HTML5

Lesson 6: **HTML5 Multimedia and Drag and Drop**

Windows, Bing, PowerPoint, Internet Explorer, Visual Studio, WebMatrix, DreamSpark, and Silverlight are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

The information contained in this tutorial is provided without any express, statutory, or implied warranties. Neither the authors, Microsoft Corporation, nor its resellers or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this tutorial.

Created for Microsoft by Avlade: [www.Avlade.com](http://www.Avlade.com)

Copyright 2011 by Microsoft Corporation.

Certain portions excerpted from the *HTML5 Digital Classroom* are copyright 2011 by AGI Training, and are used by permission. For more information about the *HTML5 Digital Classroom* book visit <http://www.digitalclassroombooks.com/Web-Design/HTML5-Digital-Classroom-Book.html>

Big Buck Bunny ([www.BigBuckBunny.org](http://www.BigBuckBunny.org)) ©2008 Copyright Blender Foundation: [www.Blender.org](http://www.Blender.org), licensed under the Creative Commons Attribution 3.0 license.

Lesson 6

HTML5 Multimedia and Drag and Drop

## Learning objective:

In this lesson, you’ll learn to incorporate multimedia using the HTML5 <video> and <audio> tags and incorporate the HTML5 drag-and-drop feature into your website to let your users actively interact with your site.

## Starting Up

You will work with several files from the HTML5\_06lessons folder in this lesson. Make sure you have loaded the HTML5lessons folder onto your hard drive. See “Loading lesson files” in the Starting Up section of this book.

## HTML5 multimedia and interactivity

In this lesson you’ll learn how to add multimedia content to your pages using the new HTML5 <video> and <audio> elements. Additionally, you’ll learn the foundations of how to add drag and drop functionality to your web pages with the Drag and Drop API. Although both of these HTML5 features may not seem to have much in common at first glance, both features illustrate a key aspect of HTML5 which is browser support. In the following exercises you will not only learn the basics of adding multimedia and drag and drop functionality, but also the techniques necessary to ensure cross-browser compatibility.

## Drag and Drop on the Web

Drag-and-drop operations let users visually position elements on the screen instead of positioning elements by clicking a button. In the realm of software applications and operating systems this ability to drag and drop is well established and intuitive to use. Drag and drop functionality has been slow to translate to the web, and the drag and drop features you may have encountered on web sites were likely done with custom JavaScript or via a browser plug-in such as Flash or Silverlight.

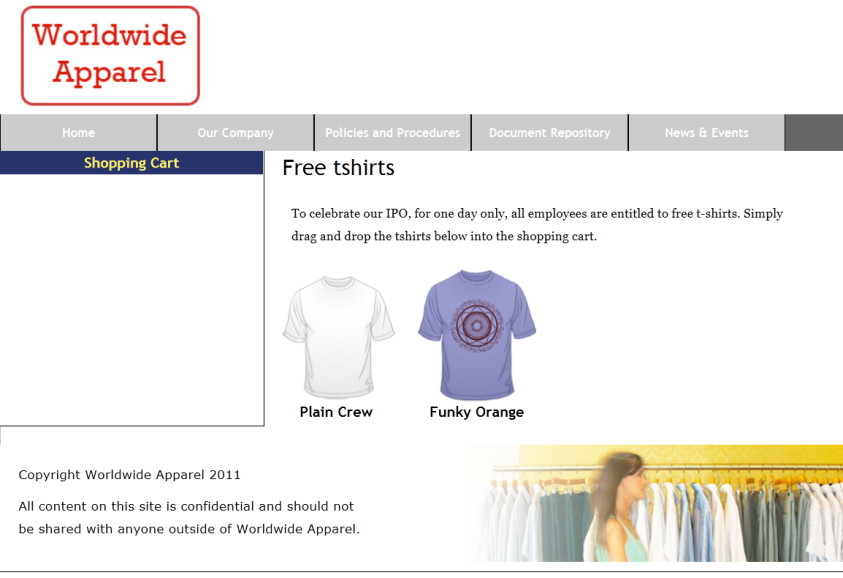
The HTML5 Drag and Drop API is based on the original Internet Explorer implementation. Some details have changed and certain browsers have an alternate syntax, but a standardized API is close to being finalized. At the time of this writing, support for certain drag-and-drop features are not shared between browsers. For example, some browsers allow selections or files from other windows or applications to be dropped into the browser; others do not.

Note that in most browsers, you can drag images, links, and text selections by default. This functionality was included to allow selections from the browser to be dragged into other applications. You can also drag file and text selections from the operating system into some browsers.

