Steven Holzner

Sams Teach Yourself HTML5 in 10 Minutes

SAMS

Sams Teach Yourself HTML5 in 10 Minutes

International Standard Book Number-10: 0-672-33333-3

International Standard Book Number-13: 978-0-672-33333-0

Library of Congress Cataloging-in-Publication Data

Holzner, Steven.

Sams teach yourself HTML5 in 10 minutes / Steven Holzner.

p. cm.

ISBN 978-0-672-33333-0 (pbk.)

1. HTML (Document markup language) I. Title. II. Title: Teach yourself HTML5 in 10 minutes.

QA76.76.H94H647 2011

006.7'4—dc22

2010045971

Printed in the United States of America

First Printing: December 2010

13 12 11 10 4 3 2 1

Editor In Chief
Mark Taub

Aquisitions Editor
Mark Taber

Development Editor
Songlin Qiu

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Barbara Hacha

Indexer
Heather McNeill

Proofreader
Debbie Williams

Publishing Coordinator
Vanessa Evans

Composition
Mark Shirar

Book Designer
Gary Adair

Table of Contents

This page intentionally left blank

Introduction

Welcome to HTML5, the new edition of HTML.

Many people are saying that it's about time for HTML5—HTML 4.01 was completed in 1999. Others are saying that what HTML5 offers is just too good to pass up. We hope you'll agree with both opinions.

HTML5 goes beyond all previous versions of HTML in scope and power. In fact, its biggest additions are in the scripting realm, not in the traditional realm of HTML elements at all. So if you're expecting just a list of new HTML elements, this book may surprise you. HTML has learned about JavaScript, and puts it to work extensively.

For example, HTML5 supports drag and drop, but you've got to use a scripting language like JavaScript to make it work. HTML5 also supports a Canvas control in which you can draw—using JavaScript. There are many more such areas that we'll see come alive in the new HTML.

What's in This Book

This book gives you a guided tour of the new features of HTML. We assume you know the previous version of HTML—HTML 4.01—well enough so that we can discuss only what's new in version 5. Here are the stops on your guided tour:

- Lesson 1, "Essential HTML5"—In this lesson, you'll get an overview of HTML5, as well as learning the rules for constructing an HTML5 document.
- Lesson 2, "Drawing with the Canvas Element"—Here you'll learn how to use JavaScript to draw in HTML5's new Canvas element.
- Lesson 3, "Dragging and Dropping with HTML5"—This lesson shows how to make items in Web pages "draggable" with the mouse.

- Lesson 4, “Web Form Controls”—HTML5 includes new controls (controls are elements such as radio buttons or check boxes that the user interacts with), including new telephone and datetime controls. We’ll put them to work here.
- Lesson 5, “Inline Editing”—With HTML5, you can edit the text contents of elements such as or interactively, and we’ll see how here.
- Lesson 6, “Working With Browser History”—In this lesson, we take a look at the built-in support in HTML for navigating the browser through its history, revisiting pages it has already been to.
- Lesson 7, “Getting the Point Across with Messaging”—HTML5 lets you send messages from one document to another, and we’ll get a glimpse into how that works here, by sending messages from one document to another that appears in an <iframe> in the first document.
- Lesson 8, “Using Video and Audio”—Some of the most exciting aspects of HTML5 are the <video> and <audio> elements. We’ll see how to play videos and audio using them in this lesson.
- Lesson 9, “Web Storage”—One thing web page authors have missed with traditional HTML and JavaScript is some place to store data between page accesses by the user. HTML5 gives you a couple of options that we’ll take a look at in this lesson.
- Lesson 10, “The New HTML5 Elements”—HTML5 comes with many new elements in addition to the ones we’ve already covered in the book, and we’ll see them here.

This page intentionally left blank

LESSON 3

Dragging and Dropping with HTML5

HTML5 supports drag-and-drop operations, where you can move elements and text around the browser window using a mouse or other pointing device.

That's useful for such operations as letting the user move items into a shopping cart, or letting them customize what elements appear in their home page, and it's a very popular part of HTML5.

Drag and drop is supported by a number of attributes added to HTML5 elements, such as the draggable attribute, which you set to true to make the element draggable. However, you do most of the work supporting drag and drop yourself, in a scripting language, such as JavaScript, as you'll see.

