

Steven Holzner

Sams **Teach Yourself**

HTML5

in **10**
Minutes

SAMS

Sams Teach Yourself HTML5 in 10 Minutes

Copyright © 2011 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

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

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales

international@pearsoned.com

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

Introduction	1
What's in This Book	1
What You Need	3
 1 Essential HTML5	 5
Welcome to HTML5	5
Drawing With the Canvas Element	6
Dragging and Dropping	7
Getting Data With the New Web Form Controls	7
Edit Web Pages on the Fly	8
Remembering With Browser History	8
Saying Hello With Interdocument Messaging	8
Awesome Audio and Video	9
Making Use of Web Storage	9
Using the New Elements	10
 2 Drawing with the Canvas Element	 13
Welcome to the Canvas Element	13
Getting to Know the Canvas API	14
Starting the Canvas Example	18
Drawing Rectangles	20
Drawing Line Art	22
Filling Line Art	24
Drawing with Bezier Curves	25
Drawing with Quadratic Curves	27
Drawing Arcs	28
Drawing Text	30
The canvas.html Example Code	31

3 Dragging and Dropping with HTML5	35
Welcome to Drag and Drop	35
Getting to Know the Drag-and-Drop API	37
Starting the Drag-and-Drop Example	41
Styling the Draggable and Target Elements	43
Starting the Drag Operation	46
Allowing Dragged Objects to Enter the Targets	47
Allowing Dragged Objects to Be Dropped on Certain Targets	48
Handling Drop Events	50
Ending Drop Operations	51
The draganddrop.html Example Code	52
 4 Web Form Controls	 57
Welcome to Web Form Controls	58
Getting to Know the Web Form Controls API	60
Starting the Web Forms Example	66
Creating a Default Control	67
Creating a URL Control	68
Creating an Email Control	69
Creating Range and Number Controls	70
Creating Date and Time Controls	72
Creating a Color Control	74
Creating a Search Control	75
The webforms.html Example Code	76
The webforms.php Example Code	78
 5 Inline Editing	 79
Welcome to Inline Editing	79
Starting the editdiv.html Example	81
Adding a Bold Button	83
Adding an Italic Button	85
Adding an Underline Button	87
Adding an Add Link Button	88

Adding a Display Source Button	91
Spellchecking	93
The editdiv.html Example Code	95
Starting the editiframe.html Example	96
Adding the editiframe.html Buttons	98
The editiframe.html Example Code	100
6 Working with Browser History	103
Welcome to Browser History	103
Getting to Know the History API	104
Starting the pophistory.html Example	106
Adding a Back Button	107
Adding a Forward Button	110
Adding a Go Button	112
Getting History Length	114
Pushing Data into the History	116
Popping Data from the History	119
The pophistory.html Example Code	121
7 Getting the Point Across with Messaging	125
Welcome to Messaging	125
Getting to Know the Messaging API	127
Starting the parent.html Example	129
Sending a Cross-Window Message	130
Starting the child.html Example	132
Receiving a Cross-Window Message	134
The parent.html Example Code	135
The child.html Example Code	136
Starting the domainparent.html Example	137
Sending a Cross-Domain Message	138
Starting the domainchild.html Example	140
Receiving a Cross-Domain Message	142

The domainparent.html Example Code	143
The domainchild.html Example Code	144
8 Using Video and Audio	147
Welcome to the Video Media Control	147
Getting to Know the Video Element API	148
Converting to OGG Format	150
Starting the video.html Example	153
Adding Controls to the video.html Example	155
Looping a Video	156
Playing a Video Automatically	156
Detecting When a Video Has Failed	157
Welcome to the Audio Media Control	160
Getting to Know the Audio Element API	160
Starting the audio.html Example	162
Detecting When an Audio Has Failed	164
9 Web Storage	167
Welcome to Session Storage	167
Getting to Know the Session Storage API	169
Starting the sessionstorage.html Example	171
Storing Data in the Session	172
Getting Data from the Session	174
Clearing Session Data	175
The sessionstorage.html Code	177
Welcome to Local Storage	178
Getting to Know the Local Storage API	180
Starting the localStorage.html Example	181
Storing Data in the Browser	182
Getting Data from the Browser	184
Clearing Local Data	186
The localStorage.html Code	188

10 The New HTML5 Elements	191
Adding SVG and MathML	191
Welcome to the New Elements	192
The <article> Element	194
The <aside> Element	195
The <audio> Element	196
The <canvas> Element	196
The <command> Element	196
The <datalist> Element	198
The <details> Element	198
The <embed> Element	199
The <figcaption> Element	200
The <figure> Element	201
The <footer> Element	202
The <header> Element	202
The <hgroup> Element	204
The <keygen> Element	204
The <mark> Element	205
The <meter> Element	206
The <nav> Element	207
The <output> Element	208
The <progress> Element	209
The <rp> Element	211
The <rt> Element	211
The <ruby> Element	212
The <section> Element	213
The <source> Element	214
The <summary> Element	215
The <time> Element	215
The <video> Element	216
Index	217

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 `<div>` 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 <div> elements that you can drag around, labeled 1, 2, and 3. We’ve set up the example so that not all <div> elements can be dropped on the large square targets in the page. For example, if you try to drop <div> 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 <div> 1. On the other hand, you can drop <div> 1 onto the third target, as shown in Figure 3.2.

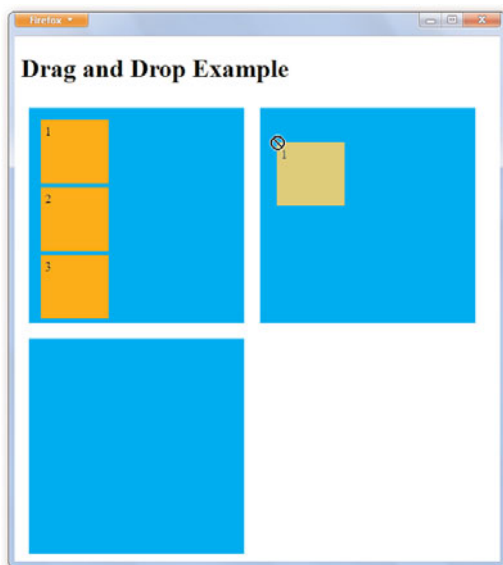


FIGURE 3.1 Denying a drag-and-drop operation.

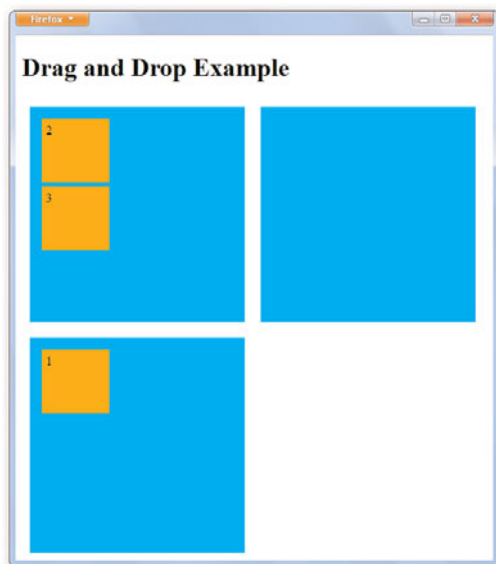


FIGURE 3.2 Allowing a drag-and-drop operation.

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 `<div>` 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 <div> elements as children of the first target. Note that we set each draggable <div> 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 `<div>` 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 `<div>` elements and the target `<div>` 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.

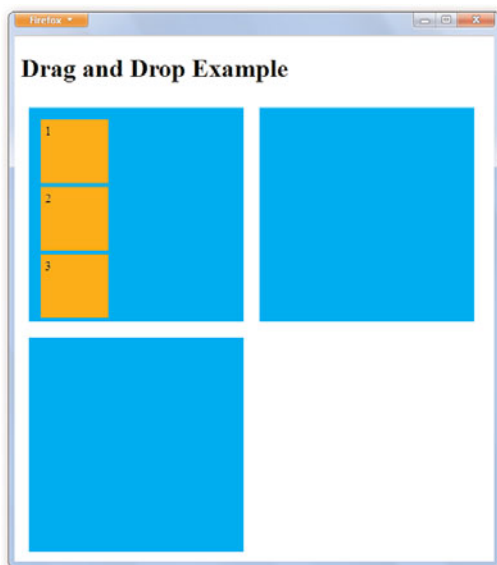


FIGURE 3.3 The draggables and targets in draganddrop.html.

Starting the Drag Operation

When the user starts dragging a draggable `<div>` element in our example, that `<div>` 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 `<div>` 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 `<div>` 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 `<div>` 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 `<div>` 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 `<div>` elements in this example.

Allowing Dragged Objects to Enter the Targets

When the user drags a draggable `<div>` element to a target `<div>` element, the target `<div>` 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 `<div>` elements to the targets.

Allowing Dragged Objects to Be Dropped on Certain Targets

When the user drags a draggable `<div>` 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 (`idraggable`) and the ID of the target (`id`):

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

```
    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 `<div>` element 3 may be dropped on target 2 only, and that draggable `<div>` elements 1 and 2 may be dropped on target 3 only:

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

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

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

    else if(id == 'target3'
        && (idraggable == 'draggable1' ||
            idraggable == '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 `<div>` elements may be dropped on which target `<div>` elements.

Handling Drop Events

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

When the user drops a draggable `<div>` 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 (`idraggable`):

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

3. Add the following code to the `drop()` function to append the draggable `<div>` element to the target `<div>` 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 idraggable =
        e.dataTransfer.getData("Data");
    e.target.appendChild
        (document.getElementById(idraggable));
    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 `<div>` 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.

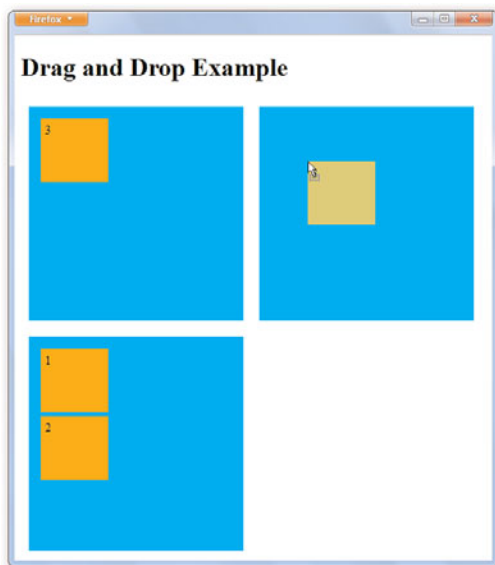


FIGURE 3.4 Dragging and dropping with draganddrop.html.

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;
      }
    </style>
  </head>
  <body>
    <div id="target1">1</div>
    <div id="target2">2</div>
    <div id="target3">3</div>
  </body>
</html>
```

```
#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

accept attribute, 61

alt attribute, 61

Amaya test browser, 192

annotations (text)

 <rp> element, 211

 <rt> element, 211-212

APIs (application programming interfaces), 14-15

 <audio> element, 160-162

 <canvas> element, 14-15

- complex shapes, 16-17*
- images, 17-18*
- line styles, 15*
- rectangles, 16*
- shadows, 16*
- styling, 15*
- text, 17*

transformations, 18

drag-and-drop, 37-40

form controls, 60-65

allowed attributes, 61-62

*built-in attributes/
 functions, 63*

history object, 104-106

local storage, 180-181

messaging, 128-129

data attribute, 128-129

onMessage() function, 128

origin attribute, 129

postMessage() function, 128

source attribute, 129

session storage, 169-188

specification website, 37

 <video> element, 148-150

arc() function, 16, 28

arcs, 28-29

arcTo() function, 16

<article> element, 194-195

<aside> element, 195-196

attributes

 <article> element, 194

 <aside> element, 195-196

 <audio> element, 160-161

 <canvas> element

- line styles, 15*
- shadows, 16*
- styling, 15*
- text, 17*

 <command> element, 196-197

contenteditable, 79-80

<datalist> element, 198

designmode, 80

<details> element, 198-199

drag-and-drop, 35, 37-38

draggable, 38

ondragend, 40

ondragenter, 38

ondragover, 39

ondragstart, 39

ondrop, 39

<embed> element, 199-200

<figcaption> element, 200-201

<figure> element, 201

<footer> element, 202

form controls, 61-62

allowed, 61-62

built-in, 63

<header> element, 202-203

<hgroup> element, 204

history object, 105

inline editing, 79

<keygen> element, 204-205

length, 114-116

<mark> element, 205-206

- messaging
 - data*, 128-129
 - origin*, 129
 - source*, 129
- <meter> element, 206-207
- <nav> element, 207-208
- <output> element, 208-209
- <progress> element, 209-210
- <rp> element, 211
- <rt> element, 211-212
- <ruby> element, 212-213
- <section> element, 213
- <source> element, 214
- spellcheck, 80
- <summary> element, 215
- <time> element, 215-216
- <video> element, 148-150
- <audio> element
 - API, 160-162
 - browser support, 161
 - error handling, 164-166
 - support, 9
 - W3C website, 161
- audio.html, 162-164
- autocomplete attribute, 61
- autoplay attribute
 - <audio> element, 161
 - <video> element, 149

B

-
- Back buttons, 107-110
 - back() function, 105
 - beginPath() function, 16, 22
 - bezier curves, 25-26
 - bezierCurveTo() function, 16, 25
 - bolding text, 83-85
 - browsers
 - Amaya test, 192
 - history
 - Back buttons*, 107-110
 - forward buttons*, 110-112
 - length*, 114-116

- onpopstate events*, 106
- overview*, 103-104
- popping data*, 119-121
- pushing data*, 116-119
- support, 5
 - audio support*, 161
 - drag and drop support*, 35
 - editable documents support*, 80
 - editable elements support*, 79-80
 - local storage support*, 181
 - MathML support*, 192
 - spell check support*, 80
 - SVG support*, 191
 - video support*, 149
- button controls, 65

C

-
- <canvas> element
 - API, 14-15
 - complex shapes*, 16-17
 - images*, 17-18
 - line styles*, 15
 - rectangles*, 16
 - shadows*, 16
 - styling*, 15
 - text*, 17
 - transformations*, 18
 - arcs, 28-29
 - bezier curves, 25-26
 - canvas.html example
 - Canvas*, creating, 18-19
 - code*, 31-33
 - JavaScript*, adding, 19-20
 - creating, 13
 - Firefox example, 14
 - line art
 - drawing*, 22-23
 - filling*, 24-25
 - overview, 6
 - quadratic curves, 27-28

- rectangles
 - drawing*, 20-21
 - functions*, 16
 - specification, 13
 - text, 30-31
- change events, 64
- checkbox controls, 65
- checked attribute, 61-63
- child.html
 - creating, 132-133
 - receiving messages, 134-135
- clear() function, 171
- clearRect() function, 16
- clip() function, 16
- closePath() function, 16, 22
- color controls
 - creating, 74-75
 - data types, 65
- <command> element, 196-197
- complex shapes, drawing, 16-17
- contenteditable attribute, 79-80
- controls
 - form
 - allowed attributes*, 61-62
 - API*, 60-65
 - built-in attributes/ functions*, 63
 - color*, 74-75
 - creating*, 57
 - data extraction*, 65
 - data types*, 65
 - date and time*, 72-73
 - default, creating*, 67-68
 - email*, 69-70
 - events*, 64
 - new*, 7
 - number*, 70-71
 - range*, 70-71
 - search*, 75-76
 - text*, 67-68
 - URL*, 68-69
 - W3C specification*, 57
 - webforms.html example*, 66, 76-78
 - webforms.php example*, 78
 - <video> element, 155
- controls attribute
 - <audio> element, 161
 - <video> element, 149
- cross-domain messaging, 126-127
 - domainchild.html, 140-141
 - code*, 144-145
 - receiving messages*, 142-143
 - domainparent.html
 - code*, 143-144
 - creating*, 137-138
 - receiving, 142-143
 - sending, 138-140
- cross-window messaging, 126
 - child.html
 - creating*, 132-133
 - receiving messages*, 134-135
 - parent.html
 - code*, 135-136
 - creating*, 129-130
 - sending message to child.html*, 130-132
 - receiving, 134-135
 - sending, 130-132
- curves
 - bezier, 25-26
 - quadratic, 27-28

D

- data attribute, 128-129
- <datalist> element, 198
- dataTransfer object, 40
- data types (form controls), 65
- date and time controls

- creating, 72-73
- data types, 65
- dates, 65, 72-73
- times, 65, 72-73
- datetime control, 72-73
- default controls, creating, 67-68
- designmode attribute, 80
- <details> element, 198-199
- display sources, adding, 91-93
- <div> elements
 - drag and drop example, 36
 - inline editing example, 82
- documents (editable)
 - attributes, 79
 - contenteditable*, 79-80
 - designmode*, 80
 - spellcheck*, 80
 - links, 88-90
 - making, 80
 - spell checking, 93-94
 - text
 - bolding*, 83-85
 - italicizing*, 85-86
 - underlining*, 87-88
- domainchild.html, 140-141
 - code, 144-145
 - receiving messages, 142-143
- domainparent.html
 - code, 143-144
 - creating, 137-138
- draggable attribute, 38
- dragging and dropping, 7
 - API, 37-40
 - attributes, 35, 37-38
 - draggable*, 38
 - ondragend*, 40
 - ondragenter*, 38
 - ondragover*, 39
 - ondragstart*, 39
 - ondrop*, 39
 - browser support, 35
 - dataTransfer object, 40
 - <div> elements, 36
 - draganddrop.html example
 - code*, 52-55
 - draggable elements*,
 - creating*, 42-43
 - targets*, *creating*, 41-42
 - dragging elements
 - starting*, 46-47
 - styling*, 43-45
 - target entrance*, *allowing*, 47-48
 - dropping elements
 - allowing*, 48-49
 - drop events*, *handling*, 50
 - ending*, 51
 - JavaScript functions,
 - connecting*, 35-36
 - targets
 - creating*, 41-42
 - styling*, 43-45
- drawImage() function, 17
- drawing
 - arcs, 28-29
 - complex shapes, 16-17
 - curves
 - bezier*, 25-26
 - quadratic*, 27-28
 - hearts, 25-26
 - images, 17-18
 - line art, 22-23
 - rectangles, 16, 20-21
 - text, 30-31
 - triangles
 - green triangle example*, 24-25
 - three triangles example*, 22-23
- drop() function, 50
- dropped elements (HTML5)

dropping elements

- allowing, 48-49
- ending, 51
- handling, 50

E

editdiv.html example

- bolding, 83-85
- code, 95-96
- display sources, 91-93
- <div> element, creating, 82
- italicizing, 85-86
- links, 88-90
- underlining, 87-88

editiframe.html example

- buttons, adding, 98-100
- code, 100-101
- iframe, creating, 97-98

editing

- attributes, 79
 - contenteditable*, 79-80
 - designmode*, 80
 - spellcheck*, 80
- text, 8
 - bolding*, 83-85
 - display sources*, 91-93
 - italicizing*, 85-86
 - links*, 88-90
 - spell checking*, 91-94
 - underlining*, 87-88

elements

- <audio>
 - API, 160-162
 - browser support, 161
 - error handling, 164-166
 - support, 9
 - W3C website, 161
- <canvas>
 - API, 14-15
 - arcs, 28-29

bezier curves, 25-26

canvas.html example,
18-20, 31-33

complex shapes, 16-17

creating, 13

Firefox example, 14

images API, 17-18

line art, drawing, 22-23

line art, filling, 24-25

line styles, 15

overview, 6

quadratic curves, 27-28

rectangle functions, 16

rectangles, drawing, 20-21

shadows, 16

specification, 13

styling, 15

text, 17, 30-31

transformations API, 18

<div>

- drag and drop example*, 36
- inline editing example*, 82

dropped, 11, 193

editable

- bolding text*, 83-85
- italicizing text*, 85-86
- links*, 88-90
- making*, 79-80
- spell checking*, 93-94
- underlining text*, 87-88

new

- <article>, 194-195
- <aside>, 195-196
- <audio>. *See* <audio>
element
- <canvas>. *See* <canvas>
element
- <command>, 196-197
- <datalist>, 198
- <details>, 198-199
- <embed>, 199-200
- <figcaption>, 200-201

- `<figure>`, 201-202
- `<footer>`, 202
- `<header>`, 202-203
- `<hgroup>`, 204
- `<keygen>`, 204-205
- listing of*, 10-11, 192-193
- `<mark>`, 205-206
- `<meter>`, 206-207
- `<nav>`, 207-208
- `<output>`, 208-209
- `<progress>`, 209-210
- `<rp>`, 211
- `<rt>`, 211-212
- `<ruby>`, 212-213
- `<section>`, 213
- `<source>`, 214
- `<summary>`, 215
- `<time>`, 215-216
- `<video>`
 - API*, 148-150
 - controls, adding*, 155
 - looping*, 156
 - OGG conversions*, 150-152
 - playing automatically*, 156-157
- email controls
 - creating, 69-70
 - data types, 65
- `<embed>` element, 199-200
- `end()` function, 51
- `enter()` function, 48
- error handling
 - `<audio>` element, 164-166
 - `<video>` element, 157-159
- events
 - control, 64
 - drop events, 50
 - input, 64
 - onpopstate, 106, 119-121
- `execCommand()` function, 83

F

- `fail()` function, 157
- `<figcaption>` element, 200-201
- `<figure>` element, 201-202
- file controls, 65
- file converters, 151
- files attribute, 63
- `fill()` function, 16, 24
- filling
 - bezier curves, 25-26
 - line art, 24-25
 - rectangles, 20-21
- `fillRect()` function, 16, 20
- `fillStyle` attribute
 - `<canvas>` element, 15
 - line art, 24
 - rectangles, 20
- `fillText()` function, 17
- font attribute, 17, 30
- `<footer>` element, 202
- formation attribute, 61
- formatting text
 - bolding, 83-85
 - italicizing, 85-86
 - links, 88-90
 - underlining, 87-88
- form controls
 - API, 60-65
 - allowed attributes*, 61-62
 - built-in attributes/functions*, 63
 - data types*, 65
 - events*, 64
 - color, 74-75
 - creating, 57
 - data extraction, 65
 - date and time, 72-73
 - default, creating, 67-68
 - email, 69-70
 - new, 7
 - number, 70-71
 - range, 70-71

- search, 75-76
- text, 67-68
- URL, 68-69
 - creating*, 68-69
 - data types*, 65
- W3C specification, 57
- webforms.html example
 - code*, 76-78
 - HTML table, creating*, 66
- webforms.php example, 78
- formenctype attribute, 61
- formmethod attribute, 61
- formnovalidate attribute, 61
- formtarget attribute, 61
- Forward buttons, 110-112
- forward() function, 105
- functions
 - <canvas> element
 - complex shapes*, 16-17
 - images, drawing*, 17-18
 - rectangles*, 16
 - text*, 17
 - transformations*, 18
 - drop(), 50
 - end(), 51
 - enter(), 48
 - execCommand(), 83
 - fail(), 157
 - form controls, 61-62
 - getData(), 40
 - history object
 - back()*, 105, 107-110
 - forward()*, 105, 110-112
 - go()*, 105, 112-114
 - pushState()*, 106, 116-118
 - replaceState()*, 106
 - messaging
 - onMessage()*, 128
 - postMessage()*, 128
 - send()*, 130, 138-140

- over(), 48-49
- select(), 63
- setData(), 40
- setDragImage(), 40
- showSource(), 91
- start(), 46
- updateBar(), 210

G

- getData() function, 40
- getItem() function, 170
- go() function, 105, 112-114
- green triangle example, 24-25

H

handling

- drop events, 50
- errors
 - <audio> element*, 164-166
 - <video> element*, 157-159

- <header> element, 202-203

- hearts, drawing, 25-26

- height attribute, 61, 149

- <hgroup> element, 204

- hidden controls, 65

history object

- API, 104-106
- back buttons, 107-110
- forward buttons, 110-112
- functions
 - back()*, 105
 - forward()*, 105
 - go()*, 105, 112-114
 - pushState()*, 106
 - replaceState()*, 106
- length, 105, 114-116
- onpopstate events, 106
- overview, 103-104
- pophistory.html
 - code*, 121-123
 - creating*, 106-107

popping data, 119-121

pushing data, 116-119

HTML5

new features

audio/video support, 9

browser history, 8

Canvas, 6

dragging and dropping, 7

elements, 10-11

form controls, 7

interdocument

messaging, 8

text editing, 8

web storage, 9

overview, 6

W3C specification, 6

HTML tables, creating, 66

I

iframe, creating, 98-100

images

controls, 65

drawing, 17-18

inline editing

attributes, 79

contenteditable, 79-80

designmode, 80

spellcheck, 80

editdiv.html example

code, 95-96

<div> element,

creating, 82

editiframe.html example, 96-98

buttons, adding, 98-100

code, 100-101

iframe, creating, 97-98

text

bolding, 83-85

display sources, 91-93

italicizing, 85-86

links, 88-90

spell checking, 91-94

underlining, 87-88

input events, 64

interdocument messaging.

See messaging

isPointInPath() function, 17

italicizing text, 85-86

J

JavaScript, drag and drop functions,
35-36

K

key() function, 170

<keygen> element, 204-205

L

length attribute, 105

browser history, 114-116

session storage, 170

line art

drawing, 22-23

filling, 24-25

lineCap attribute, 15

lineJoin attribute, 15

line styles, 15

lineTo() function, 17, 22

lineWidth attribute, 15

links, adding, 88-90

list attribute, 62-63

local datetime control, 72-73

local storage, 178-179

API, 180-181

browser support, 181

data

clearing, 186-187

retrieving, 184-185

storing, 182-183

localStorage.html example,
181-182, 188-189

W3C website, 180

loop attribute

<audio> element, 161

<video> element, 149

looping video, 156

M

<mark> element, 205-206

MathML, 192

max attribute, 62

maxlength attribute, 62

measureText() function, 17

messaging, 8

API, 127-129

data attribute, 128-129

onMessage() function, 128

origin attribute, 129

postMessage() function, 128

source attribute, 129

cross-domain, 126-127

domainchild.html, 140-141, 144-145

domainparent.html, 137-138, 143-144

receiving, 142-143

sending, 138-140

cross-window, 126

child.html, 132-133

parent.html, 129-130, 135-136

receiving, 134-135

sending, 130-132

introduction website, 127

<meter> element, 206-207

min attribute, 62

miterLimit attribute, 15

month controls, 65, 72-73

moveTo() function, 17, 22

multiple attribute, 62

N

<nav> element, 207-208

navigating browsers

Back buttons, 107-110

Forward buttons, 112-114

specific number of pages, 112-114

new elements

<article>194-195

<aside>195-196

<audio>. *See* <audio> element

<canvas>. *See* <canvas> element

<command>, 196-197

<datalist>, 198

<details>, 198-199

<embed>, 199-200

<figcaption>, 200-201

<figure>, 201-202

<footer>, 202

<header>, 202-203

<hgroup>, 204

<keygen>, 204-205

listing of, 10-11, 192-193

<mark>, 205-206

<meter>, 206-207

<nav>, 207-208

<output>, 208-209

progress>, 209-210

<rp>, 211

<rt>, 211-212

<ruby>, 212-213

<section>, 213

<source>, 214

<summary>, 215

<time>, 215-216

new features

audio/video support, 9

browser history, 8

<canvas> element, 6

controls, 7

- dragging and dropping, 7
- elements, 10-11
- interdocument messaging, 8
- text editing, 8
- web storage, 9

number controls

- creating, 70-71
- data types, 65

O

objects

- dataTransfer, 40
- history
 - API, 104-106
 - back buttons, 107-110
 - back() function, 105
 - data, pushing, 116-119
 - forward buttons, 110-112
 - forward() function, 105
 - go() function, 105, 112-114
 - length, 114-116
 - length attribute, 105
 - onpopstate events, 106
 - overview, 103-104
 - pophistory.html, 106-107, 121-123
 - popping data, 119-121
 - pushState() function, 106
 - replaceState() function, 106

OGG video conversions, 150-152

- ondragend attribute, 40
- ondragenter attribute, 38
- ondragover attribute, 39
- ondragstart attribute, 39
- ondrop attribute, 39
- ondrop events, 50
- onerror attribute

- <audio> element, 162
- <video> element, 150

onMessage() function, 128

- onpopstate events, 106, 119-121
- origin attribute, 129
- <output> element, 208-209
- over() function, 48-49

P

parent.html

- code, 135-136
- creating, 129-130
- sending message to child.html, 130-132

password controls, 65

pattern attribute, 62

placeholder attribute, 62

playing video, 156-157

pophistory.html

- code, 121-123
- creating, 106-107

popping data, 119-121

poster attribute, 150

postMessage() function, 128

preload attribute

- <audio> element, 161
- <video> element, 150

<progress> element, 209-210

pushing data, 116-119

pushState() function, 106, 116-118

Q

quadratic curves, 27-28

quadraticCurveTo() function, 17, 27

R

radio controls, 65

range controls

- creating, 70-71
- data types, 65

readonly attribute, 62

receiving messages

- cross-domain, 142-143
- cross-window, 134-135

recommendations (W3C), 6
 rectangles, drawing
 examples, 20-21
 functions, 16
 rect() function, 17
 removeItem() function, 171
 replaceState() function, 106
 required attribute, 62
 reset controls, 65
 rotate() function, 18
 <rp> element, 211
 <rt> element, 211-212
 <ruby> element, 212-213

S

Scalable Vector Graphics (SVG),
 191
 scale() function, 18
 search controls
 creating, 75-76
 data types, 65
 <section> element, 213
 selectedOption attribute, 63
 select() function, 63
 selectionEnd attribute, 63
 selectionStart attribute, 63
 send() function
 cross-domain messaging,
 138-140
 cross-window messaging, 130
 sending messages
 cross-domain, 138-140
 cross-window, 130-132
 session storage, 167-168.
 See also local storage
 API, 169-188
 browser support, 170
 data
 clearing, 175-177
 erasing, 167-168
 restoring, 168-169
 retrieving, 174-175
 storing, 172-173
 sessionstorage.html, 167, 171,
 177-178
 setData() function, 40
 setDragImage() function, 40
 setItem() function, 170
 setSelectionRange() function, 63
 shadowBlur attribute, 16
 shadowColor attribute, 16
 shadowOffsetX attribute, 16
 shadowOffsetY attribute, 16
 shadows, 16
 showSource() function, 91
 size attribute, 62
 <source> element, 214
 source attribute, 129
 specifications
 <canvas> element, 13
 drag-and-drop, 37
 form controls, 57
 HTML5, 6
 spellcheck attribute, 80
 spell checking
 enabling, 80
 text, 88-90
 src attribute, 62
 <audio> element, 162
 <video> element, 150
 start() function, 46
 starting drag operations, 46-47
 step attribute, 62
 stepDown() function, 63
 stepUp() function, 63
 stopping dropping operations, 51
 storage
 local, 178-179
 API, 180-181
 browser support, 181
 clearing data, 186-187

- data retrieval, 184-185*
- localStorage.html example, 181-182, 188-189*
- storing data, 182-183*
- W3C website, 180*
- session, 167-168
 - API, 169-188*
 - browser support, 170*
 - clearing data, 175-177*
 - data retrieval, 174-175*
 - erasing data, 167-168*
 - restoring data, 168-169*
 - storing data, 172-173*
- web, 9
- stroke() function, 17
- strokeRect() function, 16, 20
- strokeStyle attribute, 15, 22
- strokeText() function, 17, 30
- styling
 - <canvas> element, 15
 - draggable elements, 43-45
 - targets, 43-45
- submit controls, 65
- <summary> element, 215
- SVG (Scalable Vector Graphics), 191

T

- tables, creating, 66
- targets
 - creating, 41-42
 - dragging elements
 - creating, 42-43*
 - entrance, allowing, 47-48*
 - starting, 46-47*
 - dropping elements
 - allowing, 48-49*
 - ending, 51*
 - handling, 50*
 - styling, 43-45
- tel controls, 65

- text
 - annotations
 - <rp> element, 211
 - <rt> element, 211-212
 - <canvas> element, 17
 - controls
 - creating, 67-68*
 - data types, 65*
 - data
 - popping, 119-121*
 - pushing, 116-119*
 - display sources, 91-93
 - drawing, 30-31
 - editing, 8
 - formatting
 - bolding, 83-85*
 - italicizing, 85-86*
 - links, 88-90*
 - underlining, 87-88*
 - spell checking, 88-90
- textAlign attribute, 17
- textBaseline attribute, 17
- three triangles example, 22-23
- <time> element, 215-216
- time controls. *See* date and time controls
- transformations, 18
- translate() function, 18

U

- underlining text, 87-88
- updateBar() function, 210
- URL controls
 - creating, 68-69
 - data types, 65

V

- valueAsDate attribute, 63
- valueAsNumber attribute, 63
- value attribute, 63

- <video> element
 - API, 148-150
 - browser support, 149
 - controls, adding, 155
 - error handling, 157-159
 - looping, 156
 - OGG conversions, 150-152
 - playing automatically, 156-157
 - support, 9
 - video.html example
 - body*, 153
 - <video> element, *adding*, 154-155
- W3C website, 149

W-Z

- W3C (World Wide Web Consortium), 5
 - Amaya test browser, 192
 - <audio> element, 161
 - new elements, 193
 - recommendations, 6
 - specifications
 - drag-and-drop*, 37
 - HTML5*, 6
 - local storage*, 180
 - web form controls*, 57
 - <video> element, 149
 - website, 5
- web form controls. *See* form controls
- webforms.html example
 - code, 76-78
 - HTML table, creating, 66
- webforms.php example, 78
- websites
 - Amaya test browser, 192
 - <canvas> element API, 14-15
 - file converters, 151
 - messaging introduction, 127

- W3C (World Wide Web Consortium), 5
 - Amaya test browser*, 192
 - drag-and-drop specification*, 37
 - form control specification*, 57
 - HTML5 specification*, 6
 - local storage*, 180
 - new elements*, 193
 - <video> element, 149
 - <audio> element, 161
- web storage, 9
 - local, 178-179
 - API, 180-181
 - browser support, 181
 - clearing data, 186-187
 - localStorage.html example*, 181-182, 188-189
 - retrieving data, 184-185
 - storing data, 182-183
 - W3C website, 180
- session, 167-168
 - API, 169-188
 - browser support, 170
 - clearing data, 175-177
 - erasing data, 167-168
 - restoring data, 168-169
 - retrieving data, 174-175
 - storing data, 172-173
 - sessionstorage.html*, 167, 171, 177-178

- week control, 72-73
- week data types, 65
- width attribute, 62, 150
- World Wide Web Consortium (W3C). *See* W3C