

Tema 8. Drag and Drop

Drag and Drop	1
Introducción.....	1
El objeto dataTransfer	3
Ejemplo Drag&Drop	6
Ejemplo Drag&Drop entre navegador y escritorio.....	12
Ejercicios	17
Ejercicio 1: Crear un puzzle infantil.....	17
Lo necesario para comenzar	18
Pasos a seguir	18
Ejercicio 2: Crear un sistema para arrastrar productos a un carrito de la compra	18
Lo necesario para comenzar	19
Pasos a seguir	19

Drag and Drop Introducción

Drag and Drop, o *arrastrar y soltar*, es una de opciones más novedosas que presenta HTML5. De esta manera, podremos desplazar o copiar elementos de nuestra página de un lado a otro simplemente arrastrándolos con el ratón, de la misma manera que hacemos en cualquier sistema operativo para mover archivos de unas carpetas a otras.

Como veremos a continuación, el proceso es bastante sencillo. En primer lugar, deberemos definir cuáles serán los elementos arrastrables, y cuáles serán los contenedores donde vamos a permitir arrastrar dichos elementos, y una vez definidos especificar las características de dichos movimientos.

Una vez más, como en la mayoría de los casos en HTML5, tenemos la necesidad de utilizar un lenguaje de Script, en este caso JavaScript, para poder utilizar esta opción que nos ofrece el nuevo estándar. De hecho en este caso, prácticamente todo se realiza desde dicho lenguaje.

¡IMPORTANTE!

Podemos consultar todos los aspectos relacionados con Drag and Drop en la recomendación W3C desde este enlace: [***Drag&Drop***](#)

¡IMPORTANTE!

La posibilidad de hacer Drag&Drop está disponible en los siguientes navegadores:



Pero bien es cierto, que no en todos están disponibles todas las opciones que vamos a ir viendo a lo largo de la unidad. Para ver en funcionamiento todo, utilizaremos Chrome.

Eventos y propiedades

En primer lugar vamos a ver qué propiedades y eventos tenemos disponibles para poder arrastrar y soltar elementos en nuestra página. Mediante ellos, definiremos cuáles serán los elementos arrastrables, y qué funciones de JavaScript deberán ejecutarse a medida que vayan sucediendo dichos eventos. Son las siguientes:

draggable: Es uno de los atributos globales de HTML5. Indica si un elemento es arrastrable o no. Los valores *true* o *false*.

ondragstart: Define la función a ejecutar en el momento en el que se empieza a arrastrar un elemento arrastrable.

ondragend: Define la función a ejecutar en el momento en el que dejemos de arrastrar un elemento arrastrable.

ondragenter: Define la función a ejecutar en el momento en el que entremos al contenedor con un elemento arrastrable.

ondragover: Define la función a ejecutar mientras estemos sobre el contenedor con un elemento arrastrable.

ondrop: Define la función a ejecutar cuando soltemos un elemento arrastrable sobre el contenedor.

ondragleave: Define la función a ejecutar cuando abortamos el movimiento de arrastre sin finalizar, o cuando nos salimos de los límites donde podemos arrastrar.

Podemos apreciar que las tres primeras propiedades se asignarán a los elementos que vayamos a arrastrar, ya que definen si un elemento es arrastrable y las funciones a ejecutar durante el proceso de arrastrado, y que las cuatro últimas se aplicarán a los contenedores donde vayamos a permitir arrastrar elementos, ya que definen las funciones a ejecutar mientras un elemento entra en ellos, está sobre ellos, es soltado sobre ellos o cuando abortamos un movimiento de arrastre sin terminar.

Por lo tanto, un elemento arrastrable lo definiremos así en HTML:

```
<div id="arrastrable" draggable="true" ondragstart="f()" ondragend="f()">
```

Y un contenedor donde vayamos a permitir arrastrar elementos, lo definiremos así en HTML:

```
<div id="contenedor" ondragenter="f()" ondragover="f()" ondrop="f()" ondragleave="f()">
```

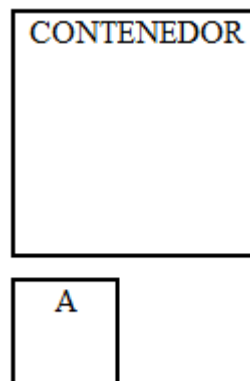
Una vez definidos que elementos serán arrastrables y donde los podremos arrastrar, las funciones definidas para cada evento se crearán en JavaScript, donde tendrá especial importancia el objeto *dataTransfer* que veremos en el siguiente apartado.

En este caso, vamos a definir un elemento arrastrable, y un elemento contenedor donde podremos arrastrarlo. El código sería el siguiente:

```
DragDrop.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Ejemplo DragDrop</title>
5   <link rel="stylesheet" href="DragDrop.css" type="text/css">
6 </head>
7 <body>
8 <div id="destino" ondragenter="return entrar(event)" ondragover="return encima(event)"
9   ondrop="return soltar(event)" ondragleave="return abortar(event)">
10   CONTENEDOR
11 </div>
12
13 <div id="arrastrable" draggable="true"
14   ondragstart="return empezar(event)" ondragend="return terminar(event)">
15   A
16 </div>
17 </body>
18 </html>
```

Vemos que para cada uno de ellos hemos aplicado las propiedades correspondientes, y les hemos asignado una serie de funciones que iremos creando a medida que vayamos estudiando el objeto *dataTransfer*. Vemos que en todas ellas hemos puesto un *return* delante, ya que dichas propiedades necesitan una respuesta por parte de la función como veremos más adelante. Y además, les pasamos a todas ellas como parámetro el evento que las a lanzado.

Por lo tanto, lo que tenemos de momento:



El elemento pequeño será el que arrastremos, y el elemento más grande será donde lo soltemos. El formato, se le han dado desde un fichero CSS referenciado en el cabecero como vemos.

El objeto dataTransfer

El objeto *dataTransfer* disponible en JavaScript, tendrá gran importancia a la hora de definir las funciones que deberán ejecutarse cuando sucedan los eventos disponibles, para que los movimientos de arrastrar y soltar funcionen de manera correcta. Este objeto dispone de una serie de propiedades y métodos que vamos a ver a continuación:

effectAllowed: Mediante esta propiedad definiremos los efectos disponibles a la hora de arrastrar. Los valores disponibles son: *none*, *copy*, *copyLink*, *copyMove*, *link*, *linkMove*, *move*, *all* y *uninitialized*.

setData(nombre, datos): Función que añade los datos especificados al objeto.

getData(nombre): Función que devuelve los datos especificados del objeto.

clearData(nombre): Función que elimina los datos especificados del objeto.

setDragImage(objeto, x, y): Mientras dura el movimiento de arrastrar, al ratón le puede acompañar una imagen en miniatura. Mediante esta función, definimos cuál será dicha imagen.

