Centro de Tecnología y Artes Visuales

Proyecto final de JavaScript.

JavaScript 2

Nombre del proyecto: Conecta tuberías

API: drag and drop

Integrantes:

Lilliam Montoya

Saúl López

Documentación

Elementos básicos de arrastrar y soltar

¿Que es?

Es la propiedad de html5 que te permite agarrar un objeto y cambiarlo de ubicación.

¿Cuando utilizarlo?

Cuando queramos crear una aplicación interactiva que nos permita mover elementos de un lugar a otro y/o arrastrar archivos desde el ordenador al browser.

Creación de contenido arrastrable

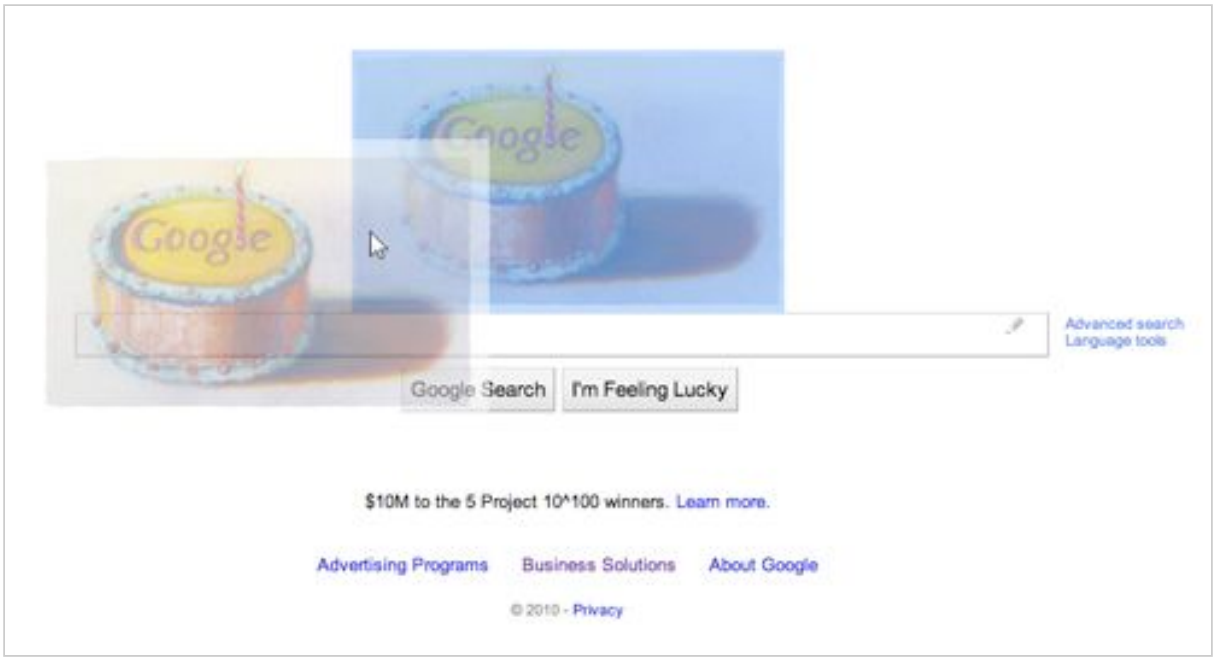
Hacer que un objeto se pueda arrastrar es muy sencillo. Solo hay que establecer el atributo `draggable=true` en el elemento que se quiere mover. La función de arrastre se puede habilitar prácticamente en cualquier elemento, incluidos archivos, imágenes, enlaces u otros nodos DOM.

Para ver un ejemplo, vamos a empezar creando columnas reorganizables. El marcado básico puede ser así:

```
<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>
```

Cabe destacar que, en la mayoría de los navegadores, los elementos de anclaje, los elementos de imágenes y las selecciones de texto que tienen un atributo `href` se pueden arrastrar de forma predeterminada. Por

ejemplo, al arrastrar el logotipo de google.com, se genera una imagen fantasma:



La mayoría de los navegadores admiten la función de arrastrar imágenes de forma predeterminada.

Esa imagen se puede soltar en la barra de direcciones, en un elemento `<input type="file" />` e, incluso, en el escritorio. Si quieres habilitar la función de arrastre en otros tipos de contenido, tendrás que utilizar las API de DnD de HTML5.

