SWEN1005

MOBILE WEB PROGRAMMING

Session Nine

DRAG AND DROP

Drag and Drop API - Definition

- Drag and Drop (DnD) is powerful User Interface concept which makes it easy to copy, reorder and deletion of items with the help of mouse clicks. This allows the user to click and hold the mouse button down over an element, drag it to another location, and release the mouse button to drop the element there.
- To achieve drag and drop functionality with traditional HTML4, developers would either have to either have to use complex JavaScript programming or other JavaScript frameworks like jQuery etc.

Drag and Drop API - Definition

- Adds the ability to drag a text selection or file to a target area on the page or another web page.
- The **draggable** attribute indicates the element can be selected and dragged.
- The **dropzone** attribute is used on the target area and defines what type of content it can accept (text or file type) and what to do with it when it gets there (**copy**, **link**, **move**).

Drag and Drop Events

Events	Description
dragstart	Fires when the user starts dragging of the object.
dragenter	Fired when the mouse is first moved over the target element while a drag is occurring. 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.
dragover	This event is fired as the mouse is moved over an element when a drag is occurring. Much of the time, the operation that occurs during a listener will be the same as the dragenter event.
dragleave	This event is fired when the mouse leaves an element while a drag is occurring. Listeners should remove any highlighting or insertion markers used for drop feedback.
drag	Fires every time the mouse is moved while the object is being dragged.
drop	The drop event is fired on the element where the drop was occurred 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.
dragend	Fires when the user releases the mouse button while dragging an object.

NOTE: Mouse events are not triggered with drag operations.

Drag and Drop API – The DataTransfer Object

- The **event.dataTransfer** returns the **DataTransfer** object associated with the event as follows:

```
function EnterHandler(event) {
  DataTransfer dt = event.dataTransfer;
  …
}
```

Drag and Drop API – The DataTransfer Object

- The event listener methods for all the drag and drop events accept the **Event** object as a parameter which has a read only attribute called **dataTransfer**.
- The *DataTransfer* object holds data about the drag and drop operation. This data can be retrieved and set in terms of various attributes associated with DataTransfer object.

DataTransfer attributes

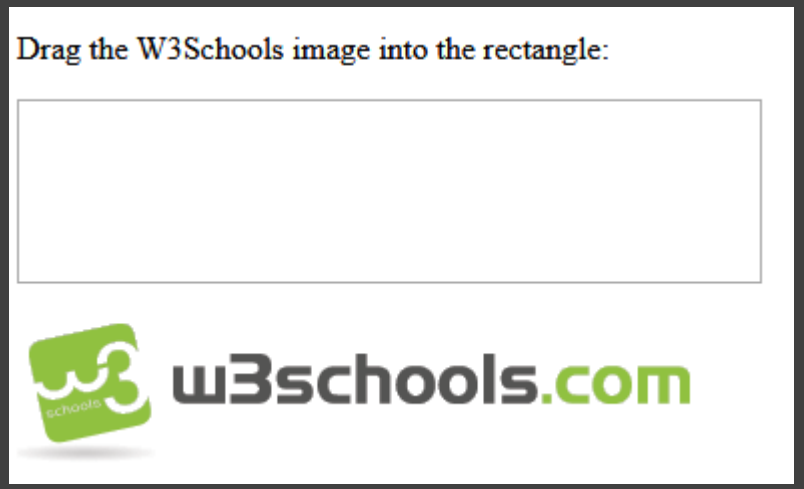
S.N.	DataTransfer attributes and their description
1	**dataTransfer.dropEffect [= value]** • Returns the kind of operation that is currently selected. • This attribute can be set, to change the selected operation. • The possible values are none, copy, link, and move.
2	**dataTransfer.effectAllowed [= value]** • Returns the kinds of operations that are to be allowed. • This attribute can be set, to change the allowed operations. • The possible values are **none, copy, copyLink, copyMove, link, linkMove, move, all and uninitialized**.
3	**dataTransfer.types** Returns a DOMStringList listing the formats that were set in the dragstart event. In addition, if any files are being dragged, then one of the types will be the string "Files".
4	**dataTransfer.clearData ([format])** Removes the data of the specified formats. Removes all data if the argument is omitted.