Let's jump into drag and drop operations immediately.

Welcome to Drag and Drop

From the point of view of HTML5 elements, drag and drop is pretty simple, involving these element attributes:

- Required attributes: draggable, ondragenter, ondragover, ondrop, ondragstart, ondragend
- Supported browsers: Chrome, Firefox, Opera, Safari

The real story takes place in scripting languages such as JavaScript, as you'll see. You connect each of the "on" attributes, such as ondragstart, to a JavaScript function like this for ondragstart, which occurs when the user starts dragging a draggable element:

```
ondragstart = "return start(event)";
```

It's up to you to write the code for the JavaScript function you connect to each of the "on" attributes.

TIP: Note that all the "on" attributes start with "ondrag" with one exception—ondrop, which occurs when you drop a dragged item. It's worth bearing in mind that this attribute is ondrop, not ondrag-drop, or you're going to confuse some browsers, which will not run your code.

In this lesson, we'll create the drag-and-drop example, draganddrop.html, you see in Figures 3.1 and 3.2. There are three elements that you can drag around, labeled 1, 2, and 3. We've set up the example so that not all elements can be dropped on the large square targets in the page. For example, if you try to drop 1 onto the second target, you'll just get a "no" symbol, as shown in Figure 3.1, that indicates that target won't accept 1. On the other hand, you can drop 1 onto the third target, as shown in Figure 3.2.

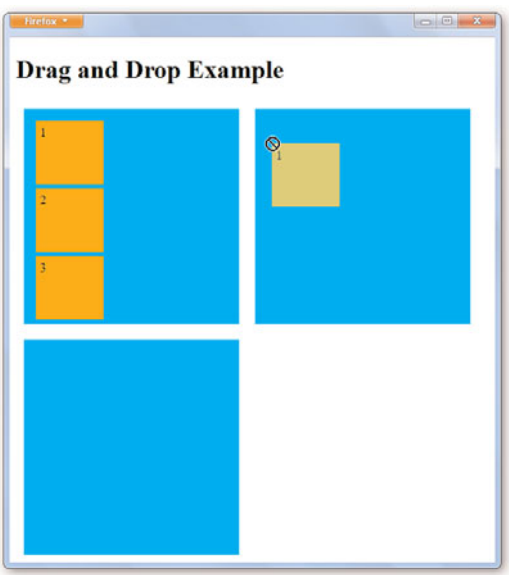

FIGURE 3.1 Denying a drag-and-drop operation.

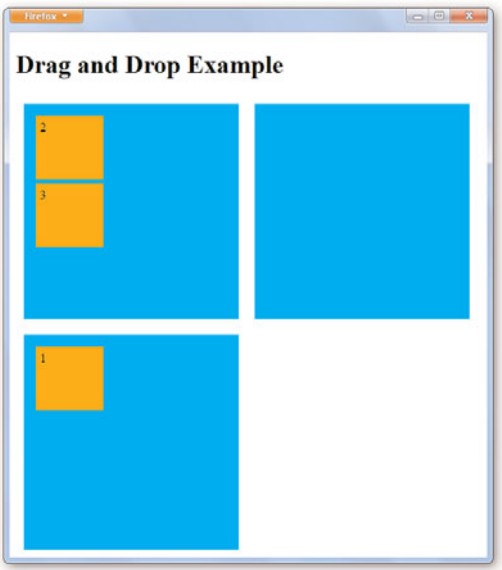

FIGURE 3.2 Allowing a drag-and-drop operation.

Now let's take a look at the draggable attribute and the "on" attributes and how you use them to support drag and drop.

Getting to Know the Drag-and-Drop API

You can read all about the drag-and-drop specification according to the W3C at: http://dev.w3.org/html5/spec/dnd.html.

From an HTML point of view, drag and drop is supported with these attributes:

- draggable
- ondragenter
- ondragover
- ondrop

- ondragstart
- ondragend

The draggable attribute of an element, as you might guess, is set to true if you want to allow that element to be dragged. The "on" attributes are used to connect JavaScript functions to various events. For example, you use ondragenter to call a JavaScript function when a draggable element is being dragged over another element (and in the JavaScript function, you can indicate to the browser whether you can drop the draggable item there).

Let's take a look at each of these attributes briefly; then we'll put them to work in the draganddrop.html example.

