

Dragging files "out" from the browser to the desktop

On most modern desktop operating systems and browsers, dragging out a file from the browser to the desktop is done natively, and does not require any particular programming. This part of the course addresses cases where you want more control over this operation (i.e renaming files on the fly).

Dragging out files from the browser to the desktop is supported by most desktop browsers, except Internet Explorer (version <= 10). Note that you can drag and drop not only from browser to desktop, but also from browser to browser.

[The W3C specification about drag and drop](#) makes no reference to dragging files out of the browser; it defines a drag and drop model based on events, but does not define the way the data that is drag and dropped will be handled.

Browser vendors have defined a *de facto* standard for dragging files out of the browser.

Specifically, they have defined:

1. A standard way to copy a file to the clipboard during a drag, if we want this file to be draggable out of the browser.
2. They have implemented the download code for copying the content of the clipboard if the drop is on the desktop.

For step 1, the file copied to the clipboard must have a key/name equal to `DownloadURL`, and the data itself should follow this format:

- `MIME type:filename:URL of source file`

Where the `filename` is the name of the file downloaded on the desktop, once the download is complete.

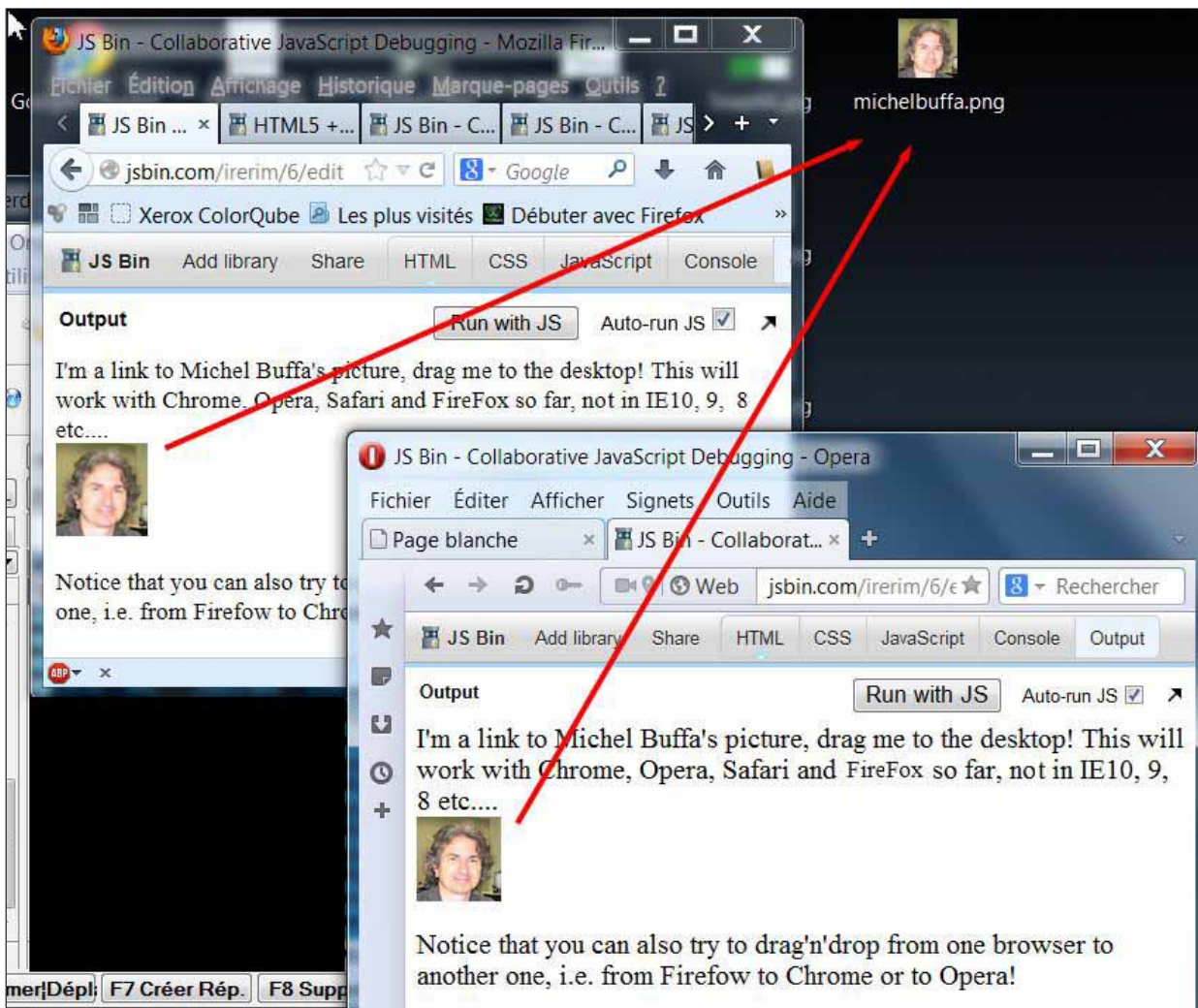
Example of a `dragStart` handler that copies a PNG image in the clipboard:

```
ondragstart="event.dataTransfer.setData('DownloadURL',  
'image/png:logo.png:http://www.w3devcampus.com/logo.png');"
```

COMPLETE EXAMPLE: DRAG AN TO THE DESKTOP

Tested with Chrome, FireFox and Opera. Does not work with Internet Explorer, even in version 10.

