Arrastrar y soltar JavaScript

javascripttutorial.net/web-apis/javascript-drag-and-drop

Introducción a la API de arrastrar y soltar de JavaScript

HTML5 introdujo formalmente la especificación de arrastrar y soltar. La mayoría de los navegadores web modernos han implementado funciones nativas de arrastrar y soltar basadas en la especificación HTML5.

De forma predeterminada, solo la imagen y el texto se pueden arrastrar. Para arrastrar una imagen, simplemente mantenga presionado el botón del mouse y luego muévalo. Para arrastrar el texto, debe resaltar algún texto y arrastrarlo de la misma manera que arrastraría una imagen.

La especificación HTML5 especifica que casi todos los elementos se pueden arrastrar. Para hacer que un elemento se pueda arrastrar, agrega la `draggable` propiedad con el valor de `true` a su etiqueta HTML. Por ejemplo:

```
<div class="item" draggable="true"></div>
Lenguaje de código:  HTML, XML  ( xml )
```

Eventos en elementos arrastrables

Cuando arrastra un elemento, estos eventos se activan en la siguiente secuencia:

- `dragstart`
- `drag`
- `dragend`

Cuando mantiene presionado un botón del mouse y comienza a mover el mouse, el `dragstart` evento se activa en el elemento arrastrable que está arrastrando. El cursor cambia a un símbolo de no soltar (un círculo con una línea que lo atraviesa) para indicar que no puede soltar el elemento sobre sí mismo.

Después de que se activa el `dragstart` evento, el `drag` evento se activa repetidamente siempre que arrastre el elemento.

Y el `dragend` evento se activa cuando deja de arrastrar el elemento.

El destino de todos los eventos (`e.target`) es el elemento que se está arrastrando.

De forma predeterminada, el navegador no cambia la apariencia del elemento arrastrado. Por lo tanto, puede personalizar su apariencia según sus preferencias.

Eventos en destinos donde soltar

Cuando arrastra un elemento sobre un destino de colocación válido, estos eventos se activan en la siguiente secuencia:

- `dragenter`
- `dragover`
- `dragleave` o `drop`

El `dragenter` evento se activa tan pronto como arrastra el elemento sobre un destino de colocación.

Después de `dragenter` que se activa el `dragover` evento, se activa repetidamente mientras arrastra el elemento dentro del límite del destino de colocación.

Cuando arrastra el elemento fuera del límite del destino de colocación, el `dragover` evento deja de activarse y `dragleave` se activa.

En caso de que suelte el elemento en el objetivo, se activa el `drop` evento en lugar del `dragleave` evento.

El destino (`e.target`) de los eventos `dragenter` , `dragover` , `dragleave` y `drop` son los elementos de destino de colocación.

Destino de colocación válido

Casi todos los elementos admiten los eventos de destino de colocación (`dragenter` , `dragover` , `dragleave` y `drop`). Sin embargo, no permiten soltar por defecto.

Si suelta un elemento sobre un destino de colocación que no permite colocarlo, el `drop` evento no se activará.

Para convertir un elemento en un destino de colocación válido, puede anular el comportamiento predeterminado de los eventos `dragenter` y `dragover` llamando al `event.preventDefault()` método en sus controladores de eventos correspondientes. (Consulte la sección de ejemplos para obtener más información)

Transferir datos usando el objeto dataTransfer

Para transferir datos en una acción de arrastrar y soltar, utiliza el `dataTransfer` objeto.

El `dataTransfer` objeto es una propiedad del evento. Le permite transferir datos desde el elemento arrastrado al destino de colocación.

El `dataTransfer` objeto tiene dos métodos: `setData()` y `getData()` .

`setData()` le permite establecer los datos de la operación de arrastre en el formato y los datos especificados :

```
dataTransfer.setData(format, data)
Lenguaje de código:  CSS  ( css )
```

El formato puede ser `text/plain` o `text/uri-list` . Y los datos pueden ser una cadena que represente los datos para agregar al objeto de arrastre.

