HTML 5 – Part IV
Drag & drop

Laura Farinetti
Dipartimento di Automatica e Informatica
Politecnico di Torino
laura.farinetti@polito.it

Drag & drop

- HTML 5 includes a Drag and Drop API that brings native drag&drop support to the browser
- Drag & drop requires
  - Something to drag
  - A drop target
  - JavaScript event handlers
- See
  - https://developer.mozilla.org/en/DragDrop/Drag_and_Drop

Example

Something to drag

Drop target

index.html

Drag events

- A number of events are fired during various stages of the drag and drop operation
  - Only drag events are fired: mouse events such as mousemove are not fired during a drag operation
- The dataTransfer property of all drag events holds data about the drag and drop operation
- dragstart
  - Fired on an element when a drag is started
  - The user requests to drag the element the dragstart event is fired at
  - A listener for this event should set information such as the drag data and the image to be associated with the drag

Drag events

- dragenter
  - Fired when the mouse is first moved over an element while a drag occurs
  - A listener for this event should indicate whether a drop is allowed over this location
  - If there are no listeners, or the listeners perform no operations, then a drop is not allowed by default
  - This is also the event to listen to if you want to provide feedback that a drop is allowed such as displaying a highlight or insertion marker
- dragover
  - This event is fired as the mouse is moved over an element when a drag occurs
  - Much of the times, the operation that occurs during a listener will be the same as the dragenter event

Drag events

- dragleave
  - This event is fired when the mouse leaves an element while a drag is occuring
  - Listeners should remove any highlighting or insertion markers used for drop feedback
- drag
  - This event is fired at the source of the drag, that is, the element where dragstart was fired, during the drag operation

Drag events

- drop
  - This event is fired on the element where the drop occurs at the end of the drag operation
  - A listener would be responsible for retrieving the data being dragged and inserting it at the drop location
  - This event will only fire if a drop is successful
  - It will not fire if the user cancelled the drag operation, for example by pressing the Esc key, or if the mouse button was released while the mouse was not over a valid drop target
- dragend
  - The source of the drag will receive a dragend event when the drag operation is complete, whether it was successful or not

Something to drag

- In HTML the <img> elements and the <a> elements (with an href) are draggable by default
- In order to make another HTML element draggable, two things must be done
  - Set the draggable attribute to true on the element that you wish to make draggable
  - Add a listener for the dragstart event and set the drag data within this listener

```
<div draggable="true"
  ondragstart="event.dataTransfer.setData('text/plain',
    'This text may be dragged')">
  This text <strong>may</strong> be dragged. </div>
```

- Within the dragstart event, you can specify the drag data, the feedback image and the drag effects
  - Only drag data is required

Drag data

- All drag events have a property called dataTransfer which is used to hold the drag data
- Information to be provided
  - The data to be dragged
  - The drag feedback image which appears beside the mouse pointer during the drag operation; this image may be customized but most of the time it is not specified, and a default image is generated
  - The drag effects that are allowed

Drag data

- When a drag occurs, data must be associated with the drag which identifies what is being dragged
  - Example: when dragging the selected text within a textbox, the data associated with the drag is the text itself; when dragging a link on a web page, the drag data is the URL of the link
- The drag data contains two pieces of information
  - the type (or format, or data): a type string (e.g text/plain for text data): types are a MIME-type like string, such as text/plain or image/jpeg
  - the data value: a string of text
- When the drag begins, you add data by providing a type and the data
- During a drop event, a listener would retrieve the data being dragged and insert it at the drop location
- To set data within the dataTransfer: setData method

```
event.dataTransfer.setData("text/plain", "Text to drag");
```

Drag feedback

- When a drag occurs, a translucent image is generated from the drag target and follows the mouse pointer during the drag
- This image is created automatically
- Or, it is possible to use the setDragImage to specify a custom drag feedback image

```
event.dataTransfer.setDragImage(image, xOffset, yOffset);
```

Drag effects

- The copy operation is used to indicate that the data being dragged will be copied from its present location to the drop location
- The move operation is used to indicate that the data being dragged will be moved
- The link operation is used to indicate that some form of relationship or connection will be created between the source and drop locations
- To specify which of the three operations are allowed for a drag source: set the effectAllowed property within a dragstart event listener

```
event.dataTransfer.effectAllowed = "copy";
```

- Legal values are: “none”, “copy”, “move”, “link”, “copyMove”, “copyLink”, “linkMove”, “all” (default)

Specifying drop targets

- A listener for the dragenter and dragover events are used to indicate valid drop targets, that is, places where dragged items may be dropped
- Most areas of a web page or application are not valid places to drop data
  - The default handling for these events is to not allow a drop
- If you want to allow a drop, you must prevent the default handling by cancelling the event
- You can do this either by returning false from an attribute-defined event listener, or by calling the event's event.preventDefault method

```
<div ondragover="return false">
<div ondragover="event.preventDefault()">
```

Specifying drop targets

- It is most common to accept or reject a drop based on the type of drag data in the data transfer
  - For instance, allowing images or links or both
- To do this, you can check the types of the dataTransfer object

```
function doDragOver(event) {
  var isLink =
          event.dataTransfer.types.contains("text/uri-list");
  if (isLink)
    event.preventDefault(); }
```

- The contains method checks if the type text/uri-list is present in the list of types
  - If it is, we will cancel the event so that a drop may be allowed
  - If the drag data does not contain a link, the event will not be cancelled and a drop cannot occur at that location