Con un pequeño toque mágico de CSS3, podemos retocar el marcado para obtener columnas. Si añadimos `cursor: move`, los usuarios recibirán una señal visual de que un elemento se puede mover:

```
<style>
/* Prevent the text contents of draggable elements from being
selectable. */
[draggable] {
  -moz-user-select: none;
  -khtml-user-select: none;
  -webkit-user-select: none;
  user-select: none;
  /* Required to make elements draggable in old WebKit */
  -khtml-user-drag: element;
  -webkit-user-drag: element;
}
.column {
  height: 150px;
  width: 150px;
  float: left;
  border: 2px solid #666666;
  background-color: #ccc;
  margin-right: 5px;
  -webkit-border-radius: 10px;
  -ms-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
  -webkit-box-shadow: inset 0 0 3px #000;
  -ms-box-shadow: inset 0 0 3px #000;
  box-shadow: inset 0 0 3px #000;
  text-align: center;
  cursor: move;
}
.column header {
  color: #fff;
  text-shadow: #000 0 1px;
  box-shadow: 5px;
  padding: 5px;
  background: -moz-linear-gradient(left center, rgb(0,0,0),
rgb(79,79,79), rgb(21,21,21));
  background: -webkit-gradient(linear, left top, right top,
                               color-stop(0, rgb(0,0,0)),
                               color-stop(0.50, rgb(79,79,79)),
                               color-stop(1, rgb(21,21,21)));
  background: -webkit-linear-gradient(left center, rgb(0,0,0),
rgb(79,79,79), rgb(21,21,21));
  background: -ms-linear-gradient(left center, rgb(0,0,0),
rgb(79,79,79), rgb(21,21,21));
  border-bottom: 1px solid #ddd;
  -webkit-border-top-left-radius: 10px;
  -moz-border-radius-topleft: 10px;
  -ms-border-radius-topleft: 10px;
  border-top-left-radius: 10px;
  -webkit-border-top-right-radius: 10px;
  -ms-border-top-right-radius: 10px;
  -moz-border-radius-topright: 10px;
  border-top-right-radius: 10px;
}
</style>
```

Resultado (se puede arrastrar, pero no hace nada):

Con el ejemplo anterior, la mayoría de los navegadores crean una imagen fantasma del contenido que se arrastra. Otros (concretamente, Firefox) requieren el envío de algunos datos en la operación de arrastre. En la siguiente sección, haremos que nuestro ejemplo de las columnas empiece a ponerse interesante añadiendo detectores para el procesamiento del modelo de eventos de arrastrar/soltar.

Detección de eventos de arrastre

Se deben tener en cuenta distintos eventos para controlar todo el proceso de arrastrar y soltar:

`dragstart`	**`drag`**
`dragenter`	**`dragover`**
`drop`	**`dragend`**
`dragleave`	

Para organizar el flujo de DnD, necesitamos un elemento de origen (en el que se origina el movimiento de arrastre), la carga de datos (lo que queremos soltar) 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 otra cosa. El elemento de destino es la zona de arrastre (o un conjunto de zonas de

arrastre) donde se aceptan los datos que el usuario intenta soltar. Ten en cuenta que no todos los elementos pueden ser elementos de destino (por ejemplo, las imágenes).

1. Comienzo de la operación de arrastre

Una vez que hayas definido atributos **`draggable="true"`** en tu contenido, incluye controladores de eventos **`dragstart`** para poner en marcha la secuencia de DnD de cada columna.

Este código hará que la opacidad de la columna pase a un 40% cuando el usuario empiece a arrastrarla:

```
function handleDragStart(e) {
  this.style.opacity = '0.4';  // this / e.target is the source node.
}

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

Resultado:

Como el elemento de destino del evento `dragstart` es nuestro elemento de origen, si se establece `this.style.opacity` en un 40%, el usuario notará que el elemento corresponde al objeto seleccionado que se está moviendo. No debemos olvidarnos de volver a fijar la opacidad de la columna en el 100% una vez completada la operación de arrastre. Un lugar bastante obvio para indicarlo es el evento `dragend`. Volveremos sobre este punto más adelante.

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 una columna se puede convertir en una línea discontinua al colocar el cursor sobre la columna durante una operación de arrastre. De esa forma, los usuarios sabrán que las columnas también son elementos de destino en los que se pueden soltar los objetos arrastrados.

```
<style>
.column.over {
  border: 2px dashed #000;
}
</style>
```

```
function handleDragStart(e) {
  this.style.opacity = '0.4';  // this / e.target is the source node.
}

function handleDragOver(e) {
  if (e.preventDefault) {
    e.preventDefault(); // Necessary. Allows us to drop.
  }

  e.dataTransfer.dropEffect = 'move';  // See the section on the
DataTransfer object.

  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);
});
```

Se deben tener en cuenta varios aspectos del código utilizado en este ejemplo:

- `this`/`e.target` cambia con cada tipo de evento en función del lugar del modelo de eventos de DnD en el que nos encontremos.
- Cuando se arrastra un elemento como un enlace, hay que impedir el comportamiento predeterminado del navegador, que es abrir la página a la que dirige el enlace. Para ello, se debe llamar a `e.preventDefault()` en el evento `dragover`. Otro sistema que puede funcionar es utilizar `return false` en ese mismo controlador. No todos los navegadores lo necesitan, pero no está de más añadirlo.
- El controlador `dragenter` permite utilizar la clase "over" en lugar de `dragover`. Con `dragover`, la clase CSS se tendría que utilizar muchas veces mientras se siguiera activando el evento `dragover` al colocar el ratón sobre una columna. Finalmente, ese procedimiento se traduciría en una gran cantidad de trabajo innecesario para el renderizador del navegador. Siempre es aconsejable reducir este tipo de operaciones al mínimo.

3. Finalización de la operación de arrastre

Para que se procese la operación de soltar, se debe añadir un detector para los eventos `drop` y `dragend`. En este controlador, habrá que impedir el comportamiento predeterminado del navegador para este tipo de operaciones, que suele consistir en algún tipo de molesto redireccionamiento. Se puede evitar que el evento active el DOM con `e.stopPropagation()`.

En nuestro ejemplo de las columnas no podríamos hacer gran cosa sin el evento `drop`, pero antes debemos llevar a cabo una mejora inmediata, que consiste en utilizar `dragend` para eliminar la clase "over" de cada columna:

```
<style>
/* Prevent the text contents of draggable elements from being
selectable. */
[draggable] {
  -moz-user-select: none;
  -khtml-user-select: none;
  -webkit-user-select: none;
  user-select: none;
  /* Required to make elements draggable in old WebKit */
  -khtml-user-drag: element;
  -webkit-user-drag: element;
}
.column {
  height: 150px;
  width: 150px;
  float: left;
  border: 2px solid #666666;
  background-color: #ccc;
  margin-right: 5px;
  -webkit-border-radius: 10px;
  -ms-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
  -webkit-box-shadow: inset 0 0 3px #000;
  -ms-box-shadow: inset 0 0 3px #000;
  box-shadow: inset 0 0 3px #000;
  text-align: center;
  cursor: move;
}
.column header {
  color: #fff;
  text-shadow: #000 0 1px;
  box-shadow: 5px;
  padding: 5px;
  background: -moz-linear-gradient(left center, rgb(0,0,0),
rgb(79,79,79), rgb(21,21,21));
  background: -webkit-gradient(linear, left top, right top,
                               color-stop(0, rgb(0,0,0)),
                               color-stop(0.50, rgb(79,79,79)),
                               color-stop(1, rgb(21,21,21)));
  background: -webkit-linear-gradient(left center, rgb(0,0,0),
rgb(79,79,79), rgb(21,21,21));
  background: -ms-linear-gradient(left center, rgb(0,0,0),
rgb(79,79,79), rgb(21,21,21));
  border-bottom: 1px solid #ddd;
  -webkit-border-top-left-radius: 10px;
  -moz-border-radius-topleft: 10px;
  -ms-border-radius-topleft: 10px;
  border-top-left-radius: 10px;
  -webkit-border-top-right-radius: 10px;
  -ms-border-top-right-radius: 10px;
  -moz-border-radius-topright: 10px;
  border-top-right-radius: 10px;
}
</style>
```

Resultado:

Si has seguido detenidamente los pasos del ejemplo que se han descrito hasta ahora, puede que observes que la columna aún no se puede soltar correctamente. Introduce el objeto `DataTransfer`.

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 de la columna de origen:

```
var dragSrcEl = null;

function handleDragStart(e) {
  // Target (this) element is the source node.
  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 MIME. A continuación se indica la modificación necesaria para el procesamiento de la acción de arrastre de columna:

```
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.

Propiedades de arrastre

`dataTransfer` expone una serie de propiedades para ofrecer señales visuales al usuario durante el proceso de arrastre. Estas propiedades también se pueden utilizar para controlar la respuesta de cada elemento de destino de la operación de arrastre a un determinado tipo de datos.

`dataTransfer.effectAllowed`

> Restringe el "tipo de arrastre" que puede realizar el usuario en el elemento. Se utiliza en el modelo de procesamiento de la operación de arrastrar y soltar para la inicialización de `dropEffect` durante los eventos `dragenter`y `dragover`. Esta propiedad admite los

siguientes valores: `none`, `copy`, `copyLink`, `copyMove`, `link`, `linkMove`, `move`, `all` y `uninitialized`.

`dataTransfer.dropEffect`

Controla la información que recibe el usuario durante los eventos `dragenter` y `dragover`. Cuando el usuario coloca el ratón sobre un elemento de destino, el cursor del navegador indica el tipo de operación que se va a realizar (por ejemplo, una operación de copia, un movimiento, etc.). La propiedad de efecto admite los siguientes valores: `none`, `copy`, `link` y `move`.

`e.dataTransfer.setDragImage(img element, x, y)`

En lugar de utilizar la información predeterminada del navegador (la "imagen fantasma"), puedes establecer un icono de arrastre.

```
var dragIcon = document.createElement('img');
dragIcon.src = 'logo.png';
dragIcon.width = 100;
e.dataTransfer.setDragImage(dragIcon, -10, -10);
```

Resultado (debería verse el logotipo de Google al arrastrar estas columnas):

Arrastre de archivos

Las API de DnD permiten arrastrar archivos del escritorio a una aplicación web en la ventana del navegador. Como ampliación de este concepto, Google Chrome permite arrastrar objetos de archivo del navegador al escritorio.

Arrastre de archivos del escritorio al 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.
  }
}
```