Tutorial 10
Programming with JavaScript

HTML, CSS, and Dynamic HTML

5TH EDITION

Carey, P. (2013). New perspectives on HTML, CSS, and Dynamic HTML: Comprehensive (5th ed.). Australia ; United Kingdom: Course Technology Cengage Learning.

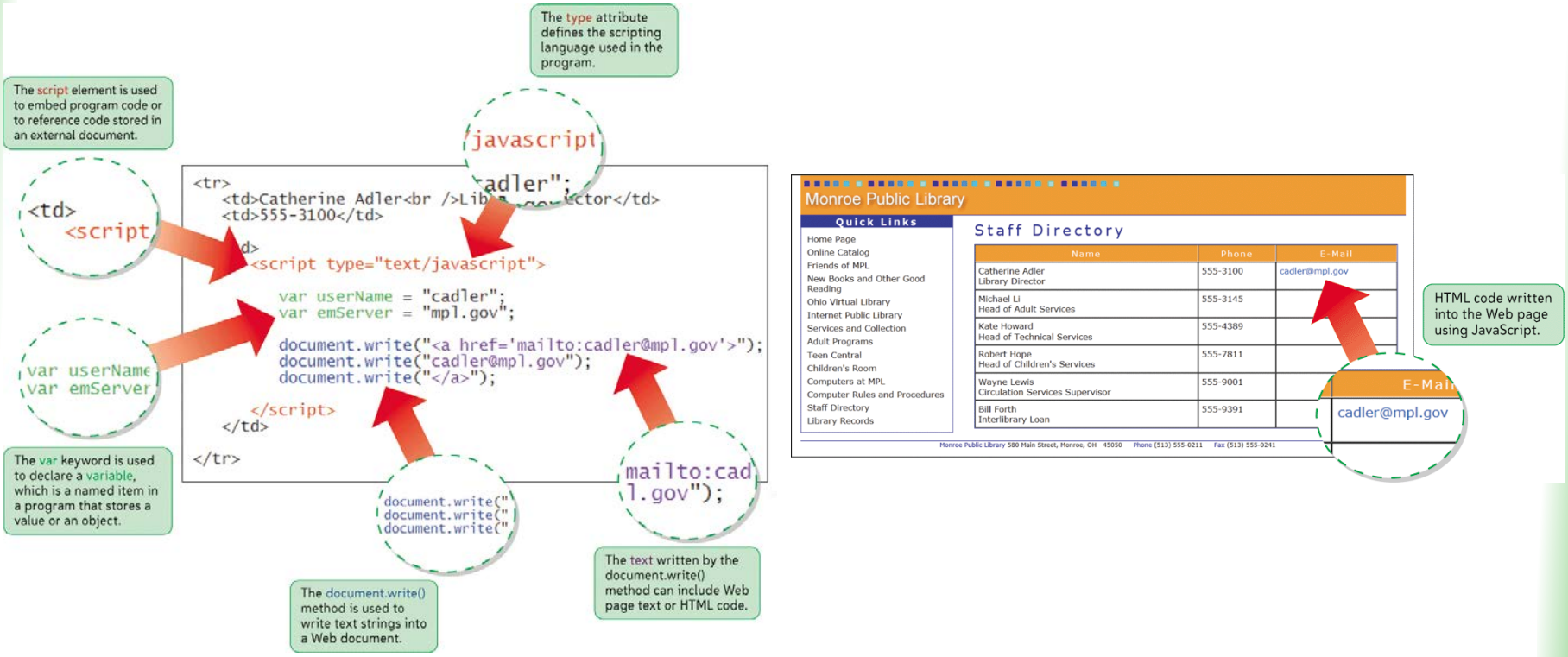
Objectives

- Learn the history of JavaScript
- Create a script element
- Write text to a Web page with JavaScript
- Understand basic JavaScript syntax
- Declare and work with variables
- Learn about JavaScript data types

Objectives

- Create and call a JavaScript function
- Access an external JavaScript file
- Add comments to JavaScript code
- Learn about basic debugging techniques and tools

Using JavaScript Variables



Introducing JavaScript

- **Spam** is essentially junk e-mail—messages that advertise products and services not requested by the recipient
  - A **spammer** is a person who sends these unsolicited e-mails, sometimes in bulk
- An **e-mail harvester** is a program that scans documents, usually Web pages, looking for e-mail addresses

Introducing JavaScript

Figure 10-2 Viewing e-mail addresses in the HTML file

e-mail addresses in the staff directory

```
<tr>
   <td>Catherine Adler<br />Library Director</td>
   <td>555-3100</td>
   <td><a href="mailto:cadler@mpl.gov">cadler@mpl.gov</a>
   </td>
</tr>
<tr>
   <td>Michael Li<br />Head of Adult Services</td>
   <td>555-3145</td>
   <td><a href="mailto:mikeli@mpl.gov">mikeli@mpl.gov</a>
   </td>
</tr>
<tr>
   <td>Kate Howard<br />Head of Technical Services</td>
   <td>555-4389</td>
   <td><a href="mailto:khoward@mpl.gov">khoward@mpl.gov</a>
   </td>
</tr>
```