The draggable Attribute

The draggable attribute is the most basic of all drag-and-drop attributes. To make an element draggable, you set its draggable attribute to true:

```
<div id="draggable3" draggable="true">
</div>
```

Doing so informs the browser that this element can be dragged, but setting this attribute isn't all that's needed—you also have to connect up JavaScript functions to the "on" attributes to make this work.

The ondragenter Attribute

Drag enter events occur in a drop target when the user drags a draggable element over that target.

You can connect this event to a JavaScript handler function (which it's up to you to write) like this:

```
<div id="target1"
  ondragenter="return enter(event)"
    .
    .
    .
```

Note that this event occurs in drop targets, not in draggable elements.

The ondragover Attribute

Dragover events occur in a drop target while users drag a draggable element over that target. You can connect this event to a JavaScript handler function like this:

```
<div id="target1"
  ondragenter="return enter(event)"
  ondragover="return over(event)"
    .
    .
    .
```

This event occurs in drop targets.

The ondrop Attribute

Drop events occur in a drop target while users drop a draggable element onto that target. You can connect this event to a JavaScript handler function like this:

```
<div id="target1"
  ondragenter="return enter(event)"
  ondragover="return over(event)"
  ondrop="return drop(event)">
```

This event occurs in drop targets; note that it's ondrop, not ondragdrop!

The ondragstart Attribute

This event occurs in draggable elements when users start dragging them. You can connect JavaScript function handlers to this event like this:

```
<div id="draggable1" draggable="true"
  ondragstart="return start(event)"
  .
  .
  .
```

This event occurs in draggable elements.

The ondragend Attribute

This event occurs in draggable elements when users stop dragging them. You can connect JavaScript function handlers to this event like this:

```
<div id="draggable1" draggable="true"
  ondragstart="return start(event)"
  ondragend="return end(event)">1
  .
  .
  .
```

This event occurs in draggable elements.

The dataTransfer Object

There is one more item you should know about—the dataTransfer that comes built in to event objects in HTML5—because it offers support for drag-and-drop operations. You access this object through the event object passed to you when drag-and-drop operations start.

For example, the dataTransfer object has a property named effectAllowed that lets you specify what drag-and-drop operation is allowed. It has functions named setData() and getData() to allow you to specify what data you want to drag and drop with a draggable element, and another function named setDragImage() lets you specify the image of the item being dragged.

Here's how using dataTransfer in JavaScript might work, where we're specifying that move operations are OK, storing the ID of the draggable element so we know what element to move when the drag operation is complete, and setting the image that the user drags to be a copy of the draggable element that the mouse clicked (as given by the event object e's target attribute):

```
e.dataTransfer.effectAllowed='move';
e.dataTransfer.setData("Data",
  e.target.getAttribute('id'));
e.dataTransfer.setDragImage(e.target, 0, 0);
```

To make this clear, let's see all this at work in an example.

Starting the Drag-and-Drop Example

To show how to put drag and drop to work, we're going to create an example named draganddrop.html, which you can see running in Figures 3.1 and 3.2, and whose code appears in its entirety at the end of this lesson.

To get started with the draganddrop.html example, follow these steps:

1. Create draganddrop.html using a text editor such as Windows WordPad.

2. Enter the following code to create the three targets onto which draggable elements can be dropped. Note that we will use elements for the targets and that we connect the drag-and-drop events that targets support to JavaScript functions that we will write later.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>
      Drag and Drop Example
    </title>
  </head>

  <body>
    <h1>Drag and Drop Example</h1>

    <div id="target1"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">
    </div>

    <div id="target2"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">
    </div>

    <div id="target3"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
```

```
      ondrop="return drop(event)">
    </div>
  </body>
</html>
```

3. Add the following code to create the three draggable elements as children of the first target. Note that we set each draggable element's draggable attribute to true and also connect the events that draggables support to JavaScript functions, which we will write later.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>
      Drag and Drop Example
    </title>
  </head>

  <body>
    <h1>Drag and Drop Example</h1>

    <div id="target1"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">

      <div id="draggable1" draggable="true"
        ondragstart="return start(event)"
        ondragend="return end(event)">1
      </div>

      <div id="draggable2" draggable="true"
        ondragstart="return start(event)"
        ondragend="return end(event)">2
      </div>

      <div id="draggable3" draggable="true"
        ondragstart="return start(event)"
        ondragend="return end(event)">3
      </div>
    </div>

    <div id="target2"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">
    </div>
```

```
    <div id="target3"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">
    </div>
  </body>
</html>
```

4. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now we've got our example started with the three targets and three draggable elements. All that is invisible so far, however, so we will style them next.

