|  |
| --- |
| HTML5 Elements Cookbook |
| H5E Experiment: Drag & Drop |
| DRAFT  Published 15 July, 2011 |
|  |

|  |
| --- |
|  |
| H5E Scout Team  Windows Web Partners  Microsoft Corporation  Microsoft Confidential |

# Executive Summary

Drag and drop functionality is a ubiquitous feature of aspirational experiences, including mobile and touch devices. This experiment from the H5E team demonstrates a straightforward approach to implementing drag and drop using the DOM, and explains the process.



## Scope

This document describes an experiment conducted by the H5E scout team using Clarity Consulting. Our objective is to test the limits of HTML5 solving real-world partner questions. This document assumes an existing knowledge of JavaScript and jQuery. This document does not supersede any requirements or instructions provided by the IE team.

## Keywords

HTML5, drag, drop, JavaScript, DOM, Canvas

## Contact us

To contact us for questions or support, please email Chewy Chong ([ChewyC](mailto:ChewyC?subject=HTML5%20Cookbooks)). Feedback is welcome.

Contents

Executive Summary 2

Scope 2

Keywords 2

Contact us 2

Drag & Drop Overview 4

Technical description of drag and drop 4

Experience Walkthrough 5

How do I build this? 8

Implementing Drag and Drop 8

Conclusions and Recommendations 13

Resources 13

Relevant Web sites and specifications 13

Microsoft Resources 13

Appendix A: About H5E 14

What is an HTML5 Elements Cookbook? 14

Contact us 14

Copyright and trademark information 14

Document Revision History 15

# Drag & Drop Overview

To better understand how HTML5-based features can be used to provide aspirational experiences, the H5E team has categorized these experiences into fundamental elements.

Drag and drop functionality is a ubiquitous feature of aspirational experiences, including mobile and touch devices. This experiment from the H5E team demonstrates a straightforward method of implementing drag and drop, and explains the process.

## Technical description of drag and drop

The W3C defines drag and drop behavior as the following:

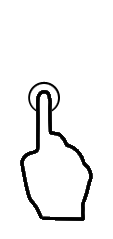
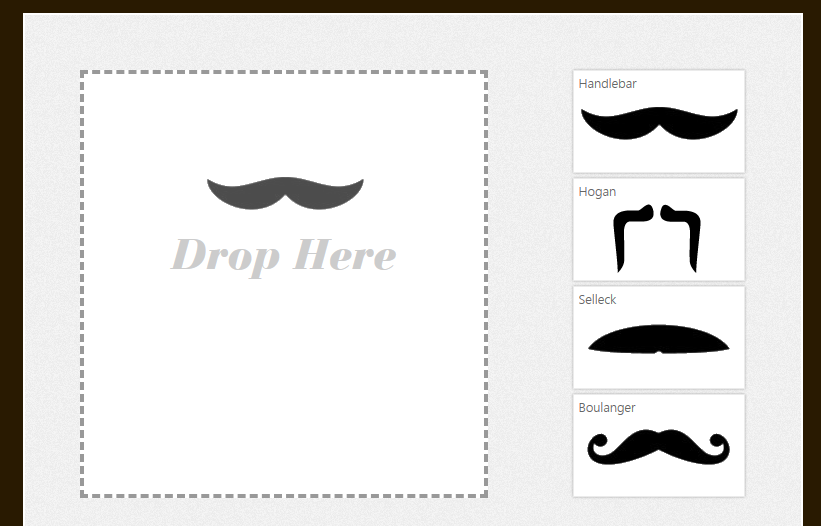
*“On a visual medium with a pointing device, a drag operation could be the default action of a mousedown event that is followed by a series of mousemove events, and the drop could be triggered by the mouse being released.*

*On media without a pointing device, the user would probably have to explicitly indicate his intention to perform a drag-and-drop operation, stating what he wishes to drag and where he wishes to drop it, respectively.*

*However it is implemented, drag-and-drop operations must have a starting point (e.g. where the mouse was clicked, or the start of the selection or element that was selected for the drag), may have any number of intermediate steps (elements that the mouse moves over during a drag, or elements that the user picks as possible drop points as he cycles through possibilities), and must either have an end point (the element above which the mouse button was released, or the element that was finally selected), or be canceled. The end point must be the last element selected as a possible drop point before the drop occurs (so if the operation is not canceled, there must be at least one element in the middle step).”-* <http://www.w3.org/TR/html5/dnd.html>

## Experience Walkthrough

The drag and drop demo demonstrates a relatively simple method of implementing drag and drop functionality using the DOM. The demo application includes the following behavior:

1. The initial page is displayed, which includes multiple mustache objects that can be dragged and an empty drop area.  
     
   
2. The user drags one of the items.  
   
3. The user positions the object in the drop area.  
     
   
4. The user drops the object in the desired location.  
     
   
5. Hovering over the object provides tools to change the rotation or scale of the object.  
     
    
6. The user finalizes the object position and scale.   
     
   

# How do I build this using HTML5?

Depending on the type of application you are building, you can achieve a drag and drop effect using primarily canvas and DOM elements. This document focuses on the DOM implementation, and will briefly talk about the canvas implementation.