[Online example at JSBin](#)



Source code from this example:

```

<!DOCTYPE html>
<html lang="en">
<head>
< meta charset=utf-8 />
<title>Example of drag out from browser to desktop</title>
</head>
<body>
<a href="http://www.w3devcampus.com/wp-content/uploads/2013/01/michelbuffa.png"
draggable="true"
10.   ondragstart="event.dataTransfer.setData('DownloadURL',
      'image/png:michelbuffaDownloaded.png:http://www.w3devcampus.com/wp-
      content/uploads/2013/01/michelbuffa.png');"
      onclick="return false";
>
I'm a link to Michel Buffa's picture, drag me to the desktop!
</a>
</body>
</html>

```

Notes:

- In this example, there is no javascript - we wrote the `dragstarthandler` instructions directly in the `ondragstart` attribute (*line 12*)
- The `` element is wrapped by an `<a href>...` element that is draggable (*lines 8-14*) and not clickable (prevent default behavior, *line 13*).
- When dragged, this element calls the `dragstart` handler that just adds to the clipboard an object with a key equal to `DownloadURL`, and a value equal to `image/png:michelbuffaDownloaded.png`

You need to indicate the MIME type of the file, followed by ":", then by the filename of the file that will be copied to the desktop, and finally by the URL of the file.

EXAMPLE: DRAG OUT A CANVAS IMAGE

This example draws in one canvas, builds one `` element from the canvas content, and allows either the canvas or the image to be dragged out from the browser to the desktop:


[Interactive example on JSBin:](#)

Drag out a canvas or an image to the desktop


Demo adapted by M.Buffa from: : <http://jsfiddle.net/bgrins/xgdSC/> (courtesy of TheCssNinja & Brian Grinstead)

Drag out the image or the canvas. Notice that the filenames will be different on desktop. This example is interesting as it works with canvas data, img (in the form of a data URL or classic internal/external URL).

The Canvas:



The Image



Code from the example:

```
<html lang="en">
<head>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
  <script>
    function dragStartHandler(e) {
```

```

console.log("drag start");
var element = e.target;
var src;
10. if (element.tagName === "IMG" &&
11.     element.src.indexOf("data:") === 0) {
    src = element.src;
}
if (element.tagName === "CANVAS") {
    src = element.toDataURL();
}

if (src) {
    var name = element.getAttribute("alt") || "download";
21. var mime = src.split(";")[0].split("data:")[1];
    var ext = mime.split("/")[1] || "png";
    var download = mime + ":" + name + "." + ext + ":" + src;
    // Prepare file content to be draggable to the desktop
    e.dataTransfer.setData("DownloadURL",download);
}
}

function drawCanvas(){
31. var canvas =document.getElementById('mycanvas');
    var ctx = canvas.getContext('2d');
    var lingrad =ctx.createLinearGradient(0,0,0,150);
    lingrad.addColorStop(0, '#000');
    lingrad.addColorStop(0.5, '#669');
    lingrad.addColorStop(1, '#fff');
    ctx.fillStyle = lingrad;
41. ctx.fillRect(0, 0, canvas.width,canvas.height);
    // Create an image from the canvas
    var img = new Image();
    img.src = canvas.toDataURL("image/png");
    img.alt = 'downloaded-from-image';
    //img.draggable='true' is not necessary, images are draggable
    // by default
    img.addEventListener('dragstart',dragStartHandler, false);
    // Add the image to the document
52. document.querySelector("body").appendChild(img);
}
</script>
</head>
<body onload="drawCanvas()">
<h2>Drag out a canvas or an image to the desktop</h2>
<p>Demo adapted by M.Buffa from:
: <a href="http://jsfiddle.net/bgrins/xgdSC/">http://jsfiddle.net/bgrins/xgdSC</a> (courtesy
of TheCssNinja & Brian Grinstead)</p>
Drag out the image or the canvas. The filenames will be different on desktop. This
example is interesting as it works with canvas data, img (in the form of a data URL or
classic internal/external URL).
<br/>
62. <br/>The Canvas:<br/>
    <canvas id='mycanvas' alt='downloaded-from-canvas' draggable='true'
        ondragstart="dragStartHandler(event)"></canvas>
<br/>

```

```
<br/>  
The Image<br/>  
</body>  
</html>
```

Notice the way we build the data corresponding to the canvas or image (*lines 23-29*).