Drag and Drop | HTML5

arkaitzgarro.com/html5/capitulo-10.html

Capítulo 10 Drag and Drop

Durante años, hemos utilizado bibliotecas como *jQuery* y *Dojo* para conseguir funcionalidades complejas en las *interfaces* de usuario como las animaciones, las esquinas redondeadas y la función de arrastrar y soltar. Esta última funcionalidad (*Drag and Drop, DnD*) tiene una gran importancia en HTML5, y de hecho se ha integrado en el API. En la especificación, este API se define como un mecanismo basado en eventos, donde identificamos los elementos que deseamos arrastrar con el atributo `draggable` y desde JavaScript escuchamos los eventos que se producen, para proporcionar la funcionalidad deseada.

Por defecto, todos los enlaces, imágenes y nodos de texto (o selecciones de texto) de un documento HTML son arrastables, pero no hay ningún evento asociado a estas acciones, por lo que poco más podemos hacer, a excepción de la funcionalidad que nos ofrezca el navegador o el propio sistema operativo (guardar las imágenes en el escritorio, crear ficheros de texto, etc).

10.1 Detección de la funcionalidad

Aunque la compatibilidad actual de los navegadores con esta API es bastante amplia, hay que tener en cuenta que los dispositivos móviles no soportan esta funcionalidad. Si nuestro sitio web está siendo accedido desde un dispositivo móvil, y tenemos implementada esta funcionalidad (por ejemplo en una cesta de la compra), debemos proveer otra solución para que nuestro sitio web se comporte de manera correcta, y no perjudicar la experiencia del usuario.

La manera mas sencilla de comprobar la disponibilidad de este API, es utilizar la biblioteca *Modernizr*, la cual nos indica si el navegador soporta esta funcionalidad:

```
if (Modernizr.draganddrop) {
    // Browser supports HTML5 DnD.
} else {
    // Fallback to a library solution or disable DnD.
}
```

10.2 Creación de contenido *arrastrable*

Hacer que un elemento se pueda arrastrar es muy sencillo. Solo hay que establecer el atributo `draggable="true"` en el elemento que se quiere mover. La función de arrastre se puede habilitar prácticamente en cualquier elemento, incluidos archivos, imágenes, enlaces, listas u otros nodos DOM.

```
<div id="columns">
    <div class="column" draggable="true"><header>A</header></div>
    <div class="column" draggable="true"><header>B</header></div>
    <div class="column" draggable="true"><header>C</header></div>
</div>
```

Una *ayuda visual* al usuario, para indicar que un elemento es arrastable, es transformar el aspecto tanto del elemento como del cursor. Con CSS esto es muy sencillo:

```
[draggable] {
    user-select: none;
}

.column:hover {
    border: 2px dotted #666666;
    background-color: #ccc;
    border-radius: 10px;
    box-shadow: inset 0 0 3px #000;
    cursor: move;
}
```

10.3 Eventos de arrastre

La especificación define hasta siete eventos que son lanzados tanto por los elementos de origen (los que son arrastrados) com para los elementos de destino (donde *soltamos* el elemento arrastrado). Son los siguientes:

- `dragstart` : comienza el arrastrado. El `target` del evento hace referencia al elemento que está siendo arrastrado.
- `drag` : el elemento se ha movido. El `target` del evento hace referencia al elemento que está siendo arrastrado. Este evento se dispara tantas veces como se mueva el elemento.
- `dragenter` : se dispara cuando un elemento que está siendo arrastrado entra en un contenedor. El `target` del evento hace referencia al elemento contenedor.
- `dragleave` : el elemento arrastrado ha salido del contenedor. El `target` del evento hace referencia al elemento contenedor.
- `dragover` : el elemento se ha movido dentro del contenedor. El `target` del evento hace referencia al elemento contenedor. Como el comportamiento por defecto es denegar el `drop` , la función debe retornar el valor `false` o llamar al método `preventDefault` del evento para que indicar que se puede el soltar elemento.
- `drop` : el elemento arrastrado has sido exitosamente soltado en el elemento contenedor. El `target` del evento hace referencia al elemento contenedor.
- `dragend` : se ha dejado de arrastrar el elemento, con éxito o no. El `target` del evento hace referencia al elemento arrastrado.

Para organizar el flujo de *DnD*, necesitamos un elemento de origen (en el que se origina el movimiento de arrastre), la carga de datos (la información que va asociada al elemento arrastrado) y un elemento de destino (el área en la que se soltarán los datos). El elemento

de origen puede ser una imagen, una lista, un enlace, un objeto de archivo, un bloque de HTML o cualquier otro elemento, al igual que la zona de destino.

10.3.1 Comienzo de la operación de arrastre

Una vez que se haya definido el atributo `draggable="true"` en los elementos que queremos convertir en arrastables, debemos añadir los escuchadores necesarios para reaccionar antes los eventos que se lanzan desde los elementos. El primer evento de los que define la especificación se produce cuando un elemento comienza a ser arrastrado. Un ejemplo muy sencillo en el que cambiamos la transparencia de un elemento al comenzar a ser arrastrado:

```
function handleDragStart(e) {
    this.style.opacity = '0.4';
}

var cols = document.querySelectorAll('#columns .column');
[].forEach.call(cols, function(col) {
    col.addEventListener('dragstart', handleDragStart, false);
});
```

No debemos olvidarnos de volver a fijar la transparencia del elemento en el 100% una vez completada la operación de arrastre, ya que de otro modo, seguiría siendo transparente una vez finalizado la operación de arrastre.