***NOTE****: Many of the HTML5 Experiments are still under development. Our initial target is to build prototypes that work on current HTML5-supported browsers and tablet devices, including the iPad. The experiments do not aim for full cross-browser support at this stage, but we will likely build in graceful degradation in future updates.*

To ensure that users have a similar cross-browser experience, the following table describes the compatibility of the solutions in this document:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HTML5 Logo**HTML5 Feature** | IE6.0 | IE7.0 | IE8.0 | IE9.0 | IEPP | Chrome11 | | Chrome12 | Safari4.x | Safari5.x | Firefox3.6 | Firefox4.x | Firefox5.x | Opera11 |
| File API |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Drag and Drop (API support) |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| CSS 2D Transforms |  |  |  |  |  |  | |  |  |  |  |  |  |  |
|  | | | | | | | *Full Support* | | | | | | |  |
|  | | | | | | | Supported with Shim | | | | | | |  |
|  | | | | | | | No current support | | | | | | |  |

***NOTE****: In general, shims are not incorporated into the HTML5 Experiments at this stage. If a shim or polyfill is required for cross-browser support, see* [*http://browserexperiments.com*](http://browserexperiments.com) *for details on shim implementation.*

## Primary files in this solution

### Source Location

<https://github.com/molant/BrowserExperiments/tree/master/cookbook/8_DragDrop>

### Sample Location

<http://employees.claritycon.com/eklimczak/html5Cookbook/8_DragDrop/>

## Implementing Drag and Drop

There are a couple HTML5/CSS3 features we can use to create this experience:

* Drop file support
* 2D Transforms: In addition to being able to drag and drop an element, the goal of the project includes implementing functionality to rotate and scale the element. We can also use a space transform (x and y coordinates) to move the element. We could also use absolute positioning to move the element in the page.

2D Transforms are supported by the latest browsers. To provide support for older browsers that do not support these features, a workaround is discussed later in this document.

### Dropping the file

This demo assumes browser support for the drag and drop API (<http://www.w3.org/TR/html5/dnd.html>). This API adds several new events to your DOM elements: *dragstart*, *drag*, *dragenter*, *dragleave*, *dragover*, *drop* and *dragend*.

The names are self-explanatory and to subscribe to them you just have to use the known **addEventListener** or the **bind** function in jQuery.

In this example we will only need *drop* (to respond when a file has been dropped), *dragover* (to respond when a file is over the drop area) and *dragend* (to respond if a drop has ended, even if the file has not been dropped).

The flow of the drag and drop API is simple:

1. A user drags a file from the starting location.
2. The user drops it in a designated area in the browser window.
3. The drag and drop API throws a *drop* event with the dropped files so they can be used from JavaScript.

Using jQuery the code will look like this:

var $dropArea = $('.dropArea');

// Attach our drag and drop handlers.

$dropArea.bind({

dragover: function () {

$(this).addClass('hover');

return false;

},

dragend: function () {

$(this).removeClass('hover');

return false;

},

drop: function (e) {

// drop logic in here! Files are accessible through e.files

}

});

**Stop the propagation:**  If no one else is going to receive the *drop* event, you should cancel its propagation. Depending on the browser, the following code demonstrates cancellation:

e.preventDefault();

if (e.stopPropagation) {

e.stopPropagation();

}

### What if the browser doesn’t support the Drag & Drop API?

If the browser does not support the drag and drop API, you can use a polyfill such as the one available in <http://sandbox.knarly.com/js/dropfiles/> to support this feature. To detect if the API is supported, use Modernizr (<http://www.modernizr.com/>)

### Reading the file

**Side notes**

**jQuery:** jQueryencapsulates the original event. Keep this in mind when accessing your files!

e = e.originalEvent || e;

**Polyfill**: If you are supporting older browsers and using a polyfill, you should do the following to access the files:

var files = (e.files || e.dataTransfer.files);

Now the image is “in the browser” we want to display it. As we have said previously, the files are accessible in the drop event through the files property. The files are in base64.

To read them we need to use the File API (<http://www.w3.org/TR/FileAPI>) and its **FileReader** object. This object has several asynchronous methods:

* **readAsDataUrl**
* **readasArrayBuffer**
* **readAsBinaryString**
* **readAsText**

The **FileReader** object also includes a series of events:

* **onloadstart**: when the read starts
* **onprogress**: when reading
* **onabort**: when the read has been aborted
* **onerror**: when the read has failed
* **onload**: when the read has successfully completed
* **onloadend**: when the request has completed (either in success or failure)

For this example the demo uses **readAsDataUrl** and **onload**. The **readAsDataUrl** method is used because you can use a dataUrl with an img tag to load an image directly. The **onload** event is used because it fires only as a result of a successful load event, which is a prerequisite for this technique.