## Cross-Browser Drag and Drop

This lesson focuses on the drag-and-drop features that are supported across multiple browsers. Several differences exist in the visual indicators of a drag-and-drop operation; the functionality of the operation remains fairly consistent with only a small number of fallbacks.

1. Using your text editor, open the file 06\_shopping.html found within the HTML5\_06lessons folder. Save a copy of this file as 06\_shopping\_work.html. Open the file in your browser; notice that a shopping cart exists on the page. The goal of this activity is to create a drag-and-drop interface for purchasing items. We have added the initial JavaScript code for you, but you will be adding the remaining code to make the page fully functional.



The initial appearance of your page.

Move the mouse cursor over the products on the page. The cursor pointer, as well as the outline of the images, change. This will let users know that they will be able to drag the image. These rules are set in the base.css style sheet by using the class .tile\_list which has been applied to the div element containing the t-shirt images. You will learn about other visual indicators throughout this lesson.

1. Switch to your text editor and look within the second <script> section. (The first <script> element is being used for the Modernizr library, which helps older browsers recognize HTML5 elements). Locate the variable shoppingcart found within the setup function. This variable has been assigned a reference to the shopping cart div. Your code will use this reference to register the div as a drop zone.

Add the following line of code to prevent the default behavior of the dragenter event:

function setup(){

  shoppingcart = document.getElementById('shoppingcart');

  shoppingcart.ondragenter = preventDefault;

}

By default on some browsers, the dragenter event displays an icon indicating that an area is not a drop zone. You need to assign this event, the preventDefault function, to maintain consistency between browsers. The code below has already been added to your script section and defines the preventDefault function, which requires three specific tasks to occur to prevent the default action of some browsers:

function preventDefault( event ){

  event.stopPropagation();

  event.preventDefault();

  return false;

}

The stopPropagation function prevents other containers from responding to this event.

The preventDefault function prevents the default behavior of the event on some browsers.

Since preventDefault is an event callback function, returning false prevents the default behavior in some browsers.

1. In some browsers, the dragover event displays an icon similar to the dragenter event, indicating that this container is not a valid drop zone. To prevent the default behavior of the dragover event, add the code (highlighted in red) after the line of code from the previous step.

function setup(){

  shoppingcart = document.getElementById('shoppingcart');

  shoppingcart.ondragenter = preventDefault;

  shoppingcart.ondragover = preventDefault;

}

1. Scroll down to the bottom half of your HTML code. To allow dragging from within the list containing each image, add an ondragstart attribute to the div that has a CSS class called tile\_list, and assign to the attribute the value shown below:

<div class="tile\_list" ondragstart="dragStarted(event);">

At the time of this writing, the Drag and Drop API was not designed to perform drag-and-drop operations; the Drag and Drop API allows developers to respond to the events, not actually move elements. The following implementation provides the drag-and-drop capabilities you need for a basic shopping cart.

1. Within the dragStarted function, already defined for this activity, add the following line of code to save a reference to the element dragged by the user into the predefined dragging variable:

function dragStarted( event ){

  dragging = event.target;

}

1. To implement the drop behavior for the shopping cart, use the dragDrop function to respond to the shopping cart’s drop event:

function setup(){

  shoppingcart = document.getElementById('shoppingcart');

  shoppingcart.ondragenter = preventDefault;

  shoppingcart.ondragover = preventDefault;

  shoppingcart.ondrop = dragDrop;

}

1. Within the predefined dragDrop function, enter the following line of code to add to the shopping cart the object currently being dragged:

function dragDrop(event){

  addToCart( dragging, 1 );

}

| Making any object draggable in HTML5 |
| --- |
| Note that on some browsers you can, by default, drag images, links, and text selections and save their inability to be dropped, all without a JavaScript callback. To let users drag objects to other elements, add the draggable attribute with a value of true, as shown below:  <li draggable="true">List Item 1</li> |

1. Choose File > Save and preview this page in the browser. Depending on the browser you are using, you will have dramatically different results if you drag one of the images onto the shopping cart. In Webkit browsers, such as Safari 5 or Chrome 11, you will get the desired effect and the image will appear in the cart. In other browsers, such as Firefox 4, the default behavior of an image dropped into a drop zone is to open that image in the same window in which the drop zone exists. To ensure this behavior does not occur, return to your text editor and add the following line of code:

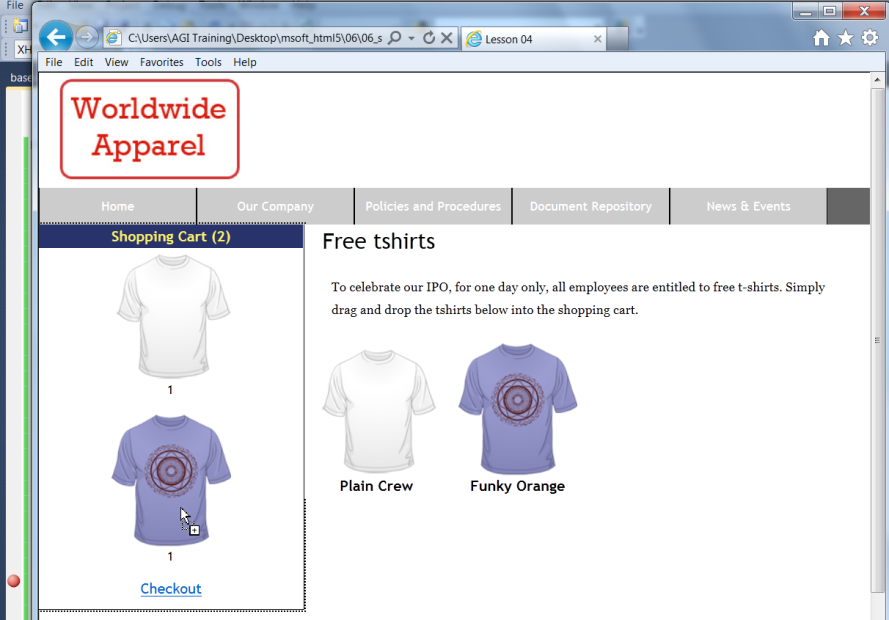
function dragDrop(event){

  addToCart( dragging, 1 );

  return preventDefault(event);

}

1. Save and test this file in your browser. Dragging t-shirts into the shopping cart adds them to the shopping cart for purchase.



The shopping cart keeps track of the items dropped within it, displays those items, and displays a checkout link.

You can choose to display an alternate image when a drag-and-drop operation starts and use any element as a visual indicator, including a canvas. The canvas provides several options, such as images or the ability to display dynamic text; for example, item names and prices.

1. Within the dragStarted function, add the following lines of code to create a canvas containing the source image and the price:

function dragStarted( event ){

  dragging = event.target;

  var dragImage = document.createElement( 'canvas' );

  var ctx = dragImage.getContext( '2d' );

  ctx.drawImage(dragging, 0, 0);

  ctx.font = 'bold 1.8em sans-serif';

  ctx.fillText( '$5', 30, 100);

}

The only visual difference from this change is text added below the image, but this will only be obvious in supported browsers (more on this in step 12). The code you added in this step is specific to the canvas element. For more information about the capabilities of the canvas element, see “Working with Canvas” in Lesson 5 of this curriculum.

1. To use the newly created canvas as the drag image for the event target, you must call the setDragImage function in the event’s dataTransfer object. Add following line of code directly after the code from the previous step:

function dragStarted( event ){

  var dragImage = document.createElement( 'canvas' );

  ...

  event.dataTransfer.setDragImage( dragImage, 0, 0);

}

The setDragImage function receives three arguments. The first argument is a reference to an element used as the drag image. The other two arguments are the x and y offset for the image. At the time of this writing, full support for the setDragImage function was not available in all browsers.

The canvas element is essential for this application, because the setDragImage function only allows a single element to be used as the drag image. With a canvas element, you can combine text, images, and vector graphics as a visual indicator.

1. Save all changes and preview the result in your browser. Drag the t-shirt onto the shopping cart. If you are using Internet Explorer 9 you will **not** see any difference at this point from the behavior of your page after step 9. This is because Internet Explorer 9 does not support the setDragImage function. A browser such as Firefox 5 does support the setDragImage function and when a user is dragging the t-shirt, the price of the object will be superimposed over the image.



Internet Explorer 9 does not have support for the setDragImage function and will not display as intended.



A browser that has support for the setDragImage will display superimposed text as the user drags the image.

## Adding multimedia video and audio elements

Adding video and sound to a web page is one way to make your website more engaging. Media provides a way to grab the attention of your visitors, and it also provides a way to reach audiences that would otherwise bypass long lengths of text. HTML5 directly addresses the need to play both video and audio. Before the advent of HTML5, you needed third party plug-ins such as Flash, QuickTime, or Silverlight to show video. HTML5 has replaced this need by specifying an HTML video element that runs natively in the browser and integrates with JavaScript.