El `getData()` método recupera los datos de arrastre almacenados por el `setData()` método.

El `getData()` acepta un argumento:

```
dataTransfer(format)
```

El formato puede ser `text/plain` o `text/uri-list` . `getData()` devuelve una cadena almacenada por el método `setData()` o una cadena vacía si la operación de arrastre no incluye datos.

Ejemplo de arrastrar y soltar de JavaScript

Desarrollaremos la siguiente aplicación simple de arrastrar y soltar para demostrar la API de arrastrar y soltar de JavaScript:

Crear la estructura del proyecto.

Primero, crea una nueva carpeta llamada `drag-n-drop-basics` . Dentro de esta carpeta, cree dos subcarpetas llamadas `css` y `js` .

En segundo lugar, cree un nuevo archivo llamado `app.js` en la `js` carpeta, `style.css` en la `css` carpeta y `index.html` en la `drag-n-drop-basics` carpeta.

En tercer lugar, coloque el enlace a la `style.css` etiqueta de secuencia de comandos que se vincula a `app.js` en el `index.html` archivo de esta manera:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript - Drag and Drop Demo</title>
    <link rel="stylesheet" href="css/style.css">
</head>

<body>

    <script src="js/app.js"></script>
</body>

</html>
Lenguaje de código:  HTML, XML  ( xml )
```

Para el CSS, puede obtenerlo desde aquí .

Cree el archivo index.html

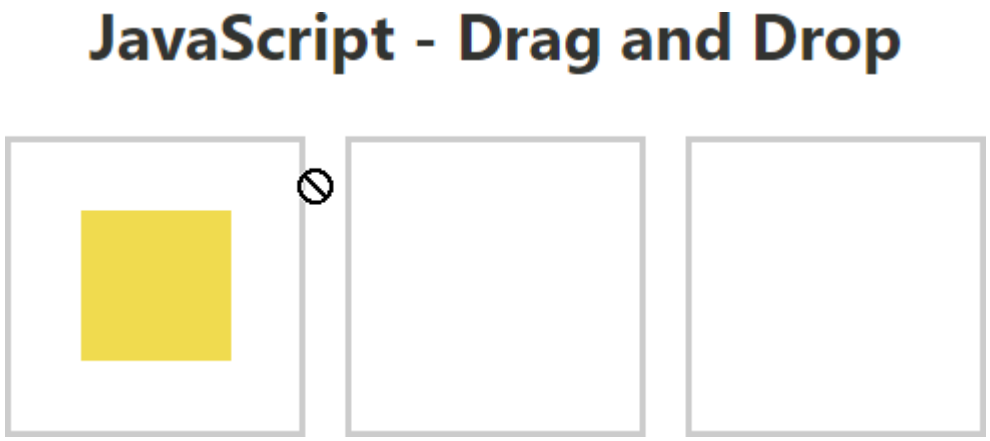
Coloque el siguiente código en el `index.html` archivo:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript - Drag and Drop Demo</title>
    <link rel="stylesheet" href="css/style.css">
</head>

<body>
    <div class="container">
        <h1>JavaScript - Drag and Drop</h1>
        <div class="drop-targets">
            <div class="box">
                <div class="item" id="item">
                </div>
            </div>
            <div class="box"></div>
            <div class="box"></div>
        </div>
    </div>
    <script src="js/app.js"></script>
</body>

</html>
```
Lenguaje de código: HTML, XML (xml)

En este archivo index.html, usamos el `.container` elemento para alinear el encabezado y el `drop-targets` elemento.

Dentro del elemento drop-targets, colocamos tres `div` elementos con la misma clase `box` . Y colocamos otro elemento div con la clase `item` en el primer cuadro.