Figure 10-3 Scrambling e-mail addresses

```
<script type="text/javascript">
   showEM("reldac","vog.lpm");
</script>
```

e-mail address scrambled with JavaScript, keeping it from appearing in the page code

the browser runs a JavaScript program to unscramble the e-mail address …

… that the end user can view

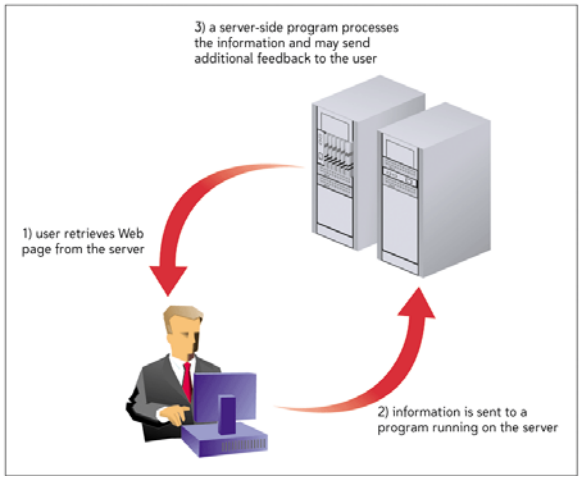
Introducing JavaScript

- Server-side programs are placed on the server that hosts a Web site
  - Can be problematic
- Client-side programming runs programs on each user's computer

Introducing JavaScript

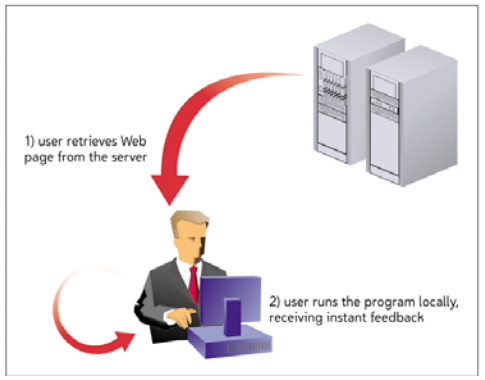
Server-Side Programming

Figure 10-4 Server-side programming



Client-Side Programming

Figure 10-5 Client-side programming



The Development of JavaScript

- JavaScript is a subset of Java
- Differences between Java and JavaScript:
  - Java is a compiled language
  - JavaScript is an interpreted language

Comparing Java and JavaScript

Figure 10-7 Comparing Java and JavaScript

Java	JavaScript
A compiled language	An interpreted language
Requires the JDK (Java Development Kit) to create an applet	Requires a text editor
Requires a Java virtual machine or interpreter to run an applet	Requires a browser that can interpret JavaScript code
Applet files are distinct from HTML files	Programs can be embedded within HTML files
Source code is hidden from users	Source code is accessible to users
Powerful, requiring programming knowledge and experience	Simpler, requiring less programming knowledge and experience
Secure; programs cannot write content to a hard disk	Secure; programs cannot write content to a hard disk; however, there are more security holes than in Java
Compiled code runs on the client side computer within an applet window	Code run on the client side computer directly within the Web browser

Working with the `script` Element

- A JavaScript program can be placed directly in an HTML file or it can be saved in an external text file
- Insert a client-side script in a Web page when using the script element

```
<script type="mime-type">
   script commands
</script>
```

Embedding a Script

- To place a `script` element in a Web page, insert the two-sided tag

```
<script type="mime-type">
   script commands
</script>
```

- where *`mime-type`* defines the language in which the script is written and *`script commands`* represents commands written in the scripting language
- For JavaScript programs, set `mime-type` to `text/javascript`.

Working with the `script` Element

- Every JavaScript program consists of a series of Statements
- Each **statement**—also known as a **command**—is a single line that indicates an action for the browser to take

Writing Output to a Web Document

- An object is any item—from the browser window itself to a document displayed in the browser to an element displayed within the document
- A method is a process by which JavaScript manipulates or acts upon the properties of an object

Writing Output to the Web Page

- To write text to a Web page with JavaScript, use the method

  `document.write("text");`

  where *text* is the HTML code to be written to the Web page

Understanding JavaScript Syntax

- JavaScript is case sensitive
- Ignores most occurrences of extra white space
- Do not break a statement into several lines
- The + symbol used in this command combines several text strings into a single text string

Working with Variables

- A variable is a named item in a program that stores a data value
- You introduce variables in your code by declaring them
  - **Declaring** a variable tells the JavaScript interpreter to reserve memory space for the variable

Declaring a JavaScript Variable

- To declare a JavaScript variable, use the statement

```
var variable;
```

where *variable* is the name assigned to the variable.