Utilizando estas propiedades y métodos de una manera adecuada, conseguiremos que los movimientos de arrastrar y soltar funcionen correctamente, y además podremos configurarlos a nuestra manera dentro de las posibilidades que ofrecen.

Siguiendo con el ejemplo que hemos comenzado en el apartado anterior, ya podemos empezar a crear las funciones que harán posible el movimiento de arrastrar y soltar que estamos buscando. Todas ellas serán creadas con JavaScript como hemos dicho.

Comenzaremos por la función *empezar()* que será la que se ejecute en el momento que empecemos a arrastrar nuestro elemento. El código será el siguiente:

```
9  function empezar(ev){
10      ev.dataTransfer.effectAllowed = 'move';
11      ev.dataTransfer.setData("Data", ev.target.getAttribute('id'));
12      ev.dataTransfer.setDragImage(ev.target, 0, 0);
13      return true;
14  }
```

Vemos como lo primero que hacemos es recoger el evento que ha lanzado la función, y mediante la propiedad *effectAllowed*, definimos los efectos disponibles para nuestro movimiento, en este caso será solo mover. Después, almacenamos el identificador del elemento que arrastramos, en la variable "Data" utilizando la función *setData()*. Definimos que mientras estamos arrastrando, es la imagen de dicho elemento la que acompaña al ratón mediante la función *setDragImage()*. Y finalmente devolvemos un *true* para dar permiso de que ese elemento sea arrastrado.

Las siguientes funciones serán las que se ejecuten al entrar al contenedor donde soltaremos el elemento, y al estar sobre él, y serán las funciones *entrar()* y *encima()*. El código será el siguiente:

```
16  function entrar(ev){
17      return true;
18  }
19
20  function encima(ev){
21      return false;
22  }
```

Lo único que hacen estas funciones en este ejemplo, es validar tanto la entrada de un elemento arrastrado, como validar que pueda ser arrastrado sobre dicho contenedor. En el primer caso está claro que devuelve un *true* para validar la entrada, pero en el segundo caso devuelve un *false*. **Pues bien, la función que lanza el evento *dragover* debe devolver un valor *false* para permitir que se pueda arrastrar un elemento sobre él.**

Después, definiremos la función *soltar()* que será la que se ejecute en el momento que soltemos el elemento sobre el contenedor. El código será el siguiente:

```
24 function soltar(ev){  
25     var arrastraId = ev.dataTransfer.getData("Data");  
26     ev.target.appendChild(document.getElementById(arrastraId));  
27     ev.stopPropagation();  
28     return false;  
29 }
```

Lo primero que hacemos, es recuperar mediante la función *getData()* el identificador del elemento arrastrado que habíamos almacenado en el objeto *dataTransfer*. Una vez tenemos ese identificador, colocamos mediante la función *appendChild()* ese elemento arrastrado sobre el contenedor, es decir, lo movemos de donde estaba y lo colocamos en el contenedor. Hecho esto, terminamos el movimiento mediante la función *stopPropagation()* propia del evento, y devolvemos un valor *false* para indicárselo al evento *drop*.

Y para finalizar el proceso, tenemos la función *terminar()*, que será la que se ejecute en el momento de dejar de arrastrar nuestro elemento. El código será el siguiente:

```
32 function terminar(ev){  
33     ev.dataTransfer.clearData("Data");  
34     return true;  
35 }
```

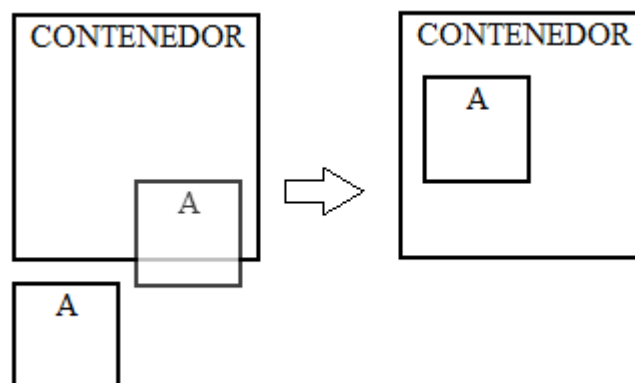
Una vez terminado el proceso por tanto, solo nos queda limpiar el objeto *dataTransfer* mediante la función *clearData()* y devolver un valor *true*.

En caso de abortar el proceso sin terminar o salirnos de los límites donde puede ser arrastrado el elemento, se ejecutará la función *abortar()*. El código será el siguiente:

```
36 function abortar(ev){  
37     return true;  
38 }
```

En este ejemplo no queremos que haga nada en especial cuando esto suceda, y por tanto, lo único que hacemos es devolver un *true*.

De esta manera ya tenemos terminado nuestro movimiento de arrastrar y soltar como vemos en la imagen:

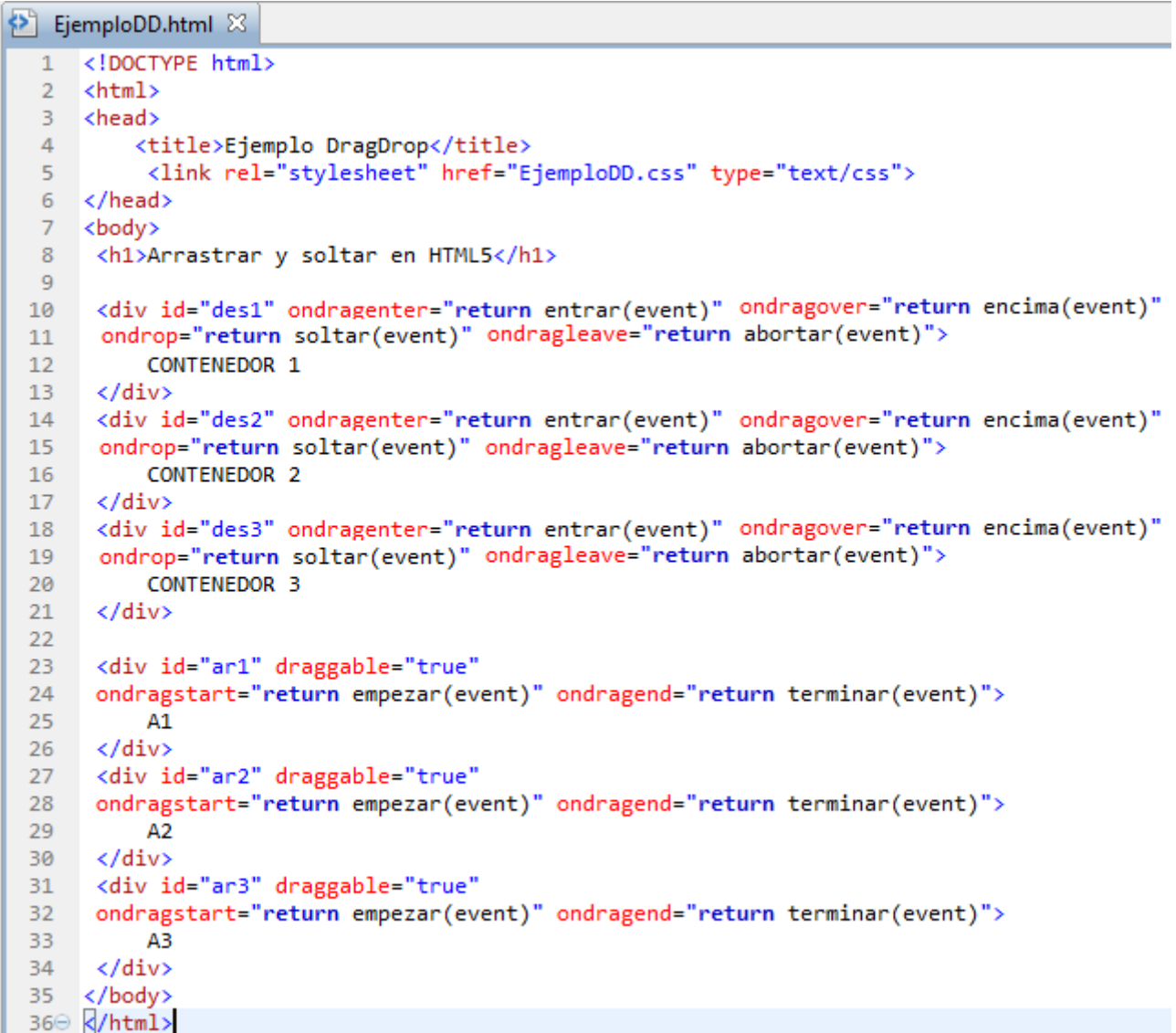


En el primer paso ya se han ejecutado las funciones *empezar()*, *entrar()* y *encima()*, y en el segundo las dos últimas de *soltar()* y *terminar()*. En caso de abortar el proceso sin terminar, o salirnos de los límites del contenedor, se ejecutaría la función *abortar()*.

Ejemplo Drag&Drop

En el siguiente ejemplo, vamos a crear tres contenedores donde podrán ser arrastrados nuestros elementos, y otros tres elementos arrastrables. Pero en este caso, vamos a limitar en cuál de los contenedores podemos soltar cada uno de los elementos, ya que no permitiremos soltar todos los elementos sobre todos los contenedores. Sobre el primer contenedor permitiremos soltar cualquiera de los elementos arrastrables, sobre el segundo contenedor únicamente se podrá soltar el segundo elemento arrastrable, y sobre el tercero únicamente el tercer elemento arrastrable.

El código HTML quedaría de la siguiente manera:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Ejemplo DragDrop</title>
5     <link rel="stylesheet" href="EjemploDD.css" type="text/css">
6 </head>
7 <body>
8     <h1>Arrastrar y soltar en HTML5</h1>
9
10    <div id="des1" ondragenter="return entrar(event)" ondragover="return encima(event)"
11        ondrop="return soltar(event)" ondragleave="return abortar(event)">
12        CONTENEDOR 1
13    </div>
14    <div id="des2" ondragenter="return entrar(event)" ondragover="return encima(event)"
15        ondrop="return soltar(event)" ondragleave="return abortar(event)">
16        CONTENEDOR 2
17    </div>
18    <div id="des3" ondragenter="return entrar(event)" ondragover="return encima(event)"
19        ondrop="return soltar(event)" ondragleave="return abortar(event)">
20        CONTENEDOR 3
21    </div>
22
23    <div id="ar1" draggable="true"
24        ondragstart="return empezar(event)" ondragend="return terminar(event)">
25        A1
26    </div>
27    <div id="ar2" draggable="true"
28        ondragstart="return empezar(event)" ondragend="return terminar(event)">
29        A2
30    </div>
31    <div id="ar3" draggable="true"
32        ondragstart="return empezar(event)" ondragend="return terminar(event)">
33        A3
34    </div>
35 </body>
36 </html>
```

Vemos que hemos creado tres elementos *DIV* que serán los contenedores y cuyos identificadores son *des1*, *des2* y *des3*, y tres elementos *DIV* que serán los elementos arrastrables y cuyos

identificadores serán *ar1*, *ar2* y *ar3*. Para cada uno de ellos hemos aplicado las propiedades correspondientes, y les hemos asignado las funciones que se ejecutarán cuando vayan sucediendo los eventos oportunos. El formato se lo hemos dado desde el fichero CSS referenciado en el cabecero.

Las funciones JavaScript quedarán de la siguiente manera:

```
8 <script type="text/javascript">
9   function empezar(ev){
10     ev.dataTransfer.effectAllowed = 'move';
11     ev.dataTransfer.setData("Data", ev.target.getAttribute('id'));
12     ev.dataTransfer.setDragImage(ev.target, 0, 0);
13     return true;
14   }
15
16   function entrar(ev){
17     return true;
18   }
19
20   function encima(ev){
21     var arrastraId = ev.dataTransfer.getData("Data");
22     var contenedorId = ev.target.getAttribute('id');
23     if(contenedorId == 'des1') return false;
24     if((contenedorId == 'des2') && (arrastraId == "ar2")) return false;
25     if((contenedorId == 'des3') && (arrastraId == "ar3")) return false;
26   }
27
28   function soltar(ev){
29     var arrastraId = ev.dataTransfer.getData("Data");
30     ev.target.appendChild(document.getElementById(arrastraId));
31     ev.stopPropagation();
32     return false;
33   }
34
35   function terminar(ev){
36     ev.dataTransfer.clearData("Data");
37     return true;
38   }
39
40   function abortar(ev){
41     return true;
42   }
43 }
44 </script>
```

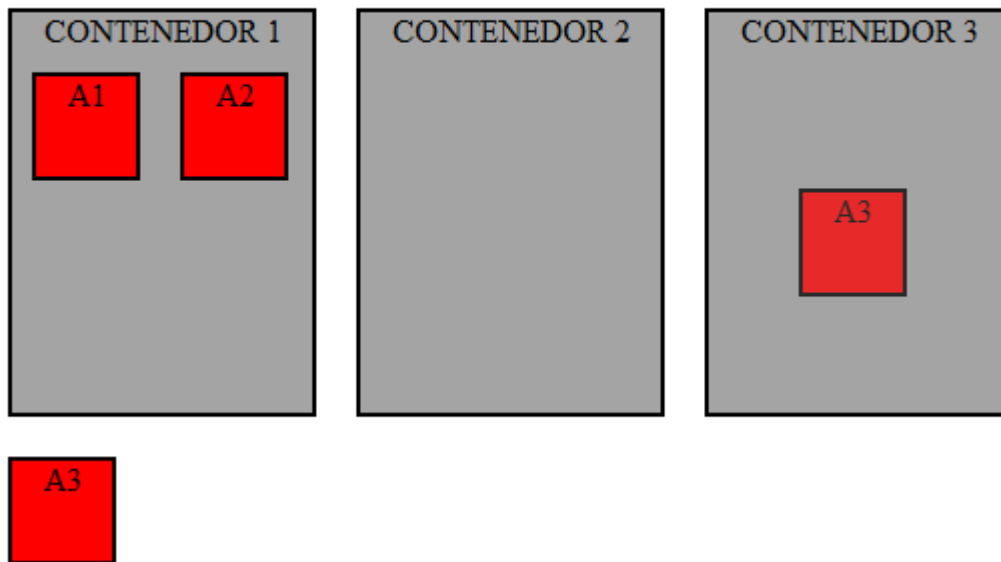
Todas las funciones son idénticas a las que hemos definido anteriormente, a excepción de la función *encima()*. Como hemos dicho antes, en esta función definimos si se permite arrastrar sobre el contenedor que lanza la función o no, por tanto, es aquí donde vamos a especificar que elementos se pueden arrastrar sobre que contenedores. Vemos que por un lado recuperamos mediante la función *getData()* el identificador del elemento arrastrable que hemos almacenado en el objeto *dataTransfer*, y después cogemos el identificador del contenedor que lanza esta función. Por lo tanto, con estos dos identificadores tenemos información suficiente para decidir donde podemos arrastrar cada uno de los elementos.

Vemos como si el contenedor es el primero, devolvemos un *false*, por lo tanto podemos arrastrar a él cualquier elemento. En los dos siguientes, solo devolvemos un *false* en caso de que el contenedor sea el segundo y el elemento arrastrable el segundo, o en caso de que el contenedor sea el tercero

y el elemento arrastrable sea el tercero. Por lo tanto solo permitimos arrastrar en esos tres casos, en todos los demás por defecto se devolverá un *true* y no estará permitido arrastrar.

Lo que veríamos en pantalla sería lo siguiente:

Arrastrar y soltar en HTML5



Vemos en la imagen, cómo hemos arrastrado ya los dos primeros elementos al primer contenedor, y nos disponemos a arrastrar el último al tercer contenedor. Si intentáramos arrastrar el tercer elemento al segundo contenedor, no nos estaría permitido.

Vamos a seguir con este ejemplo, pero ahora en lugar de ser la imagen del propio elemento que arrastramos la que acompaña al ratón hasta que lo soltamos, vamos a colocar otra imagen.

El código HTML no variará en absoluto, y el código JavaScript que vamos a utilizar en este caso será el siguiente:


```

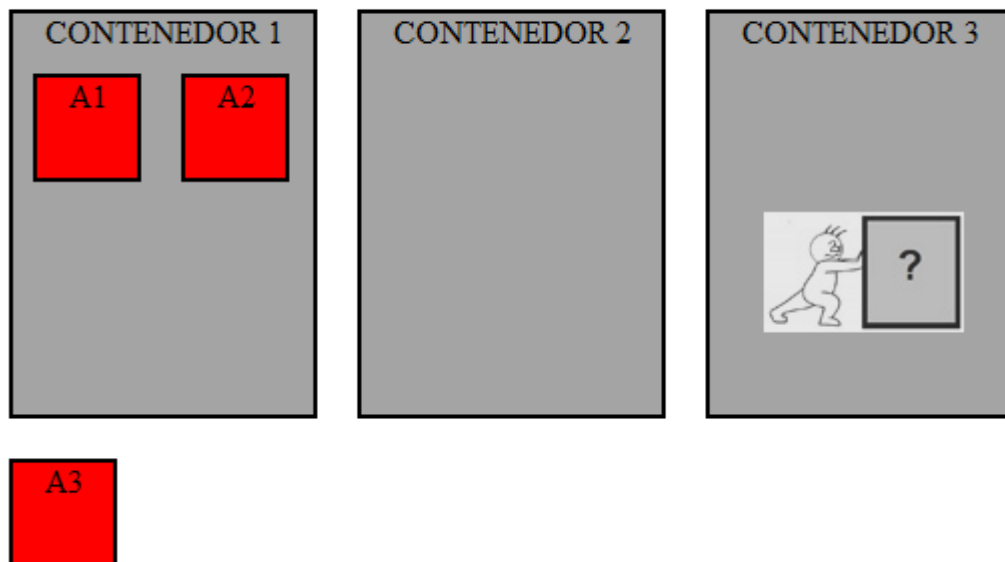
8 <script type="text/javascript">
9   function empezar(ev){
10     ev.dataTransfer.effectAllowed = 'move';
11     ev.dataTransfer.setData("Data", ev.target.getAttribute('id'));
12
13     var img = new Image();
14     img.src = "arrastrar.jpg";
15     ev.dataTransfer.setDragImage(img, 0, 0);
16
17     return true;
18   }
19
20   function entrar(ev){
21     return true;
22   }
23
24   function encima(ev){
25     var arrastraId = ev.dataTransfer.getData("Data");
26     var contenedorId = ev.target.getAttribute('id');
27
28     if(contenedorId == 'des1') return false;
29     if((contenedorId == 'des2') && (arrastraId == "ar2")) return false;
30     if((contenedorId == 'des3') && (arrastraId == "ar3")) return false;
31   }
32
33   function soltar(ev){
34     var arrastraId = ev.dataTransfer.getData("Data");
35     ev.target.appendChild(document.getElementById(arrastraId));
36     ev.stopPropagation();
37     return false;
38   }
39
40   function terminar(ev){
41     ev.dataTransfer.clearData("Data");
42     return true;
43   }
44
45   function abortar(ev){
46     return true;
47   }
48 }
49 </script>

```

Vemos que la única función que varía en este caso es la de *empezar()*. En este caso, en lugar de asignar la imagen del propio elemento que arrastramos mediante la función *setDragImage()* y el objeto *ev.target*, vamos a crear un objeto imagen asignándole la ruta de la imagen que queramos mostrar, y será dicho objeto el que asignemos mediante la función indicada.

Lo que veríamos en pantalla sería lo siguiente:

Arrastrar y soltar en HTML5



Tenemos la misma situación que antes, hemos arrastrado ya los dos primeros elementos al primer contenedor, y nos disponemos a arrastrar el último al tercer contenedor, y como vemos, en lugar de tener la imagen del propio elemento que arrastramos, tenemos la imagen *arrastrar.jpg*, que es la que hemos definido mediante la función *setDragImage()*.

Y ya para terminar con este ejemplo, vamos a hacer que el contenedor sobre el que estemos con un elemento arrastrable, cambie el nivel de opacidad, y se vea de un color más claro. Este color volverá al de inicio, tanto cuando soltemos el elemento sobre el contenedor, como cuando abortemos el movimiento o nos salgamos de los límites de arrastre.

El código HTML no variará en absoluto, y el código JavaScript que vamos a utilizar en este caso será el siguiente:

```

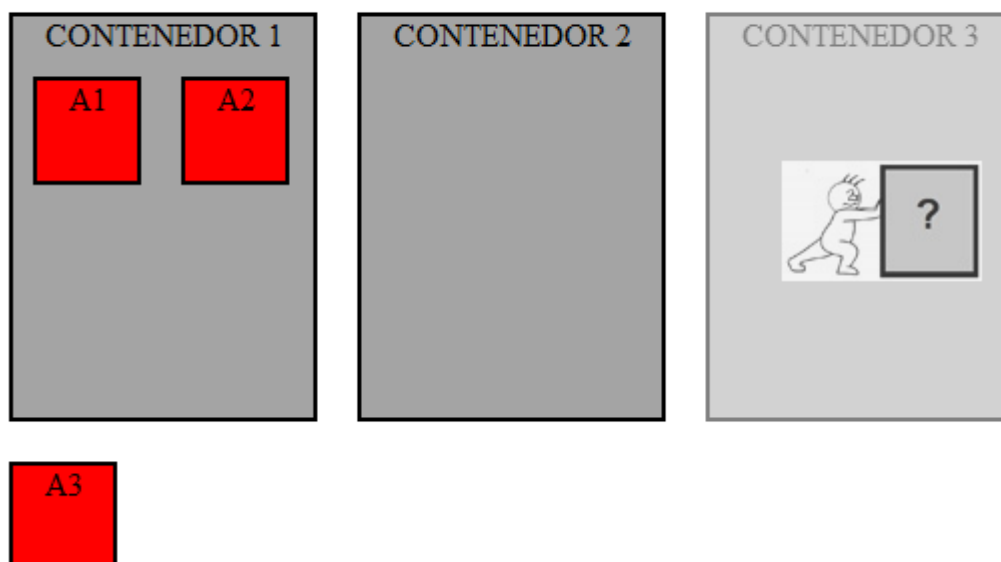
8  <script type="text/javascript">
9  function empezar(ev){
10      ev.dataTransfer.effectAllowed = 'move';
11      ev.dataTransfer.setData("Data", ev.target.getAttribute('id'));
12
13      var img = new Image();
14      img.src = "arrastrar.jpg";
15      ev.dataTransfer.setDragImage(img, 0, 0);
16
17      return true;
18  }
19
20  function entrar(ev){
21      var contenedorId = ev.target.getAttribute('id');
22      document.getElementById(contenedorId).style.opacity = "0.5";
23      return true;
24  }
25
26  function encima(ev){
27      var arrastraId = ev.dataTransfer.getData("Data");
28      var contenedorId = ev.target.getAttribute('id');
29
30      if(contenedorId == 'des1') return false;
31      if((contenedorId == 'des2') && (arrastraId == "ar2")) return false;
32      if((contenedorId == 'des3') && (arrastraId == "ar3")) return false;
33  }
34
35  function soltar(ev){
36      var arrastraId = ev.dataTransfer.getData("Data");
37      ev.target.appendChild(document.getElementById(arrastraId));
38      ev.stopPropagation();
39
40      var contenedorId = ev.target.getAttribute('id');
41      document.getElementById(contenedorId).style.opacity = "1";
42
43      return false;
44  }
45
46  function terminar(ev){
47      ev.dataTransfer.clearData("Data");
48      return true;
49  }
50
51  function abortar(ev){
52      var contenedorId = ev.target.getAttribute('id');
53      document.getElementById(contenedorId).style.opacity = "1";
54      return true;
55  }
56
57 </script>

```

Vemos que cuando entramos al contenedor, en la función *entrar()*, cambiamos la opacidad a un valor de 0.5, por lo tanto veremos más claro el color de fondo. Para volver al color original, subimos la opacidad a un valor de 1, y esto lo hacemos tanto cuando soltamos el elemento en el contenedor, en la función *soltar()*, como cuando abortamos o nos salimos de los límites, en la función *abortar()*.

Lo que veríamos en pantalla sería lo siguiente:

Arrastrar y soltar en HTML5



Tenemos la misma situación que antes, hemos arrastrado ya los dos primeros elementos al primer contenedor, y nos disponemos a arrastrar el último al tercer contenedor. Como vemos, el color de fondo de ese último contenedor está más claro, puesto que hemos bajado la opacidad, pero la de los otros dos se mantiene en el color original.

Ejemplo Drag&Drop entre navegador y escritorio

Hemos visto ya, las principales opciones que tenemos para utilizar y configurar el movimiento de arrastrar y soltar sobre nuestro navegador, pero entorno a este tema, es interesante ver la posibilidad de interactuar no solo entre elementos que tengamos en nuestra página, sino entre dichos elementos y nuestro propio equipo. De esta manera, seremos capaces de arrastrar elementos de nuestro navegador al escritorio de nuestro equipo, y también en sentido contrario. Vamos a ver a continuación un ejemplo sencillo para cada caso.

Vamos a empezar por **arrastrar un fichero desde el navegador a nuestro escritorio**, dicho movimiento proporcionará la descarga del fichero, como si por ejemplo lo descargáramos desde un enlace. El código HTML que vamos a utilizar será el siguiente:

```
NavegadorEscritorio.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>NavegadorEscritorio</title>
5
6   <style>
7     #principal{
8       border:2px solid black;
9       background:grey;
10      width:300px;
11      text-align:center;
12    }
13  </style>
14
15 </head>
16 <body>
17 <div id="principal">
18   <h4>Arrastra al escritorio para descargar el libro</h4>
19   
23   <h3>HTML5</h3>
24 </div>
25 </body>
26 </html>
```

Vemos como hemos creado un *DIV* con identificador "principal", que nos hará simplemente de contenedor de nuestros elementos, y al cuál le hemos aplicado el formato deseado desde el cabecero. Dentro de dicho elemento, hemos creado dos elementos *H4* y *H1*, que serán el texto a mostrar junto al elemento que arrastraremos hasta nuestro equipo. Pero el elemento que verdaderamente nos interesa es la imagen, este elemento será el que arrastremos hasta nuestro escritorio, y proporcione la descarga de un fichero *PDF*.

Por lo tanto, vemos que hemos definido la imagen como arrastrable mediante el atributo *draggable*, y hemos definido también el atributo *data-downloadurl* que será el que especifique los datos necesarios para este proceso. Vemos como este atributo recibe tres valores separados por los dos puntos, el primero será el **tipo MIME** que descargaremos, el segundo el **nombre con el que guardaremos el fichero en nuestro equipo**, y el tercero la **dirección URL del elemento a descargar**.

Una vez definido, vamos a crear la función JavaScript que se encargará de hacer posible el proceso, la función será la siguiente:

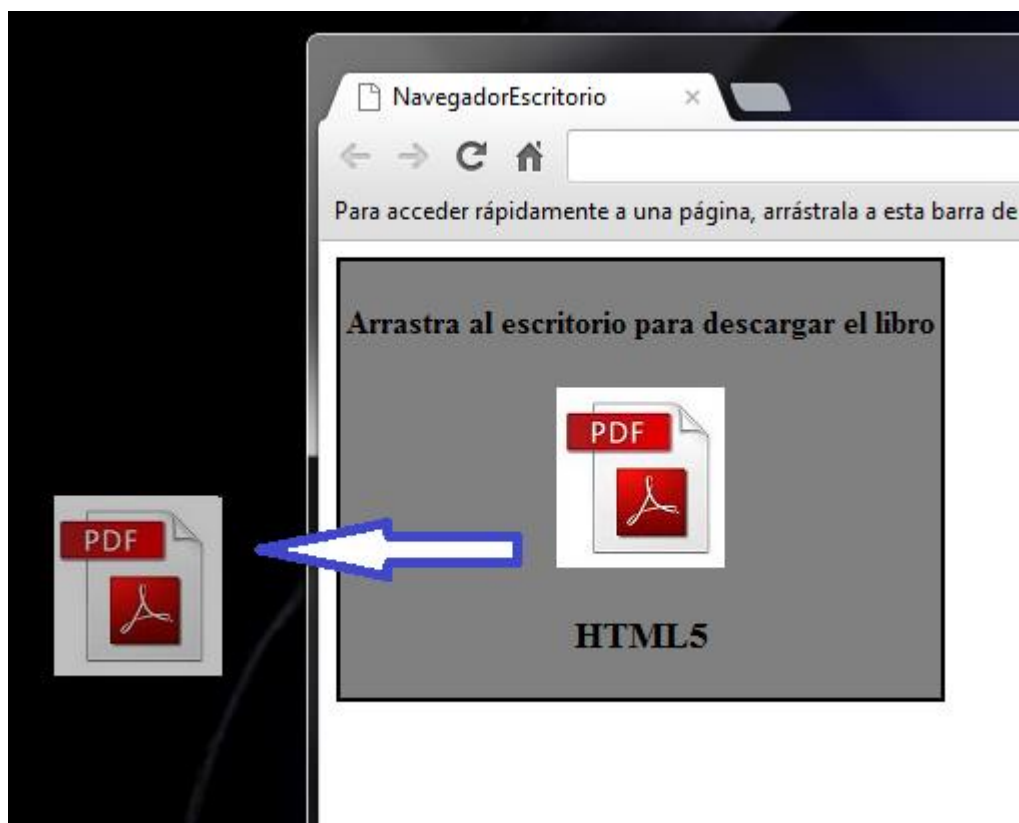
```

15 <script type="text/javascript">
16
17 window.onload = function(){
18     var file = document.getElementById("libro");
19     var fileDetails;
20
21     if(typeof file.dataset === "undefined") {
22         fileDetails = file.getAttribute("data-downloadurl");
23     } else {
24         fileDetails = file.dataset.downloadurl;
25     }
26
27     file.addEventListener("dragstart",function(ev){
28         ev.dataTransfer.setData("DownloadURL",fileDetails);
29     },false);
30 }
31
32 </script>

```

Lo primero que hacemos es coger el elemento arrastrable que hemos creado, y utilizando la propiedad *dataset* extraer los detalles de dicho fichero. Una vez extraídos, definimos para el evento *dragstart* del elemento arrastrable una función, donde asignaremos mediante la función *setData()* del objeto *dataTransfer* dichos detalles a una variable. De esta manera, cuando arrastremos la imagen hacia nuestro escritorio, la imagen es descargada sin ningún tipo de problema.

Y por último lo que veremos en el navegador será lo siguiente:



Tenemos en el navegador el *DIV* con el formato asignado, y en su interior los dos textos y la imagen arrastrable. En la imagen, estamos en pleno proceso de arrastre, en el momento que soltemos la imagen sobre el escritorio, la descarga se hará efectiva. En nuestro caso estamos trabajando con el servidor local, por lo tanto la descarga se efectuará desde allí.

A continuación, vamos a **arrastrar un fichero desde nuestro escritorio hasta el navegador**. En este caso, el movimiento no hará que el fichero suba al servidor y este disponible para utilizarlo, sino que podremos acceder a algunos datos del fichero arrastrado, nada más. El código HTML que vamos a utilizar será el siguiente:

```
EscritorioNavegador.html X
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>EscritorioNavegador</title>
5
6  <style>
7      #principal{
8          border:2px solid black;
9          background:grey;
10         width:300px;
11         min-height:300px;
12         text-align:center;
13     }
14 </style>
15
16 </head>
17 <body>
18     <div id="principal" ondrop="return soltar(event)">
19         <h2>Arrastra aquí los elementos desde el escritorio</h2>
20     </div>
21 </body>
22 </html>
```

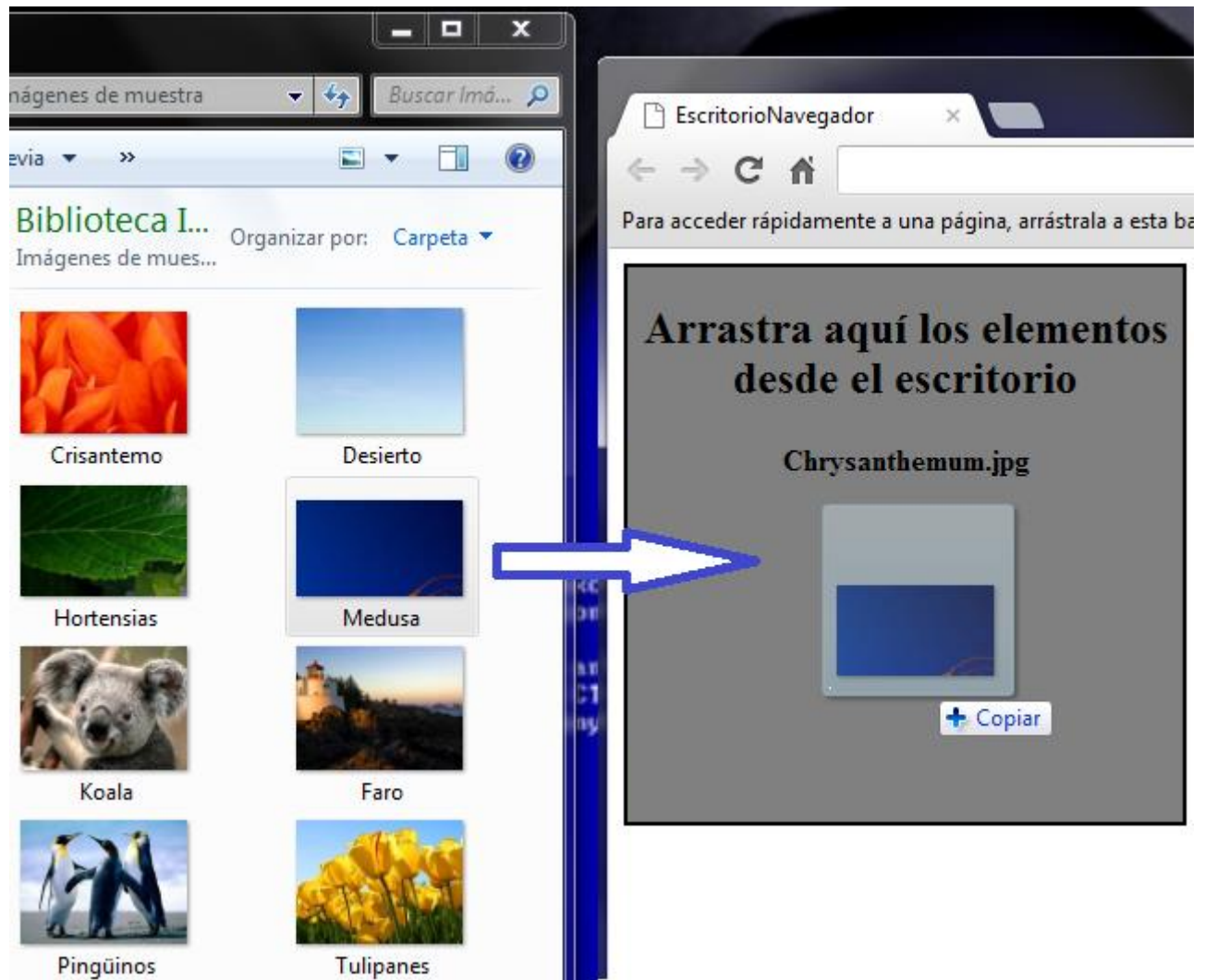
Vemos como hemos creado un *DIV* con identificador "principal", que será el contenedor donde arrastremos los elementos, y al cuál le hemos aplicado el formato deseado desde el encabezado. Dentro de dicho contenedor, hemos creado un elemento *H2*, que será el texto a mostrar junto a la información de los elementos que arrastraremos desde nuestro equipo. Únicamente, hemos definido la función a ejecutarse cuando soltemos el elemento sobre el contenedor, ya que ninguno de los demás eventos disponibles queremos que haga nada especial. Por tanto, en el momento que soltemos cualquier fichero de nuestro equipo sobre el contenedor definido, se ejecutará la función *soltar()*.

Vamos a ver como está definida dicha función en JavaScript:

```
16 <script type="text/javascript">
17 function soltar(ev){
18     var files = ev.dataTransfer.files;
19
20     for(i=0;i<files.length;i++)
21     {
22         var cadena = "<h4>" + files[i].name + "</h4>";
23         document.getElementById("principal").innerHTML += cadena;
24     }
25
26     ev.stopPropagation();
27     return false;
28
29 }
30 </script>
```

Utilizamos en este caso la propiedad *files* del objeto *dataTransfer*, que nos devuelve un array con la información de todos los ficheros arrastrados. Vamos recorriendo dicho array, y extrayendo el nombre de cada fichero arrastrado y añadiéndoselo al contenido del *DIV*. Por lo tanto, cada vez que arrastramos un fichero, escribiremos su nombre sobre el contenedor. Una vez hecho el recorrido, damos por terminado el movimiento, y devolvemos un *false* como hemos visto con anterioridad.

Por lo tanto lo que veremos en el navegador será lo siguiente:



Vamos arrastrando imágenes desde nuestro equipo, y en el momento que las soltamos se va agregando el nombre en el contenedor. Como hemos dicho al principio, este movimiento no supone que el fichero suba al servidor, por lo que no lo tenemos disponible para poder utilizarlo. Si quisiéramos utilizarlo después en nuestra página, deberíamos de completar este proceso subiendo el fichero con algún otro lenguaje o método disponible.

¡IMPORTANTE!

Esta forma de utilizar Drag&Drop entre el navegador y nuestro equipo, únicamente está disponible

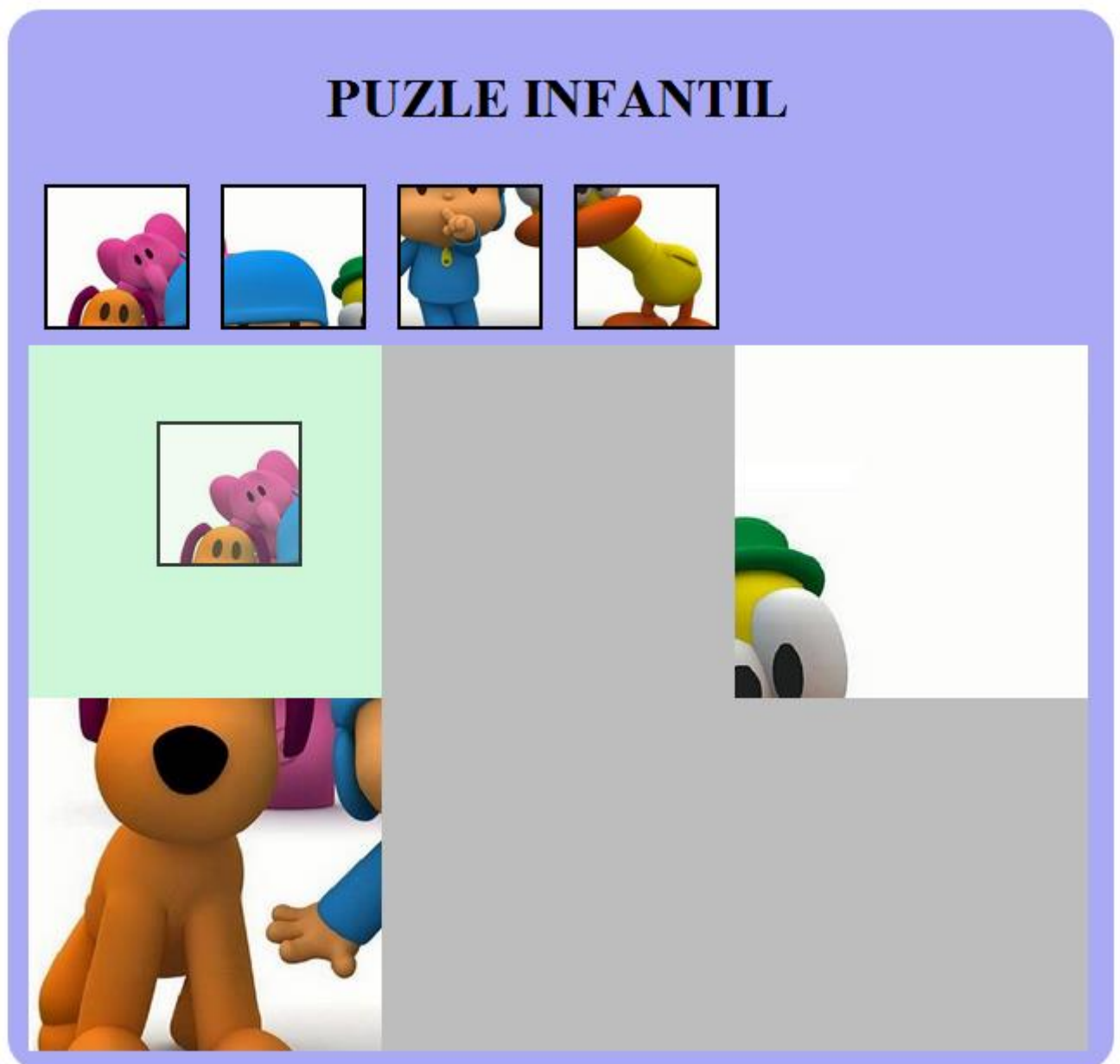
en los siguientes navegadores:



Ejercicios

Ejercicio 1: Crear un puzzle infantil

Vamos a crear un puzzle infantil de 6 piezas, aunque luego será escalable al número de piezas deseado. El usuario dispondrá de las piezas en un tamaño reducido en la parte superior, y debajo dispondrá de un espacio para ir construyendo dicho puzzle. Las piezas las deberá ir arrastrando hasta la zona donde crea que debe de colocarlas según la disposición del puzzle. Cuando este sobre la zona donde debe de soltar la pieza, el fondo de dicho espacio se volverá verde, y si no está en el lugar correcto, se volverá rojo. La página únicamente permitirá mover las piezas a los espacios donde la posición de las fichas sea la correcta. En el momento que soltemos la imagen, esta se ampliará hasta el tamaño del espacio donde la hemos colocado. En la siguiente imagen vemos el resultado que queremos conseguir:



Lo necesario para comenzar

Pasos a seguir

1. En primer lugar, estructuraremos la página para conseguir el formato que vemos en la imagen.
2. Una vez estructurada, iremos creando las imágenes en miniatura, que serán los elementos arrastrables, y asignándoles las propiedades oportunas.
3. La zona donde iremos montando el puzle, serán *DIVs*, y serán los elementos contenedor donde nos estará permitido arrastrar las imágenes. Les asignaremos también las propiedades oportunas.
4. Una vez hecho esto, tendremos que crear las funciones de JavaScript que llevarán a cabo todo el proceso que hemos indicado anteriormente.

Ejercicio 2: Crear un sistema para arrastrar productos a un carrito de la compra

Vamos a crear un sistema que nos permita simular un carrito de la compra. El objetivo será tener un listado de productos publicados, y con solo arrastrar la imagen del producto deseado al apartado donde tengamos el listado de productos que vayamos comprando, este se agregue a la lista. En este caso, únicamente pondremos en nuestra lista de la compra, el nombre del producto comprado. Por otro lado, tendremos la posibilidad de eliminar productos de nuestro carrito, lo único que tendremos que hacer, será arrastrar el nombre del producto que queramos eliminar sobre la imagen de la papelera, y el producto desaparecerá de nuestra lista.

Solo podremos arrastrar productos de nuestra lista sobre la cesta de la compra, y solo podremos arrastrar productos comprados sobre la papelera. En la siguiente imagen vemos como ya tenemos varios productos en nuestra lista, y nos disponemos a eliminar uno de ellos. En ese momento en el que estamos sobre la papelera, la imagen reduce su nivel de opacidad. Lo mismo sucede cuando arrastramos un producto a nuestra cesta, el color de fondo cambiará.

TIENDA ONLINE

IMAGEN	DESCRIPCION	PRECIO
	Ordenador DELL	987€
	Ordenador SONY	987€
	Ordenador APPLE	987€
	Ordenador PBELL	987€
	Ordenador ACER	987€



Ordenador DELL

Ordenador SONY

Ordenador APPLE



Ordenador APPLE

Lo necesario para comenzar

Pasos a seguir

1. En primer lugar, estructuraremos la página para conseguir el formato que vemos en la imagen.

2. Una vez estructurada, iremos definiendo cuáles serán los elementos arrastrables, y asignándoles las propiedades oportunas. En este caso, serán las imágenes de los productos.
3. Por otro lado, deberemos definir cuáles serán los contenedores donde podemos arrastrar los elementos, y les asignaremos también las propiedades oportunas. En este caso, serán tanto nuestra cesta de la compra, como la imagen de la papelera.
4. Una vez hecho esto, tendremos que crear las funciones de JavaScript que llevarán a cabo todo el proceso que hemos indicado anteriormente.