HTML5 Drag & Drop, Teil I: Drag & Drop für Dateien

Die Drag & Drop-API von HTML5 entstand, wie das bei Webstandards so üblich ist, durch Reverse Engineering aus einer bereits in einem Browser vorhandenen Technologie. Das wäre auch nicht weiter tragisch, wäre nur die Basis-Technologie in diesem Falle nicht dem Internet Explorer 5 (!) entsprungen. Und ja, sie ist genau so übel, wie man sich das vorstellt; die von PPK gewählt Bezeichnung steaming pile of bovine manure trifft es eigentlich ganz gut.

Nur es hilft alles Jammern nichts, denn es gibt keine Alternative. Zwar kann man mit diversen JavaScript-Bibliotheken schon seit jeher Drag-&-Drop-Funktionen in den Browser bringen, doch diese Lösungen erreichen nicht den Funktionsumfang von HTML5. Vor allem hat sich seit den Betrachtungen von PPK 2009 einiges in Browsern und Entwickler-Gehirnen getan, so dass mittlerweile tatsächlich sinnvolle Dinge mit dieser API möglich sind. Ohne native Browserunterstützung der HTML5-API ist es zum Beispiel nicht möglich, Dateien aus dem Dateisystem in die Webseite zu ziehen und zu verarbeiten. Trotz mißratener API ist das auch gar nicht mal so schwer.

Vorbereitungen

Wenn man eine Datei ins Browserfenster zieht, versteht der Browser das normalerweise als Navigationsanweisung, d.h. er leitet auf die lokale Adresse der gezogenen Datei weiter. Möchte man eine Datei in der Webapp verarbeiten, so muss man mit ein bisschen JavaScript, HTML und CSS nachhelfen. Man nehme etwas Markup für ein Drop-Zielelement …

```
<div class="dropzone">Drop!</div>
```

… garniere es mit CSS …

```
.dropzone {
  text-align: center;
  background: #EEE;
  border: 0.2em solid #000;
  padding: 3em;
}
```

… und fange die durch HTML5 spezifizierten Drag-&-Drop-Events ab. Hiervon gibt es eine ganze Menge, wobei für den Datei-Drop-Fall nur `dragover` und `drop` wirklich relevant sind:

```
$('.dropzone').on('dragover', function(evt){
});

$('.dropzone').on('drop', function(evt){
});
```

Soweit, so gut (siehe jsFiddle).

Das Lösen der Handbremse

Warum braucht man nun zwei Events für eine einfache Drop-Operation? Die API von Drag & Drop ist so gestaltet, dass in der Ausgangslage `drop`-Events *nicht* stattfinden, wenn man eine Datei auf einem

Element droppt. Die Handbremse ist also standardmäßig angezogen. Die Überlegung dahinter war wohl, dass die meisten Elemente in einer Webseite eben keine validen Dropziele darstellen . Für diese Elemente soll weiterhin das normale Verhalten (Drop = Redirect) gelten und nur wenn man die Handbremse löst, soll etwas anderes passieren können. Lösen lässt sich die Bremse, indem man ganz einfach das `dragover`-Event abbricht:

```
$('.dropzone').on('dragover', function(evt){
  evt.preventDefault();
});

$('.dropzone').on('drop', function(evt){
  evt.preventDefault();
  window.alert('Drop!');
});
```

Ohne das `evt.preventDefault()` im `dragover`-Event würde das `drop`-Event nicht feuern und das Alert würde nicht aufpoppen (Ausnahme: Chrome ist der einzige Browser bei dem dieses Lösen der Handbremse seit einiger Zeit nicht mehr nötig ist). Ein weiteres `evt.preventDefault()` im `drop`-Event selbst verhindert schließlich den standardmäßig stattfindenden Redirect. Und damit das Ganze auch im IE funktioniert, braucht es auch ein `evt.preventDefault()` im `dragenter`-Event (MSDN). Somit haben wir eine brauchbare Ausgangslage geschaffen, denn wir können zumindest mal Dateien in den Browser ziehen, ohne dass die Standardmechaniken greifen (siehe jsFiddle).

Noch zwei Anmerkungen: die HTML5-Spezifikationen schreiben, dass das `dragenter`-Event für das Lösen der Handbremse zuständig zu sein habe; in allen Browsern außer dem IE erfüllt tatsächlich nur das `dragover`-Event diese Funktion. Im IE müssen beide Events abgebrochen werden. Das ebenfalls in den Spezifikationen zu findende `dropzone`-Attribut ist stand heute ein reiner Papiertiger.

Gezogene Dateien einlesen

Der Rest ist eigentlich relativ einfach. Im Event-Objekt des Drop-Handler gibt es eine Eigenschaft `dataTransfer`, die wiederum eine Eigenschaft `files` hat, die eine Liste der gedroppten Dateien darstellt. Dabei handelt es sich um eine ganz normale `FileList` mit `File`-Objekten wie in der File API spezifiziert – die Weiterverarbeitung der gezogenen Datei ist also mit den ganz normalen HTML5-Mitteln möglich. Um beispielweise den Inhalt einer Textdatei mit einem Alert-Fenster auszugeben pickt man sich aus der `FileList` die Datei heraus und liest sie ein:

```
$('.dropzone').on('drop', function(evt){
  evt.preventDefault();
  var file = evt.originalEvent.dataTransfer.files[0];
  if(file.type === 'text/plain'){
    var reader = new FileReader();
    reader.readAsText(file);
    reader.addEventListener('load', function(){
      window.alert(reader.result);
    }, false);
  }
});
```

Dieser Code funktioniert, wenn nur eine Datei gezogen wird oder bei mehreren Dateien die erste Datei eine Plaintext-Datei ist. Das Prinzip auf mehrere Dateien auszudehnen ist ohne weiteres möglich, aber wegen der asynchronen Natur der `FileReader`-Objekte auch kein kompletter Selbstläufer – wir ersparen uns das an dieser Stelle. Alternativ wäre es natürlich auch möglich, die Dateien in Indexed DB zu speichern, mit XHR2 auf einen Server zu laden oder anderweitig zu verarbeiten.

Erwähnenswert ist, dass der Inhalt des `dataTransfer`-Objekts wirklich nur beim `drop`-Event auslesbar ist. Bei allen anderen Events befindet sich das Daten-Objekt im *protected mode* und die API behauptet, es würden gar keine Dateien gezogen. Das ist zur Zeit ein ziemliches Problem, denn eigentlich möchte man doch das Drop-Ziel hervorheben, wenn der Nutzer eine gültige Datei auf das Element zieht. Aktuell ist dies in keinem Browser möglich, lediglich Drop-Ziele an sich können hervorgehoben werden.

Drop-Ziele hervorheben

CSS Selectors Level 4 spezifiziert Pseudoklassen für Drag & Drop, ist jedoch noch weit davon entfernt in irgendwelchen Browsern aufzuschlagen. Zur Zeit führt der einzig gangbare Weg über JavaScript, speziell die Events `dragenter` und `dragleave`. Gestaltet man sich eine hübsche Klasse für gültige Dropziele …

```
.valid {
  background: #EFE;
  border: 0.2em solid #0F0;
}
```

… so kann man diese bequem in diesen beiden Events vergeben und wieder entfernen. Außerdem muss die Klasse auch nach einem erfolgten drop entfernt werden, da für Browser offenbar ein Ende der Drop-Operation kein Ende der Drag-Operation darstellt:

```
$('.dropzone').on('dragenter', function(){
  $(this).addClass('valid');
  evt.preventDefault();
});

$('.dropzone').on('dragleave', function(){
  $(this).removeClass('valid');
});

$('.dropzone').on('drop', function(evt){
  evt.preventDefault();
  // Rest der Drop-Logik
  $(this).removeClass('valid');
});
```

Fertig! War doch gar nicht mal so schwer.

Fazit

Mit der HTML5-API für Drag & Drop Dateien im Browser zu empfangen ist ohne weiteres möglich. Zwar gibt es ein paar Mängel (z.B. die fehlende Möglichkeit, Dropziele nur bei bestimmten Dateitypen hervorzuheben), aber für einfache wie Use Cases wie Reinziehen-Hochladen-Fertig reicht es allemal. Auch mit der Browserunterstützung sieht es gut aus; der limitierende Faktor dürfte eher sein, ob die HTML5-APIs zur Weiterverarbeitung der Dateien unterstützt werden.

Betrachtet man nur den Datei-Drop, sieht die HTML5-API gar nicht so schlimm aus. Das ändert sich, sobald man versucht auch DOM-Knoten ziehbar zu machen und in Dropzielen zu empfangen. In 90% der Fälle ist man hier mit einer Non-HTML5-Lösung (also z.B. einem beliebigen jQuery-Plugin) besser beraten, falls man nicht etwas sehr bestimmtes vorhat – was genau das ist, sehen wir dann im nächsten Teil.

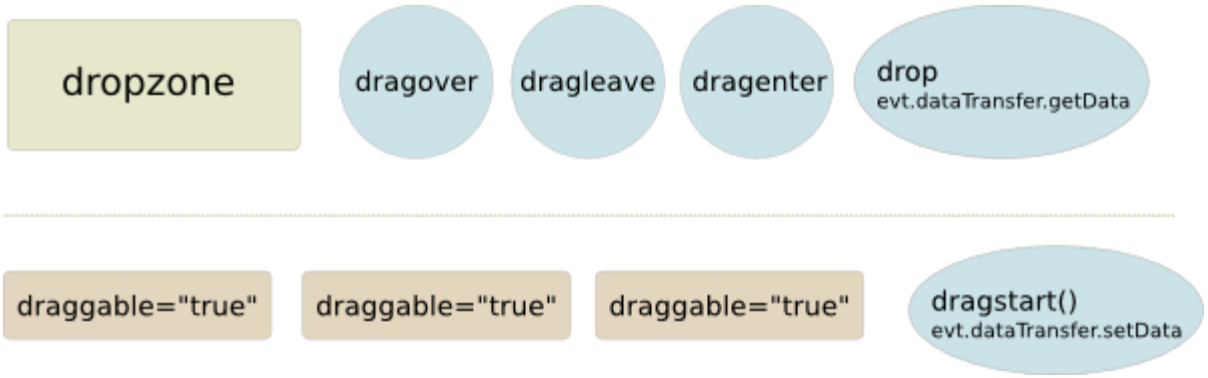
Javascript drag and drop

Ein einfaches HTML-Attribut draggable erklärt ein Element zum »Drag and Drop«, damit es auf einen anderen Bereich der Webseite gezogen werden kann.

Das HTML-Attribut draggable="true" alleine reicht noch nicht. Dazu kommen die Javascript-Events für Drag & Drop.

Drag and Drop braucht ein initiales Ereignis – z.B. wenn die Maus auf ein Element mit dem HTML-Attribut draggable klickt. Das draggable-Element muss für das drag- oder dragstart-Event registriert sein.

Die Webseite braucht ein Element als »Dropzone«. Dropzone ist ein Bereich, auf den das draggable-Element gezogen werden kann.



Events für Drag and Drop

- dragstart()
- drag()
- dragover()
- dragenter()
- dragleave()
- dragenter()
- drop()

dataTransfer.setData / getData

Alle Drag-Events haben eine Eigenschaft *dataTransfer*. *dataTransfer* enthält die Daten, die bei einer Drag-und-Drop-Operation verschoben werden.

- Die Methode *event.dataTransfer.setData()* setzt den Datentyp und die Werte der gezogenen Daten
- Die Methode *event.dataTransfer.getData()* liest die Daten

dataTransfer.setData hat zwei Parameter: den Typ der Daten und den Wert der Daten, z.B. ("text/plain", "Dieser String zieht um").

Das div id="wetter" ist Dropzone für die Wettersymbole

```
<div id="wetter">
   <ul id="wetterliste"></ul>
</div>
```

In div-Elementen mit class="thumbs" und draggable="true" liegen die Bilder mit dem Text, der in die Dropzone übernommen wird.

```
<div class="thumbs" id="schnee" draggable="true">
   <img src="bilder/schnee.png" …  />
   <p>Schnee</p>
</div>
```

Drag & Drop Probleme mit einzelnen Browsern?

Wenn Drag und Drop zu Problemen mit einzelnen Browsern führt: jquery-ndd.js ist eine Library, mit der die Unterschiede in der Implementierung von Drag & Drop in den Browsern eliminiert werden.

```
$('.thumbs').bind('dragstart', function(evt) {
      evt.dataTransfer.setData('text', this.id);
})

$('#wetter').bind('dragover', function(evt) {
      evt.preventDefault();
}).bind('dragleave', function(evt) {
      evt.preventDefault();
}).bind('dragenter', function(evt) {
      evt.preventDefault();
})
/** hier werden die Daten transferiert - die id **/.bind('drop',
function(evt) {
      var id = evt.dataTransfer.getData('text'),
          item = $('#' + id),
          wetterliste = $('#wetterliste'),
          prevFavItem = null;
      prevFavItem = null;
      prevFavItem = $('<li />', {
            text : $('p:first', item).text(),
            data : {
                  id : id
            }
      }).appendTo(wetterliste);
      evt.stopPropagation();
      return false;
})
```

Dropzone

- Blitze
- Sonne
- Mond

Drag & Drop auf TouchPads

Während Drag und Drop in den Browsern (relativ) problemlos funktioniert: Auf iPad & Co. kommt es zu Komplikationen. Die Geste »Anfassen mit der Maus und ziehen« ist auf einem Touch Device schon durch Scrollen bei »Touch und Ziehen« belegt.

Safari Developer Library: Handling Events

Um das Scrollen auf dem iPad zu unterbinden:

```
<meta name="viewport" content="width=900">
<meta name="apple-mobile-web-app-capable" content="yes">
```

Damit die Seite dann doch scrollen kann, setzt man den Block, in dem drag and drop ablaufen soll, in ein iframe.