Introduction: no need for a dragstart handler!

In this second part, we will look at how we can drag and drop files between the browser and the desktop. The process shares similarities with the methods for drag and dropping elements within an HTML document, but it's even simpler!

DRAG AND DROP FILES FROM THE DESKTOP TO THE BROWSER: THE `FILES` PROPERTY OF THE CLIPBOARD

The principle is the same as in the examples from the previous section (drag'n'drop basics), except that we do not need to worry about a `dragstart` handler. **Files will be dragged from the desktop, so the browser will do the job of copying their content to the clipboard** and make it available in our JavaScript code.

Indeed, **the main work will be done in the drop handler**, where we will use the files property from the dataTransfer/clipboard objects. This is where the browser will copy the files that have been dragged from the desktop. In the files property of the dataTransfer object (aka the clipboard).

This `files` object is the same one we saw in the chapter about the File API in the "HTML5 part 1" course: it is a collection of `file`objects (sort of file descriptors). From each `file` object we will be able to get the name of the file, its type, size, last modification date, read it, etc. like in the examples from the File API chapter from the HTML5 Part 1 course.

Source code extract: a drop handler that works on files that have been dragged and dropped from the desktop to a drop zone associated with this handler with an`ondrop=dropHandler(event);` attribute, for example:

```
function dropHandler(event) {
   // Do not propagate the event
```

```
    event.stopPropagation();
    // Prevent default behavior, in particular when we drop images or links
    event.preventDefault();
    // get the dropped files from the clipboard
    var files = event.dataTransfer.files;
    var filenames = "";
    // do something with the files...here we iterate on them and log the filenames
    for(var i = 0 ; i < files.length ; i++) {
        filenames += '\n' + files[i].name;
    }
    console.log(files.length + ' file(s) have been dropped:\n' + filenames);
}
```

At *lines 7-8*, we get the files that have been dropped. *Lines 12-15*iterate on the collection and build a string that contains the list of file names. *Line 17* displays this string on the debugging console.

Working examples are presented later on...

PREVENT THE BROWSER'S DEFAULT BEHAVIOR

When dragging and dropping images or links, we need to prevent the browser's default behavior. If we drop an image into an HTML page, the browser will open a new tab and display the image. With a .mp3 file, it will open it in a new tab and a default player will start streaming it, etc. We need to prevent this behavior in order to do custom processing with the dropped files (i.e. display an image thumbnail, add entries to a music playlist, etc.)

At the beginning of the `drop` handler in the previous piece of code, you can see the lines of code (*lines 2-6*) that stop the propagation of the drop event and prevent the default behavior of the browser. Try to drop an image or an HTTP link onto a web page: the browser will display the image or the web page pointed by the link into a new tab/window. This is not what we would like in an application that controls the drag and

drop process. These two lines are necessary to prevent the default behavior of the browser:

```
// Do not propagate the event
event.stopPropagation();
// Prevent default behavior, in particular when we drop images or links
event.preventDefault();
```

Best practice: add these lines just above the `drop`handler and to the `dragOver` handler attached to the drop zone!

... like in this example:

```
function dragOverHandler(event) {
   // Do not propagate the event
   event.stopPropagation();
   // Prevent default behavior, in particular when we drop images or links
   event.preventDefault();
   ...
}
function dropHandler(event) {
   // Do not propagate the event
   event.stopPropagation();
   // Prevent default behavior, in particular when we drop images or links
   event.preventDefault();
   ...
}
```

EXTERNAL RESOURCES

- Article from HTML5 rocks about drag'n'drop
- Article from theCSSninjas.com about dragging files from browser to desktop