Styling the Draggable and Target Elements

In this task, we'll make the elements we use for the targets and draggables visible. In particular, we'll style the targets in cyan and the draggables in orange.

To do so, follow these steps:

1. Open draganddrop.html using a text editor such as Windows WordPad.
2. Add the following code to style the draggable elements and the target elements, as well as give them a size.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>
      Drag and Drop Example
    </title>

    <style type="text/css">
      #target1, #target2, #target3
      {
```

```
      float:left; width:250px; height:250px;
      padding:10px; margin:10px;
    }

    #draggable1, #draggable2, #draggable3
    {
      width:75px; height:70px; padding:5px;
      margin:5px;
    }

    #target1 {background-color: cyan;}
    #target2 {background-color: cyan;}
    #target3 {background-color: cyan;}

    #draggable1 {background-color: orange;}
    #draggable2 {background-color: orange;}
    #draggable3 {background-color: orange;}
  </style>

</head>

<body>
  <h1>Drag and Drop Example</h1>

  <div id="target1"
    ondragenter="return enter(event)"
    ondragover="return over(event)"
    ondrop="return drop(event)">

    <div id="draggable1" draggable="true"
      ondragstart="return start(event)"
      ondragend="return end(event)">1
    </div>

    <div id="draggable2" draggable="true"
      ondragstart="return start(event)"
      ondragend="return end(event)">2
    </div>

    <div id="draggable3" draggable="true"
      ondragstart="return start(event)"
      ondragend="return end(event)">3
    </div>
  </div>

  <div id="target2"
    ondragenter="return enter(event)"
```

```
        ondragover="return over(event)"
        ondrop="return drop(event)">
      </div>

      <div id="target3"
        ondragenter="return enter(event)"
        ondragover="return over(event)"
        ondrop="return drop(event)">
      </div>
    </body>
  </html>
```

3. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now you can see the draggables and the targets as shown in Figure 3.3.

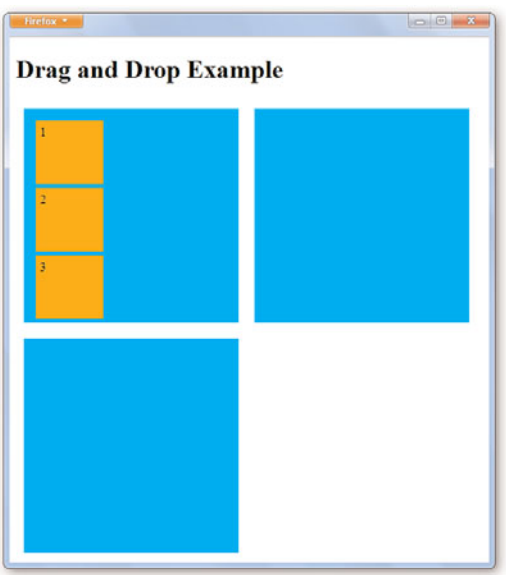

FIGURE 3.3 The draggables and targets in draganddrop.html.

Starting the Drag Operation

When the user starts dragging a draggable element in our example, that element's ondragstart event occurs, and we've tied that event to a JavaScript function named start().

In this task, we'll write the start() function to get the dragging operation started. That involves three steps: setting the allowed drag operation to "move" so the draggable element that the user wants to drag may be dragged, storing the ID of the element that's being dragged so we can move it when it's dropped, and setting the image that the user will drag around.

To do all these things, follow these steps:

1. Open draganddrop.html using a text editor such as Windows WordPad.
2. Add the following code to the <head> section of dragdrop.html, starting a new <script> element, and creating the start() function:

```
<script type="text/javascript">
  function start(e)
  {
    .
    .
    .
  }
```

3. Add the following code to the start() function to indicate that the draggable element the user is attempting to drag may indeed be moved (which you do by setting the dataTransfer.effectAllowed property of the event object passed to the start() function to "move"):

```
<script type="text/javascript">
  function start(e)
  {
    e.dataTransfer.effectAllowed='move';
    .
    .
    .
  }
```

4. Add the following code to the start() function to store the ID of the element being dragged so we can move it when it's dropped:

```
<script type="text/javascript">
  function start(e)
  {
    e.dataTransfer.effectAllowed='move';
    e.dataTransfer.setData("Data",
      e.target.getAttribute('id'));
    .
    .
    .
  }
```

5. Add the following code to the start() function to set the drag image to the draggable element, with an offset of (0, 0):

```
<script type="text/javascript">
  function start(e)
  {
    e.dataTransfer.effectAllowed='move';
    e.dataTransfer.setData("Data",
      e.target.getAttribute('id'));
    e.dataTransfer.setDragImage(e.target, 0, 0);
    return true;
  }
```

6. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now the user will be able to drag the draggable elements in this example.

Allowing Dragged Objects to Enter the Targets

When the user drags a draggable element to a target element, the target element's ondragEnter event occurs. We've tied that event to a JavaScript function named enter(), and in that function, we want to indicate that draggable objects are allowed to enter the target by returning a value of true from the enter() function.

To do that, follow these steps:

1. Open draganddrop.html using a text editor such as Windows WordPad.
2. Add the following code to the <script> section of dragdrop.html, creating the enter() function and returning a value of true from it, indicating that draggable elements may enter a target:

```
function enter(e)
{
  return true;
}
```

3. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now the user will be able to drag the draggable elements to the targets.

Allowing Dragged Objects to Be Dropped on Certain Targets

When the user drags a draggable element over a target, that target's ondragover event occurs, and we've tied that event to a function named over(). You can use the over() function to indicate whether the dragged item may be dropped on the current target. If you return a value of true from this function, the dragged item may not be dropped; returning a value of false means that it can be dropped.

To create the over() function, follow these steps:

1. Open draganddrop.html using a text editor such as Windows WordPad.
2. Add the following code to the <script> section of dragdrop.html, creating the over() function and getting the ID of the dragged item (iddraggable) and the ID of the target (id):

```
function over(e)
{
  var iddraggable =
```

```
    e.dataTransfer.getData("Data");
  var id = e.target.getAttribute('id');
  .
  .
  .
}
```

3. Add the following code to the over() function to indicate that any dragged item may be dropped on target 1, that draggable element 3 may be dropped on target 2 only, and that draggable elements 1 and 2 may be dropped on target 3 only:

```
function over(e)
{
  var iddraggable =
    e.dataTransfer.getData("Data");
  var id = e.target.getAttribute('id');

  if(id =='target1')
    return false;

  if((id =='target2')
    && iddraggable == 'draggable3')
    return false;

  else if(id =='target3'
    && (iddraggable == 'draggable1' ¦¦
    iddraggable =='draggable2'))
    return false;

  else
    return true;
}
```

4. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now you've indicated to the browser which draggable elements may be dropped on which target elements.

Handling Drop Events

When the user drops a draggable element on an allowed target element, how do we move the draggable to the target? That turns out to be simple—we'll just use the built-in JavaScript function appendChild to append the draggable element to the current target element.

When the user drops a draggable element on a target, the ondrop event occurs in the target element, and we have connected a JavaScript function named drop() to implement the drop operation. To add drop() to the draganddrop.html example, follow these steps:

1. Open draganddrop.html using a text editor such as Windows WordPad.

2. Add the following code to the <script> section of dragdrop.html, creating the drop() function and getting the ID of the dragged item (iddraggable):

```
function drop(e)
{
  var iddraggable =
    e.dataTransfer.getData("Data");
  .
  .
  .
}
```

3. Add the following code to the drop() function to append the draggable element to the target element, as well as stopping further propagation of the event in the browser with the stopPropagation() function (returning a value of false also stops further propagation of the event):

```
function drop(e)
{
  var iddraggable =
    e.dataTransfer.getData("Data");
  e.target.appendChild
    (document.getElementById(iddraggable));
  e.stopPropagation();
  return false;
}
```

4. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now you've handled the drop operation.

Ending Drop Operations

When a draggable element is dropped, its ondragEnd event occurs, and we've tied that event to the JavaScript function end(). We'll add code to the end() function to clear the data stored in the dataTransfer object (that is, the ID of the element being dragged) now that the drop operation is finished. Just follow these steps:

1. Open draganddrop.html using a text editor such as Windows WordPad.
2. Add the following code to the <script> section of dragdrop.html, creating the end() function and then ending the <script> section in draganddrop.html:

```
  function end(e)
  {
    e.dataTransfer.clearData("Data");
    return true
  }
</script>
```

3. Save draganddrop.html. Make sure you save this code in text format (the default format for WordPad, for example, is RTF, rich-text format, which won't work with browsers).

Now you've completed the draganddrop.html example and can drag and drop using any supported browser, as shown in Figure 3.4.

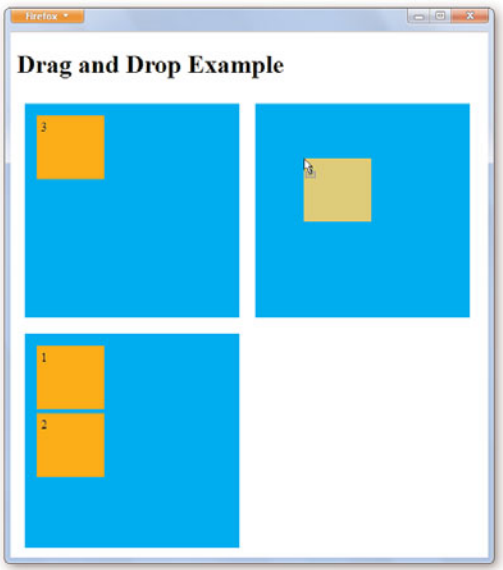

FIGURE 3.4 Dragging and dropping with draganddrop.html.

The draganddrop.html Example Code

Here's the full code of the draganddrop.html example that we developed in this lesson for reference:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>
      Drag and Drop Example
    </title>

    <style type="text/css">
      #target1, #target2, #target3
      {
        float:left; width:250px; height:250px;
        padding:10px; margin:10px;
      }
```

```
  #draggable1, #draggable2, #draggable3
  {
    width:75px; height:70px; padding:5px;
    margin:5px;
  }

  #target1 {background-color: cyan;}
  #target2 {background-color: cyan;}
  #target3 {background-color: cyan;}

  #draggable1 {background-color: orange;}
  #draggable2 {background-color: orange;}
  #draggable3 {background-color: orange;}
</style>

<script type="text/javascript">
  function start(e)
  {
    e.dataTransfer.effectAllowed='move';
    e.dataTransfer.setData("Data",
      e.target.getAttribute('id'));
    e.dataTransfer.setDragImage(e.target, 0, 0);
    return true;
  }

  function enter(e)
  {
    return true;
  }

  function over(e)
  {
    var iddraggable =
      e.dataTransfer.getData("Data");
    var id = e.target.getAttribute('id');

    if(id =='target1')
      return false;

    if((id =='target2')
      && iddraggable == 'draggable3')
      return false;

    else if(id =='target3'
      && (iddraggable == 'draggable1' ¦¦
```

```
          iddraggable =='draggable2'))
          return false;

        else
          return true;
      }

      function drop(e)
      {
        var iddraggable =
          e.dataTransfer.getData("Data");
        e.target.appendChild
          (document.getElementById(iddraggable));
        e.stopPropagation();
        return false;
      }

      function end(e)
      {
        e.dataTransfer.clearData("Data");
        return true
      }
    </script>
  </head>

  <body>
    <h1>Drag and Drop Example</h1>

    <div id="target1"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">

      <div id="draggable1" draggable="true"
        ondragstart="return start(event)"
        ondragend="return end(event)">1
      </div>

      <div id="draggable2" draggable="true"
        ondragstart="return start(event)"
        ondragend="return end(event)">2
      </div>

      <div id="draggable3" draggable="true"
        ondragstart="return start(event)"
        ondragend="return end(event)">3
```

```
      </div>
    </div>

    <div id="target2"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">
    </div>

    <div id="target3"
      ondragenter="return enter(event)"
      ondragover="return over(event)"
      ondrop="return drop(event)">
    </div>
  </body>
</html>
```

This page intentionally left blank

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

S

T

U

V

W-Z