10.3.2 `dragenter`, `dragover` y `dragleave`

Los controladores de eventos `dragenter`, `dragover` y `dragleave` se pueden utilizar para proporcionar pistas visuales adicionales durante el proceso de arrastre. Por ejemplo, el borde de un elemento de destino se puede convertir en una línea discontinua al realizar una operación de arrastre. De esa forma, los usuarios podrán distinguir cuáles son los elementos de destino donde pueden soltar lo que están arrastrando.

```
.column.over {
    border: 2px dashed #000;
}
```

```
function handleDragStart(e) {
    this.style.opacity = '0.4';
}

function handleDragOver(e) {
    if (e.preventDefault) {
        e.preventDefault(); // Necessary. Allows us to drop.
    }
    e.dataTransfer.dropEffect = 'move';
    return false;
}

function handleDragEnter(e) {
    // this / e.target is the current hover target.
    this.classList.add('over');
}

function handleDragLeave(e) {
    this.classList.remove('over');  // this / e.target is previous target element.
}

var cols = document.querySelectorAll('#columns .column');
[].forEach.call(cols, function(col) {
    col.addEventListener('dragstart', handleDragStart, false);
    col.addEventListener('dragenter', handleDragEnter, false);
    col.addEventListener('dragover', handleDragOver, false);
    col.addEventListener('dragleave', handleDragLeave, false);
});
```

Algunas consideraciones en el código anterior. Cuando *algo* es soltado sobre un elemento, se dispara el evento `dragover`, y tenemos la posibilidad de leer el objeto `event.dataTransfer`. Este objeto contiene los datos asociados a las operaciones de arrastre, como veremos más adelante. Tal y como hemos comentado anteriormente, cuando soltamos un elemento, hay que impedir el comportamiento predeterminado del navegador, que es denegar el `drop`.

10.3.3 Finalización de la operación de arrastre

Para que se procese la operación de soltar, se debe añadir un escuchador para los eventos `drop` y `dragend`. En este controlador, habrá que impedir el comportamiento predeterminado del navegador para este tipo de operaciones, que puede ser abrir un enlace o mostrar una imagen. Para ello, evitamos la propagación del evento con `e.stopPropagation()`.

En nuestro ejemplo utilizaremos `dragend` para eliminar la clase `over` de cada columna:

```
function handleDrop(e) {
    // this / e.target is current target element.
    if (e.stopPropagation) {
        e.stopPropagation(); // stops the browser from redirecting.
    }
    // See the section on the DataTransfer object.
    return false;
}

function handleDragEnd(e) {
    // this/e.target is the source node.
    [].forEach.call(cols, function (col) {
        col.classList.remove('over');
    });
}

var cols = document.querySelectorAll('#columns .column');
[].forEach.call(cols, function(col) {
    col.addEventListener('dragstart', handleDragStart, false);
    col.addEventListener('dragenter', handleDragEnter, false);
    col.addEventListener('dragover', handleDragOver, false);
    col.addEventListener('dragleave', handleDragLeave, false);
    col.addEventListener('drop', handleDrop, false);
    col.addEventListener('dragend', handleDragEnd, false);
});
```

10.3.4 El objeto datatransfer

La propiedad `dataTransfer` es el centro de desarrollo de toda la actividad de la función *DnD*, ya que contiene los datos que se envían en la acción de arrastre. La propiedad `dataTransfer` se establece en el evento `dragstart` y se lee/procesa en el evento `drop`. Al activar `e.dataTransfer.setData(format, data)`, se establece el contenido del objeto en el tipo MIME y se transmite la carga de datos en forma de argumentos.

En nuestro ejemplo, la carga de datos se ha establecido en el propio HTML del elemento de origen:

```
var dragSrcEl = null;
function handleDragStart(e) {
    this.style.opacity = '0.4';
    dragSrcEl = this;
    e.dataTransfer.effectAllowed = 'move';
    e.dataTransfer.setData('text/html', this.innerHTML);
}
```

`dataTransfer` también tiene el formato `getData` necesario para la extracción de los datos de arrastre por tipo de MIME. A continuación se indica la modificación necesaria para el procesamiento de la acción de arrastre de un elemento:

```
function handleDrop(e) {
    // this/e.target is current target element.
    if (e.stopPropagation) {
            e.stopPropagation(); // Stops some browsers from redirecting.
    }
    // Don't do anything if dropping the same column we're dragging.
    if (dragSrcEl != this) {
            // Set the source column's HTML to the HTML of the columnwe dropped
on.
            dragSrcEl.innerHTML = this.innerHTML;
            this.innerHTML = e.dataTransfer.getData('text/html');
    }
    return false;
}
```

Hemos añadido una variable global llamada `dragSrcEl` para facilitar el cambio de posición de la columna. En `handleDragStart()`, la propiedad `innerHTML` de la columna de origen se almacena en esa variable y, posteriormente, se lee en `handleDrop()` para cambiar el HTML de las columnas de origen y destino.

10.4 Arrastre de archivos

Las API de *DnD* permiten arrastrar archivos del escritorio a una aplicación web en la ventana del navegador.

Para arrastrar un archivo desde el escritorio, se deben utilizar los eventos de *DnD* del mismo modo que otros tipos de contenido. La diferencia principal se encuentra en el controlador drop. En lugar de utilizar `dataTransfer.getData()` para acceder a los archivos, sus datos se encuentran en la propiedad `dataTransfer.files`:

```
function handleDrop(e) {
    e.stopPropagation();
    // Stops some browsers from redirecting.
    e.preventDefault();
    var files = e.dataTransfer.files;
    for (var i = 0, f; f = files[i]; i++) {
        // Read the File objects in this FileList.
    }
}
```