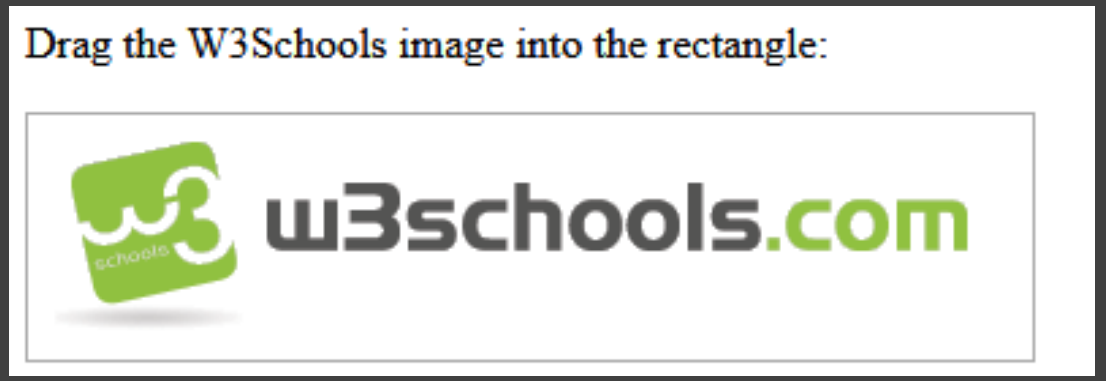
…more DataTransfer attributes

5	**dataTransfer.setData(format, data)** Adds the specified data.
6	**data = dataTransfer.getData(format)** Returns the specified data. If there is no such data, returns the empty string.
7	**dataTransfer.files** Returns a FileList of the files being dragged, if any.
8	**dataTransfer.setDragImage(element, x, y)**
	Uses the given element to update the drag feedback, replacing any previously specified feedback.
9	**dataTransfer.addElement(element)** Adds the given element to the list of elements used to render the drag feedback.

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69">
```

```
<style>
#div1 {
    width: 350px;
    height: 70px;
    padding: 10px;
    border: 1px solid #aaaaaa;
}
</style>
<script>
```





Lets take a closer look at the example

- **Make an Element Draggable**
  - First of all: To make an element draggable, set the draggable attribute to true:
  - <img draggable="true">

Lets take a closer look at the example

- **What to Drag - ondragstart and setData()**
  - Then, specify what should happen when the element is dragged.
  - In the example, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.
  - The dataTransfer.setData() method sets the data type and the value of the dragged data:
  - function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
    }
  - In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

Lets take a closer look at the example

- **Where to Drop - ondragover**
  - The ondragover event specifies where the dragged data can be dropped.
  - By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.
  - This is done by calling the event.preventDefault() method for the ondragover event:
  - *event*.preventDefault()

Lets take a closer look at the example

- **Do the Drop - ondrop**
  - When the dragged data is dropped, a drop event occurs.
  - In the example above, the ondrop attribute calls a function, drop(event):
  - function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
    }

In summary

- Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)
- Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method
- The dragged data is the id of the dragged element ("drag1")
- Append the dragged element into the drop element

```html
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1, #div2 {
    float: left;
    width: 100px;
    height: 35px;
    margin: 10px;
    padding: 10px;
    border: 1px solid black;
}
</style>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<h2>Drag and Drop</h2>
<p>Drag the image back and forth between the two div elements.</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
  <img src="img_w3slogo.gif" draggable="true" ondragstart="drag(event)"
id="drag1" width="88" height="31">
</div>

<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

</body>
</html>
```

Quick Review

- Drag and Drop API