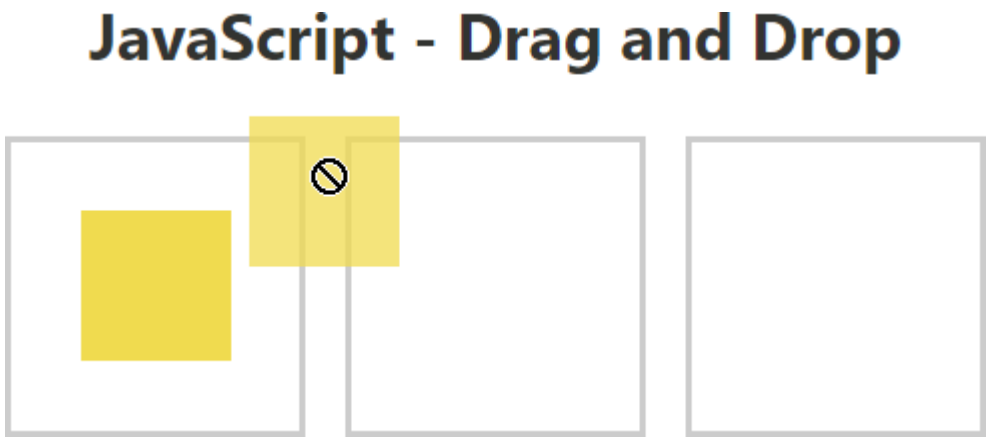
Si abre `index.html` e intenta arrastrar el cuadro amarillo, verá el cursor que indica que no puede arrastrar:



Para hacer que el elemento se pueda arrastrar, agregue la `draggable` propiedad con el valor de `true` a su etiqueta HTML de la siguiente manera:

```
<div class="item" id="item" draggable="true">
```
Lenguaje de código: JavaScript (javascript)

Ahora, si guarda el `index.html` , ábralo en el navegador nuevamente, verá que puede arrastrar el elemento del elemento de esta manera:



Manejar eventos en el elemento arrastrable

El `style.css` archivo tiene la `.hide` clase que oculta un elemento:

```
.hide {
    display: none;
}
```
Lenguaje de código: CSS (css)

En el `app.js` archivo, agregas el siguiente código:

```
// select the item element
const item = document.querySelector('.item');

// attach the dragstart event handler
item.addEventListener('dragstart', dragStart);

// handle the dragstart

function dragStart(e) {
   console.log('drag starts...');
}
```
Lenguaje de código: JavaScript (javascript)

Cómo funciona:

- Primero, seleccione el elemento arrastrable usando el `querySelector()` .
- En segundo lugar, adjunte un `dragstart` controlador de eventos al elemento que se puede arrastrar.
- Tercero, defina la `dragStart()` función para manejar el `dragstart` evento.

Si abre el archivo index.html y comienza a arrastrar el elemento arrastrable, verá el `drag starts...` mensaje en la consola.

En el `dragStart` controlador de eventos, debe almacenar el `id` del elemento que se puede arrastrar. Y necesitas ocultarlo:

```
function dragStart(e) {
    e.dataTransfer.setData('text/plain', e.target.id);
    e.target.classList.add('hide');
}
```
Lenguaje de código: JavaScript (javascript)

Si arrastra el elemento, verá que desaparece una vez que comience a arrastrar.

Para resolver esto, usas la `setTimeout()` función:

```
function dragStart(e) {
    e.dataTransfer.setData('text/plain', e.target.id);
    setTimeout(() => {
        e.target.classList.add('hide');
    }, 0);

}
```
Lenguaje de código: JavaScript (javascript)

Ahora, puede arrastrar el elemento arrastrable fuera de su posición original:

Manejar eventos en destinos donde soltar

El archivo style.css también tiene una clase CSS llamada `.drag-over` que convierte el estilo del borde del destino de colocación en discontinuo y rojo:

```
.drag-over {
    border: dashed 3px red;
}
```
Lenguaje de código: CSS (css)