Drop feedback

- There are several ways to indicate to the user that a drop is allowed at a certain location
  - The mouse pointer will update as necessary depending on the value of the dropEffect property (The actual effect that will be used, should always be one of the possible values of effectAllowed)
  - Although the exact appearance depends on the user's platform, typically a plus sign icon will appear for a 'copy' for example, and a 'cannot drop here' icon will appear when a drop is not allowed
  - This mouse pointer feedback is sufficient in many cases
- You can also update the user interface with an insertion point or highlight as needed

```
.droparea:-moz-drag-over {
  border: 1px solid black; }
```

  - The element with the class droparea will receive a 1 pixel black border while it is a valid drop target, that is, if the event.preventDefault method was called during the dragenter event

Drop feedback

- For more complex visual effects, you can also perform other operations during the dragenter event, e.g. by inserting an element at the location where the drop will occur
  - This might be an insertion marker or an element that represents the dragged element in its new location
  - You could create an image or separator element for example, and simply insert it into the document during the dragenter event
- The dragover event will fire at the element the mouse is pointing
  - You may need to move the insertion marker around a dragover event as well
  - You can use the event's clientX and clientY properties as with other mouse events to determine the location of the mouse pointer
- The dragleave event will fire at an element when the drag leaves the element
  - This is the time when you should remove any insertion markers or highlighting

Performing a drop

- When the user releases the mouse, the drag and drop operation ends
  - If the mouse is released over an element that is a valid drop target (i.e. one that cancelled the last dragenter or dragover event) then the drop will be successful, and a drop event will fire at the target
  - Otherwise, the drag operation is cancelled and no drop event is fired
- During the drop event, you should retrieve that data that was dropped from the event and insert it at the drop location
- The getData method may be used to retrieve the data from dataTransfer property
- The getData method takes one argument, the type of data to retrieve and returns the string value that was set when the setData was called at the beginning of the drag operation
  - An empty string will be returned if data of that type does not exist

Performing a drop

```
function onDrop(event) {
  var data = event.dataTransfer.getData("text/plain");
  event.target.textContent = data;
  event.preventDefault(); }
```

- Here, once we have retrieved the data, we insert the string as the textual content of the target
  - This has the effect of inserting the dragged text where it was dropped, assuming that the drop target is an area of text such as a p or div element
- In a web page, you should call the preventDefault method of the event if you have accepted the drop so that the default browser handling does not handle the dropped data as well
  - For example, when a link is dragged to a web page, Firefox will open the link
  - By cancelling the event, this behaviour will be prevented

Performing a drop

```
function doDrop(event) {
  var links =
    event.dataTransfer.getData("text/uri-list").split("\n");
  for each (var link in links) {
    if (link.indexOf("#") == 0) continue;
    var newlink = document.createElement("a");
    newlink.href = link;
    newlink.textContent = link;
    event.target.appendChild(newlink);
  }
event.preventDefault(); }
```

- The text/uri-list type actually may contain a list of URLs, each on a separate line
  - In this code, we use the split to split the string into lines, then iterate over the list of lines, inserting each as a link into the document
  - Note also that we skip links starting with a number sign (#) as these are comments

Performing a drop

- For simple cases, the special type URL to just retrieve the first valid URL in the list

```
var link = event.dataTransfer.getData("URL");
```

- Simple example:
  - http://www.w3schools.com/html/html5_draganddrop.asp

Finishing a drag

- Once the drag is complete, a dragend is fired at the source of the drag (the same element that received the dragstart event)
  - This event will fire if the drag was successful or if it was cancelled
- You can use the dropEffect to determine what drop operation occurred
  - If the dropEffect property has the value none during a dragend, then the drag was cancelled
  - Otherwise, the effect specifies which operation was performed
- A drop can occur inside the same window or over another application
- After the dragend event has finished propagating, the drag and drop operation is complete

The example: dragstart

```
<div draggable="true" id="paper"
     ondragstart="drag(this, event)"></div>
<div draggable="true" id="coke_can"
     ondragstart="drag(this, event)"></div>
<div draggable="true" id="empty_wrapper"
     ondragstart="drag(this, event)"></div>
<div draggable="true" id="bottle"
     ondragstart="drag(this, event)"></div>
<div draggable="true" id="pencil_shavings"
     ondragstart="drag(this, event)"></div>
```

```
function drag(drop_target, e) {
  e.dataTransfer.setData('Text', drop_target.id);
  document.getElementById("thanks").style.display="none";
}
```

The example: dragenter, dragover, drop

```
<div id="trash-can" ondrop="drop(this, event)"
     ondragenter="return false" ondragover="return false">
</div>
```

```
function drop(drop_target, e) {
  count++;
  var id = e.dataTransfer.getData('Text');
  document.getElementById("thanks").style.display="block";
  drop_target.appendChild(document.getElementById(id));
  document.getElementById(id).style.display="none";

  if(count==5)
  {
    document.getElementById("thanks").style.display="none";
    document.getElementById("done").style.display="block";
  }

  e.preventDefault();
}
```

Another example

drag-color

Licenza d'uso

- Queste diapositive sono distribuite con licenza Creative Commons "Attribuzione - Non commerciale - Condividi allo stesso modo 2.5 Italia (CC BY-NC-SA 2.5)"
- Sei libero:
  - di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
  - di modificare quest'opera
- Alle seguenti condizioni:
  - **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
  - **Non commerciale** — Non puoi usare quest'opera per fini commerciali.
  - **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.
- http://creativecommons.org/licenses/by-nc-sa/2.5/it/