The implementation is as follows:

var $img = $('<img src="" class="uploadPic" title="" alt="" />');

var file = files[0];

var reader = new FileReader();

reader.**onload** = function (event) {

var $newImg = $img.clone().attr({

src: event.target.result,

title: file.name,

alt: file.name

});

};

reader.**readAsDataURL**(file);

### Adding the mustache image

To add an image, in this case a mustache, the most straightforward way is to use CSS 2D transforms. 2D Transforms are supported in all modern browsers. Internet Explorer versions previous to IE9 can use built-in filters to achieve the same result.

To position the element across the page we are going to use absolute positioning, setting the left and top properties. We could use 2D transforms to achieve this, but because we wanted to develop a solution that worked across multiple browsers the sample takes this approach.

Note: Using absolute positioning to place an object in the page and update it while the mouse moves is a basic technique that is covered in depth elsewhere.

### Rotating and scaling the mustache image

To rotate the moustache we are using a dedicated button that will rotate the image depending on the difference in the y axis (the x axis could also be used, if desired).

The first thing to do is subscribe to the **mousedown** event of the rotation button, and then bind to the **mousemove** and **mouseup** events of the parent container (to improve the mouse movement) to get the diff values.

$('.placed .dragRotate').live({

mousedown: function (e) {

moveHandler = function (e) {

// The logic for the diff and update here

},

upHandler = function (e) {

// The logic for the diff and update here

$container.unbind({

mousemove: moveHandler,

mouseup: $(this).onmouseup

});

return false;

};

// bind to the parent’s mousemove and mouseup

$container.bind({

mousemove: moveHandler,

mouseup: upHandler

});

return false;

}

});

When you use the **mousemove** handler, you must calculate the difference between the last updates and update the rotation angle. Depending on the browser version, the CSS properties will need a vendor prefix to work:

var rotateVal = 'rotate(' + deg + 'deg)';

$elem.css({

'-moz-transform': rotateVal,

'-o-transform': rotateVal,

'-webkit-transform': rotateVal,

'-ms-transform': rotateVal,

'transform': rotateVal,

zoom: 1

});

If you are using IE8 or previous (you can use Modernizr to detect if the browser supports 2D transforms) then you need to use the built-in filters:

var filterVal = "progid:DXImageTransform.Microsoft.Matrix(" +

cssMath.eval['deg2matrix'](deg) + ", sizingMethod='auto expand')";

$elem.css({filter: filterVal});

# Conclusions and Recommendations

This experiment focuses on two basic techniques:

* Dealing with files in your browser
* Using transforms to manipulate objects in the DOM

The file api and drag and drop features are really useful, and can help to improve your user experience and the overall level of engagement. Using AJAX to upload files to your server instead of using the older webform approach is a significant improvement for users (see GMAIL file upload experience).

Utilizing the polyfills that are now freely available, the amount of code that is necessary to support legacy and less-capable browsers is minimal. However, in a production setting you should provide a fallback scenario for users that do not have Flash or Silverlight installed (many polyfills are dependent on these two components to varying degrees.)

Using transforms to manipulate objects based on user input is a simple, practical approach, particularly when combined with falling back to the supported filters in older versions of IE. It should however be noted that the syntax is slightly more complicated. Also, overuse of this technique can lead to performance problems particularly with legacy browsers.

# Resources

## Relevant Web sites and specifications

|  |  |
| --- | --- |
| Modernizr | <http://www.modernizr.com/> |
| jQuery | <http://jquery.com/> |

## Microsoft Resources

|  |  |
| --- | --- |
| H5E Primary Contact | Chewy Chong ([ChewyC](mailto:%20chewyc?subject=HTML5%20Experiments%20Cookbooks)) |
| H5E Development Contact | Anton Molleda Quintana ([v-anmoll](mailto:v-anmoll?subject=HTML5%20Experiments%20Cookbooks)) |

# Appendix A: About H5E

## What is an HTML5 Elements Cookbook?

Each HTML5 Elements Cookbook reflects a case study of an aspirational experience that is provided by a native or component-based application. The HTML5 Experiments that are conducted by the H5E team use HTML5 and related technologies to replicate these experiences. Our primary objective is to learn from these experiments to determine if an HTML5 alternative to component-based or native implementations is both possible, and practical. Each Cookbook provides a description of the element and technical details of the HTML5 replication of that feature. We also include recommendations on whether it makes sense to pursue this approach.

## Contact us

If you need assistance with technical solutions or have a best practice to share, please contact us by sending email to Chewy Chong ([ChewyC](mailto:chewyc?subject=H5E%20Cookbooks%20and%20Documentation)).

## Copyright and trademark information

This document is Microsoft Confidential.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

 2011 Microsoft Corporation. All rights reserved.

# Document Revision History

|  |  |  |
| --- | --- | --- |
| **Reviser** | **Date** | **Revisions** |
| **v-anmoll** | 13 July 2011 | Initial draft |
| **v-jgeige** | 20 July 2011 | Quick edit |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |