mdn web docs

File drag and drop

HTML Drag and Drop interfaces enable web applications to drag and drop files on a web page. This document describes how an application can accept one or more files that are dragged from the underlying platform's *file manager* and dropped on a web page.

The main steps to drag and drop are to define a *drop zone* (i.e. a target element for the file drop) and to define event handlers for the `drop` and `dragover` events. These steps are described below, including example code snippets. The full source code is available in MDN's drag-and-drop repository (pull requests and/or issues are welcome).

Note that `HTML drag and drop` defines two different APIs to support dragging and dropping files. One API is the `DataTransfer` interface and the second API is the `DataTransferItem` and `DataTransferItemList` interfaces. This example illustrates the use

Define the drop *zone*

The *target element* of the `drop` event needs an `ondrop` global event handler. The following code snippet shows how this is done with a `<div>` element:

```
<div id="drop_zone" ondrop="dropHandler(event);">
  <p>Drag one or more files to this Drop Zone ...</p>
</div>
```

Typically, an application will include a `dragover` event handler on the drop target element and that handler will turn off the browser's default drag behavior. To add this handler, you need to include a `ondragover` global event handler:

```
<div id="drop_zone" ondrop="dropHandler(event);"
ondragover="dragOverHandler(event);">
  <p>Drag one or more files to this Drop Zone ...</p>
</div>
```

Lastly, an application may want to style the drop target element to visually indicate the element is a drop zone. In this example, the drop target element uses the following styling:

```
#drop_zone {
  border: 5px solid blue;
  width:  200px;
  height: 100px;
}
```

> **Note:** `dragstart` and `dragend` events are not fired when dragging a file into the browser from the OS.

Process the drop

The `drop` event is fired when the user drops the file(s). In the following drop handler, if the browser supports `DataTransferItemList` interface, the `getAsFile()` method is used to access each file; otherwise the `DataTransfer` interface's `files` property is used to access each file.

This example shows how to write the name of each dragged file to the console. In a *real* application, an application may want to process a file using the `File API` .

Note that in this example, any drag item that is not a file is ignored.

```
function dropHandler(ev) {
  console.log('File(s) dropped');

  // Prevent default behavior (Prevent file from being opened)
  ev.preventDefault();

  if (ev.dataTransfer.items) {
    // Use DataTransferItemList interface to access the file(s)
```

```
      for (var i = 0; i < ev.dataTransfer.items.length; i++) {
        // If dropped items aren't files, reject them
        if (ev.dataTransfer.items[i].kind === 'file') {
          var file = ev.dataTransfer.items[i].getAsFile();
          console.log('... file[' + i + '].name = ' + file.name);
        }
      }
    } else {
      // Use DataTransfer interface to access the file(s)
      for (var i = 0; i < ev.dataTransfer.files.length; i++) {
        console.log('... file[' + i + '].name = ' + ev.dataTransfer.files[i].name);
      }
    }
}
```

Prevent the browser's default drag behavior

The following `dragover` event handler calls `preventDefault()` to turn off the browser's default drag and drop handler.

```
function dragOverHandler(ev) {
  console.log('File(s) in drop zone');

  // Prevent default behavior (Prevent file from being opened)
  ev.preventDefault();
}
```

See also

- HTML Drag and Drop API
- Drag Operations
- HTML5 Living Standard: Drag and Drop

Last modified: Jan 28, 2022, by MDN contributors