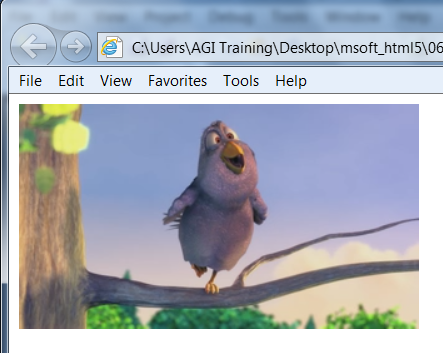
In this section, you will learn to add video to your web page such that the video begins automatically when the page loads, and takes advantage of the native browser player controls.

1. From the HTML5\_06lessons folder, open the 06\_video.html page and save a copy of this file as **06\_video\_work.html**. On line 8 inside the body element, add the following line of code:

<video src='videos/BigBuck.mp4' autoplay></video>

As is the case with the image element, the video element has a src attribute where the value points to the location of the video file you want to play. The autoplay attribute tells the browser to begin playing the video as soon as the page loads. Save the file and preview the web page in Internet Explorer 9. You should see a page similar to the figure below; the video should being playing automatically.

In earlier versions of Internet Explorer or in other browsers such as Firefox or Chrome you will likely not see any video, an issue you will address shortly!



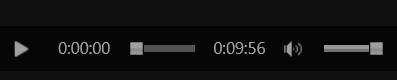
The video element displayed in Internet Explorer.

Since there is a standard way to declare a video in HTML, the browser developers have the responsibility of ensuring the browser follows the standard and displays the video properly. As a Web developer, your job is to focus on your content and customizations. You will now look at few ways you can do this.

1. Close the browser and switch back to your text editor. To provide more control over the video playback, remove the autoplay attribute and add the controls attribute so your line of code appears as follows (highlighted in red):

<video src='videos/BigBuck.mp4 controls ></video>

Save your work, and preview the page in your browser (depending on your browser’s security settings you may have to click the “Accept Blocked Content’ button). Place your cursor over the movie and the default controls will appear. The standard controls in Internet Explorer 9 are a Play/Pause button, seek bar (click and drag to forward or reverse through the video), volume mute button and volume slider. Keep in mind that these native browser controls will have a different appearance in different browsers.



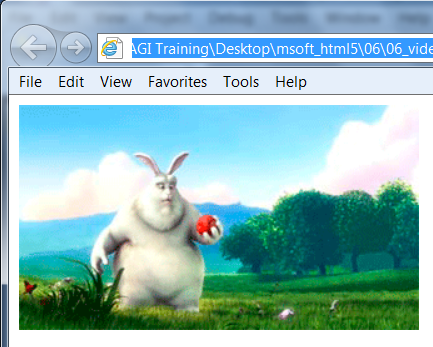
The default player controls in Internet Explorer 9.

1. Click the play button and the video should begin. Drag the slider bar and the current position of the video should change accordingly. Refresh your browser page and notice that now that the autoplay attribute is removed, the first frame is a black rectangle that appears on the screen until the video begins playing. If you want to display a custom preview image instead of the first frame or black rectangle, you need to add the poster attribute. Close your browser and switch to your text editor.
2. Add the poster attribute after the controls attribute and set the value to "poster320.png" as follows (highlighted in red):

<video src='videos/BigBuck.mp4' controls poster='poster320.png'> </video>

Setting the poster attribute tells the browser to load an image and place it above the video element. Once the video begins, the image disappears and the video is displayed.

Save your file and preview the page in your browser. Before you click the “Allow Blocked Content” button you will see the poster image now instead of a black screen.



A preview image is now shown when the page loads.

1. Return to your text editor and add the following width and height attribute and values:

<video src='videos/BigBuck.mp4' controls poster='poster320.png' width='320' height='180'>

By default, the video element is resized to fit the encoded media file. To control the size of the video element, you can use the width and height attributes. In general, it is good practice to put these values in to match the size of the video. It is also possible to scale a video by changing the width and height values, but this is not a recommended solution however. Manually scaling a video larger will result in a fuzzy or degraded image. Manually scaling a video smaller is not visually destructive, but you are forcing the user to download a larger file than is necessary. A better solution would be to export another version of the video in smaller (or larger) dimensions.

1. Save your file.

## Additional attributes

In addition to the autoplay, controls and poster attributes, the video element has a few other attributes for use:

* audio: This attribute allows you to mute the volume of your video. Currently, the only supported value is 'muted'.
* loop: This attribute will trigger your video to automatically begin playing again when it reaches the end.
* preload: This attribute will allow you to begin loading the video as soon as the page loads and can reduce the amount of time it takes to load when the user clicks play.

