏移距离。

直接看一个例子：

```
var image = new Image();
image.src = './1f603.svg';
document.addEventListener('dragstart', function (event) {
    // 设置拖拽图片
    event.dataTransfer.setDragImage(image, 10, 10);
});
```

此时，页面上拖拽任意内容，就会看到一个笑脸跟在后面了，例如：

您可以狠狠地点击这里：DataTransfer.setDragImage()设置拖动跟随图片demo

十三、最后的献祭魔法

果然是超位魔法，解构下来洋洋洒洒这么多内容，虽然学得辛苦，但是若能保命，拿一切都值得的。

国庆假期到来，于是回乡去探亲，没想到好死不活，遇到伟大的安兹·乌尔·恭正好对家乡的前端再次发动了超位魔法——拖拽献祭中的黑山羊，无数前端开发沉浸在假期的愉悦之中，根本没有意识到危险的到来，似乎过去的梦魇又将重新。我虽然是个小白前端，但是为了整个行业的兴盛不衰，我毅然站在了魔法的最前面，抵御魔法的攻击。

这攻势凶猛异常，我忙拿出分析解构DataTransfer对象学到的知识作为武器抵御，哇哦哦哦，身体开始剧痛，剧烈的魔法风暴从身体擦过，想刀片一样，双脚开始站立不稳，意识开始模糊，我连忙回想没日没月解构DataTransfer对象的经历，是不是遗漏了什么细节，Bing地一声，我意识到了什么，身体中间出现一道金光，护住的最重要的心脉，原来是由此及彼，解构DataTransfer联想到了其他对象，让自己同时学会了其他魔法，相互融合，威力大增，虽然依旧遍体鳞伤，但是主要心脉护住了，就不会有生命危险，只要顶住即可！

啊啊啊啊，输出全靠吼，我用尽力气，“解构全开，魔法破！！！”

轰隆一声巨响，瞬间眼前的黑暗烟消云散，我实在只撑不住，单腿跪地，大口喘气！

“你是什么人？居然能够借助我的超位祭祀魔法！”

“我……我……就是一个……”我咽了下喉咙里的血水，继续说道：“一个普通的小白前端……”

“搜噶！你叫什么名字？”

“我的名字不值得安兹大人挂齿……”

“莫非你要拒绝回答我的问题？！！”

“不敢不敢！我的名字是————————————空气！”

“好！空气，我记住你了！我今天放你一条生路，你好好提高你的魔法技能，等以后足够强大了，我特许你可以和我一战！哈哈哈哈！”

“好……多谢安兹大人！”

说完，安兹大人就通过时空门离开了。

而我“空气”也从此成了安兹大人最重视的敌人！

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。

« 直接剪切板粘贴上传图片的前端JS实现 搞懂SVG/Canvas中nonzero和evenodd填充规则 »

猜你喜欢

- 利用剪切板JS API优化输入框的粘贴体验
- 基于HTML5 drag/drop模块拖动插入排序删除完整实例
- 直接剪切板粘贴上传图片的前端JS实现
- HTML5 drag & drop 拖拽与拖放简介
- 小tips: 纯前端JS读取与解析本地文本类文件
- 微信网页悬浮窗交互效果的web实现
- JavaScript实现最简单的拖拽效果
- HTML5终极备忘大全（图片版+文字版）
- 有意思：textarea resize属性下纯CSS交互效果
- 开源移动端元素拖拽惯性弹动以及下拉加载两个JS

标签： DataTransfer, dragend, dragenter, dragleave, dragover, dragstart, drop, dropEffect, effectAllowed, getAsFile, getAsString, getData, setData, setDragImage, 拖拽

发表评论（目前12条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. **blossom说道：**
2019年01月8日 11:50

哈哈哈~~了

回复

2. **千寻而见说道：**
2018年12月18日 17:25

拖拽这个非常有用

回复

3. **嘻嘻说道：**
2018年12月18日 17:12

超链接下的波浪线一直动是怎么做的啊

回复

4. **骨王麾下小迪说道：**
2018年11月8日 22:50

吼吼 看在你是吾王欣赏的人 是否考虑加入大坟墓

回复

5. **大屌萌妹说道：**
2018年10月25日 11:08

献祭中的黑山羊幼崽 懂的人自然懂

回复

6. **meepo说道：**
2018年10月9日 11:37

一股中二气息。哈哈

回复

7. **广建说道：**
2018年10月9日 10:40

整理的很详细，棒棒的

回复

8. **nicholasurey说道：**
2018年10月8日 17:44

换新妹纸了？

回复

> 张 鑫旭说道：
> 2018年10月9日 10:17
>
> 换了，介绍下，名叫陈晨玲，也可以叫她CC0。
>
> 回复

9. 魏义齐说道：
2018年10月2日 10:55

提交评论的按钮效果怎么做的？

回复

> 张 鑫旭说道：
> 2018年10月2日 13:47
>
> F12
>
> 回复

> 爱吃鱼说道：
> 2018年10月21日 16:24
>
> 可以看看这一章 CSS3 box-shadow盒阴影图形生成技术 http://www.zhangxinxu.com/wordpress/?p=3813
>
> 回复

最新文章

» 常见的CSS图形绘制合集
» 粉丝群第1期CSS小测点评与答疑
» 分享三个纯CSS实现26个英文字母的案例
» 小tips: 纯CSS实现打字动画效果
» CSS/CSS3 box-decoration-break属性简介
» CSS :placeholder-shown伪类实现Material Design占位符交互效果
» 从天猫某活动视频不必要的3次请求说起
» CSS vector-effect与SVG stroke描边缩放
» CSS ::backdrop伪元素是干嘛用的?
» 周知：CSS -webkit-伪元素选择器不再导致整行无效

今日热门

» 常见的CSS图形绘制合集 (178)
» 粉丝群第1期CSS小测点评与答疑 (112)
» 未来必热：SVG Sprite技术介绍 (111)
» HTML5终极备忘大全（图片版+文字版） (85)
» 让所有浏览器支持HTML5 video视频标签 (83)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (80)
» CSS3下的147个颜色名称及对应颜色值 (78)
» 小tips: 纯CSS实现打字动画效果 (72)
» 写给自己看的display: flex布局教程 (69)
» 分享三个纯CSS实现26个英文字母的案例 (69)

今年热议

» 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)
» 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
» 看，for..in和for..of在那里吵架！ (60)

» 是时候好好安利下LuLu UI框架了！ (47)
» 原来浏览器原生支持JS Base64编码解码 (35)
» 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
» 炫酷H5中序列图片视频化播放的高性能实现 (31)
» CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
» windows系统下批量删除OS X系统.DS_Store文件 (26)
» 写给自己看的display: flex布局教程 (26)

猜你喜欢

- 利用剪切板JS API优化输入框的粘贴体验
- 基于HTML5 drag/drop模块拖动插入排序删除完整实例
- 直接剪切板粘贴上传图片的前端JS实现
- HTML5 drag & drop 拖拽与拖放简介
- 小tips: 纯前端JS读取与解析本地文本类文件
- 微信网页悬浮窗交互效果的web实现
- JavaScript实现最简单的拖拽效果
- HTML5终极备忘大全（图片版+文字版）
- 有意思：textarea resize属性下纯CSS交互效果
- 开源移动端元素拖拽惯性弹动以及下拉加载两个JS