En app.js, debe seleccionar los elementos de destino de colocación y manejar los eventos `dragenter` , `dragover` , `dragleave` y `drop` de estos elementos:

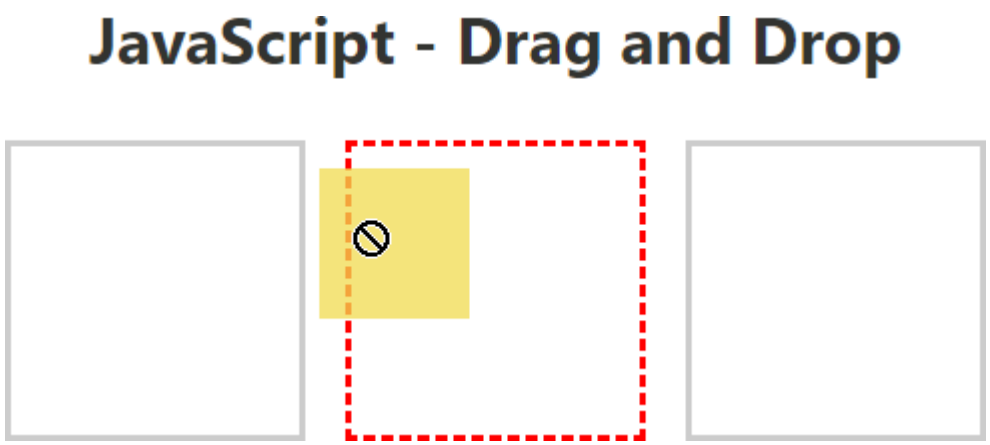
```
const boxes = document.querySelectorAll('.box');

boxes.forEach(box => {
    box.addEventListener('dragenter', dragEnter)
    box.addEventListener('dragover', dragOver);
    box.addEventListener('dragleave', dragLeave);
    box.addEventListener('drop', drop);
});

function dragEnter(e) {
}

function dragOver(e) {
}

function dragLeave(e) {
}

function drop(e) {
}
```
Lenguaje de código: JavaScript (javascript)

El estilo de borde del destino donde soltar debe cambiar cuando se produce el evento `dragenter` y . `dragover` Debería restaurar el estilo cuando ocurra el evento `dragleave` y . `drop`

Para hacerlo, agrega y elimina la `drag-over` clase al `drop` objetivo de esta manera:

```
function dragEnter(e) {
    e.target.classList.add('drag-over');
}

function dragOver(e) {
    e.target.classList.add('drag-over');
}

function dragLeave(e) {
    e.target.classList.remove('drag-over');
}

function drop(e) {
    e.target.classList.remove('drag-over');

}
```
Lenguaje de código: JavaScript (javascript)

Ahora, si arrastra el elemento arrastrable a otro destino de colocación, verá que el borde del destino de colocación cambia como se muestra en la siguiente imagen:

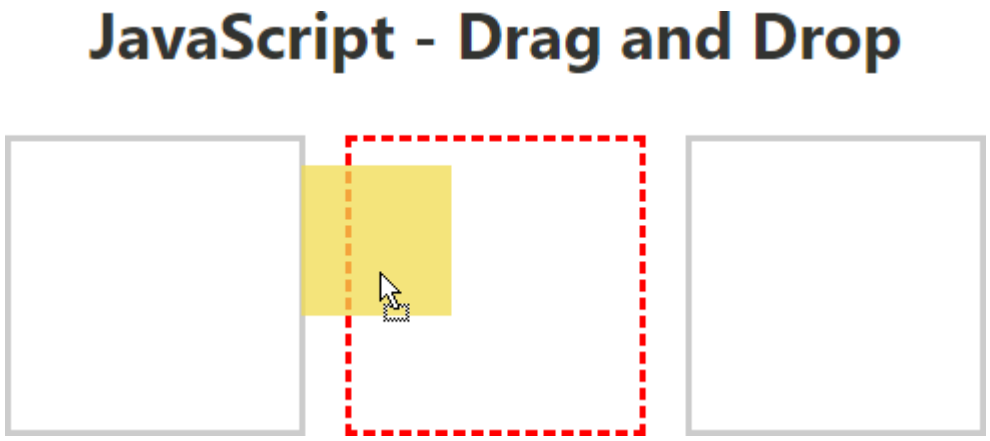


Para que el destino de colocación sea válido, debe llamar `event.preventDefault()` a los controladores de eventos `dragenter` y así: `dragover`

```
function dragEnter(e) {
    e.preventDefault();
    e.target.classList.add('drag-over');
}

function dragOver(e) {
    e.preventDefault();
    e.target.classList.add('drag-over');
}
Lenguaje de código:  JavaScript  ( javascript )
```

Si no hace esto, el `drop` evento nunca se activará porque el `div` elemento no es un destino de colocación válido de forma predeterminada.

Si arrastra el elemento arrastrable a un destino de colocación, verá que el cursor cambia para indicar que puede colocar el elemento:



Ahora, si suelta el elemento del elemento, verá que desaparece inmediatamente.

Para resolver este problema, debe agregar handle the `drop` event.

- Primero, obtenga el `id` elemento arrastrable usando el `getData()` método del `dataTransfer` objeto.
- En segundo lugar, agregue el elemento arrastrable como un elemento secundario del elemento de destino de colocación.
- Tercero, elimine la `hide` clase del `draggable` elemento.

El siguiente código muestra el `drop` controlador de eventos completo:

```
function drop(e) {
    e.target.classList.remove('drag-over');

    // get the draggable element
    const id = e.dataTransfer.getData('text/plain');
    const draggable = document.getElementById(id);

    // add it to the drop target
    e.target.appendChild(draggable);

    // display the draggable element
    draggable.classList.remove('hide');
}
```

Lenguaje de código: JavaScript (javascript)

Si arrastra y suelta el elemento arrastrable ahora, debería funcionar como se esperaba.

A continuación se muestra el archivo app.js completo:

```
/* draggable element */
const item = document.querySelector('.item');

item.addEventListener('dragstart', dragStart);

function dragStart(e) {
    e.dataTransfer.setData('text/plain', e.target.id);
    setTimeout(() => {
        e.target.classList.add('hide');
    }, 0);
}


/* drop targets */
const boxes = document.querySelectorAll('.box');

boxes.forEach(box => {
    box.addEventListener('dragenter', dragEnter)
    box.addEventListener('dragover', dragOver);
    box.addEventListener('dragleave', dragLeave);
    box.addEventListener('drop', drop);
});


function dragEnter(e) {
    e.preventDefault();
    e.target.classList.add('drag-over');
}

function dragOver(e) {
    e.preventDefault();
    e.target.classList.add('drag-over');
}

function dragLeave(e) {
    e.target.classList.remove('drag-over');
}

function drop(e) {
    e.target.classList.remove('drag-over');

    // get the draggable element
    const id = e.dataTransfer.getData('text/plain');
    const draggable = document.getElementById(id);

    // add it to the drop target
    e.target.appendChild(draggable);

    // display the draggable element
    draggable.classList.remove('hide');
}
```

Lenguaje de código: JavaScript (javascript)

Y aquí está el enlace a la demostración .

Resumen

- Agregue la `draggable` propiedad con el valor de verdadero a un elemento para que se pueda arrastrar.
- Los eventos `dragstart` , `drag` y `dragend` se activan en el elemento que se puede arrastrar.
- Los eventos `dragenter` , o `dragover` se activan en el destino de colocación. `dragleave` `drop`
- Llame `event.preventDefault()` a los controladores de eventos `dragenter` y para hacer que un elemento sea un destino de colocación válido. `dragover`
- Utilice el `event.dataTransfer` objeto con los métodos `setData()` y `getData()` para transferir datos en la operación de arrastrar y soltar.