## Adding support for more browsers

There are many formats and codecs for video, and not all are supported equally across the different HTML5-enabled Web browsers. Each video file acts as a container for multiple files that contain the audio and video. The *Ogg format* is open standard, open source–friendly, and is supported natively by the latest versions of Chrome, Firefox, and Opera. Another popular choice is the *MP4 format*, specifically, an MP4-containing video that uses the H.264 codec and audio that uses the AAC codec. Both H.264 and AAC codecs also support multiple levels of profiles, which are used to provide different levels of compression and quality. To reach the widest array of devices and browsers, you should use the baseline profile for H.264 video and the “low complexity” profile AAC.

MP4 files encoded as explained in the previous paragraph are supported by the latest versions of Internet Explorer, Safari, iOS, and Android. Making your videos available in both Ogg and MP4 formats lets you to reach the most users with modern browsers and devices. In this section, you will learn to add multiple source files to a video element, which allows the browser to select the file based on the format it supports.

1. In your 06\_video.html document remove the entire src attribute (including the reference to the BigBuck.mp4 file) from the video element.
2. Add a source element as a child of the video element and set the src attribute of the source element to 'videos/BigBuck.mp4' as follows (highlighted in red).

<video controls poster='poster320.jpg' width='320' height='180'>

<source src='videos/BigBuck.mp4' />

</video>

1. Add another child source element below the one from the previous step and set the src attribute to videos/BigBuck.ogg. as follows (highlighted in red):

<video controls poster='poster320.jpg' width='320' height='180'>

<source src='videos/BigBuck.mp4' />

<source src='videos/BigBuck.ogg' />

</video>

1. Save your file and refresh the web page in a browser. When you open your HTML code in Internet Explorer or Safari, the first source file is used because the MP4 format is supported. When you open your HTML code in Chrome, Firefox, or Opera, the second source file is used because the browsers support the Ogg file format. The web browser checks source files for compatibility in the order they appear in the video element.

For example, if a user with Firefox opened this page, the browser would download enough of the MP4 formatted file in an attempt to load it. When the file can’t be loaded, the browser moves to the next source file which happens to be OGG, a format supported by Firefox. To assist browsers in determining compatible files and prevent the user from downloading unsupported files, you can use the type attribute of the source element. The value of the type attribute describes the file format, video codec, and audio codec of the source file.

1. Add a type attribute to the first source element to describe the MP4 file format using the following syntax:

<source src='videos/BigBuck.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'/>

This tells the browser that the “BigBuck.mp4” file is saved in the MP4 format, the video encoded with H.264 using the baseline profile, and the audio is encoded with the AAC codec using the “low complexity” profile.

1. Now add the following code to the Ogg source:

<source src='videos/BigBuck.ogg' type='video/ogg; codecs="theora,

vorbis"'/>

This tells the browser that the “BigBuck.ogg” file is saved in the Ogg file format, and that the video was made with the Theora codec, and that the audio is encoded with the Vorbis codec.

When the browser loads your HTML code, it checks for compatibility based on the type attribute and saves time and bandwidth by not downloading every file. The last step is to add additional code for older browsers that do not support HTML5: when your page is loaded in an older browser that does not support the video element at all, you need to provide an alternative.

1. Add an h1 element with a link to the “BigBuck.mp4” file after the second source element as follows (highlighted in red).

<video controls poster='poster320.jpg' width='320' height='180'>

<source src='videos/BigBuck.mp4' type='video/mp4;

codecs="avc1.42E01E, mp4a.40.2"'>

<source src='videos/BigBuck.ogg' type='video/ogg; codecs="theora,

vorbis"'>

<h1><a href="videos/BigBuck.mp4">Download the video</a></h1>

</video>

You will not be able to test this in your current browser because of course it supports the existing video, but if you were to load the Web page in an older browser that does not support HTML5, you would see a simple link “Download the Video” and the user would have to click on the link and manually download and play the video.

1. Save your file.

## Review

### Questions

1. Why is it necessary to prevent the default behavior and return false for many of the drag-and-drop callback functions?
2. What is the role of the setDragImage function and what is its main drawback in Internet Explorer?
3. Name two attributes that you can add to the <video>element that can affect how video appears or works on your page.

### Answers

1. The default behavior on many browsers is to prevent drag-and-drop and display a visual indicator to the user that not all containers are drop targets. Preventing the default behavior shows a separate visual indicator and enables drag-and-drop functionality on certain browsers.
2. The setDragImage function allows single element to be used as a visual indicator. You can use the canvas element to show text, vector graphics, and multiple images together as a visual indicator. This feature is not currently supported in Internet Explorer 9 (and various other browsers).
3. The controls attribute allows you to specify whether or not you want controls visible on your page. The autoplay attribute allows you to specify whether or not the video should be playing upon launch of the page. There are additional attributes as well such as width, height, loop, poster, preload and audio.