- To declare a JavaScript variable and set its initial value, use

```
var variable = value;
```

where *value* is the initial value of the variable.

Working with Data Types

- JavaScript data types:
  - Numeric values
  - Text strings
  - Boolean values
  - Null values
- You must **declare** a variable before using it

Working with Data Types

- Numeric value is any number, such as 13, 22.5, -3.14159 etc.
  - Can also be expressed in scientific notation
- Text string is any group of text characters, such as “Hello” or “Happy Holidays!”
  - Must be enclosed within either double or single quotations (but not both)
- Boolean values accept only true and false values
- Null value has no value at all

Working with Data Types

- JavaScript is a weakly typed language
- The + symbol can be used with either numeric values or text strings

```
var total = 5 + 4;

var emLink = "cadler" + "@" +
"mpl.gov";
```

Writing JavaScript Functions

A function is a collection of commands that performs an action or returns a value.

A function can contain parameters, which are variables associated with the function.

Single-line comments are prefaced by the // characters and are used to add comments in-line with JavaScript code.

Multi-line comments are enclosed within the /* and */ symbols, and are used to add comments as blocks of text.

A function can be called using the statement function(*values*), where function is the function name and *values* represents one or more values passed to the function parameters.

```
function showEM(userName,emServer) {
   /*
      The showEM() function displays a link to the user's
      e-mail address.
      The text of the user and e-mail server names are entered in
      reverse order to thwart e-mail harvesters.
   */

   userName = stringReverse(userName); // reverse the text of the userName parameter
   emServer = stringReverse(emServer); // reverse the text of the emServer parameter

   var emLink = userName + "@" + emServer; // combine the text of userName and emServer

   document.write("<a href='mailto:" + emLink + "'>");
   document.write(emLink);
   document.write("</a>");
}
```

HTML code written by calling the showEM() function.

```
<script type="text/javascript">
   showEM("reldac","vog.lpm");
</script>
```

Staff Directory

Name	Phone	E-Mail
Catherine Adler Library Director	555-3100	cadler@mpl.gov
Michael Li Head of Adult Services	555-3145	mikeli@mpl.gov
Kate Howard Head of Technical Services	555-4389	khoward@mpl.gov
Robert Hope Head of Children's Services	555-7811	rhope@mpl.gov
Wayne Lewis Circulation Services Supervisor	555-9001	wlewis@mpl.gov
Bill Forth Interlibrary Loan	555-9391	bforth@mpl.gov

Creating a JavaScript Function

- A function is a collection of commands that performs an action or returns a value
- A function name identifies a function
- Parameters are values used by the function
- The function is executed only when called by another JavaScript command

```
function_name(parameter values)
```

Creating a JavaScript Function

Figure 10-16 Inserting the showEM() function

```
<link href="mplstyles.css" rel="stylesheet" />

<script type="text/javascript">
   function showEM(userName, emServer) {
      var emLink = userName + "@" + emServer;
      document.write("<a href='mailto:" + emLink + "'>");
      document.write(emLink);
      document.write("</a>");
   }
</script>
```

Creating and Calling a JavaScript Function

- For a function to return a value, it must include a return statement

```
   function function_name(parameters){
         JavaScript commands
         return value;
   }
```

Accessing an External JavaScript File

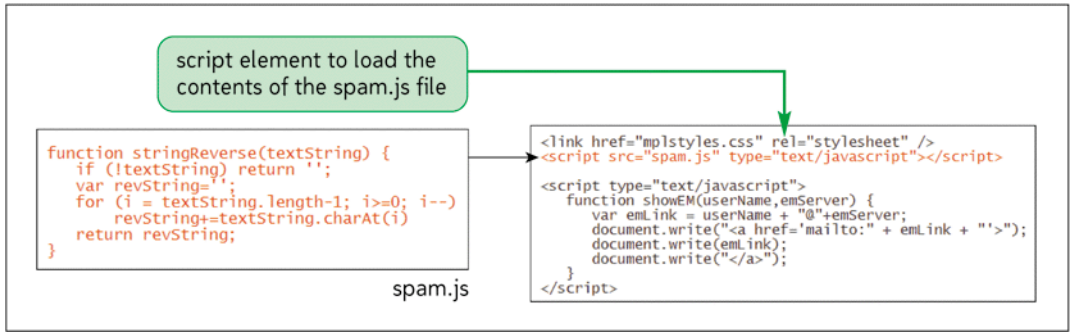
- The code to access an external script file is:

  ```
  <script src="url" type="mime-type"></script>
  ```

  to the Web page, where *url* is the URL of the external document and *mime-type* is the language of the code in the external script file.
- For JavaScript files, set the *mime-type* to *text/javascript*.

Accessing an External JavaScript File

Figure 10-21 Using an external script file



Commenting JavaScript Code

- Commenting your code is an important programming practice

  // comment text

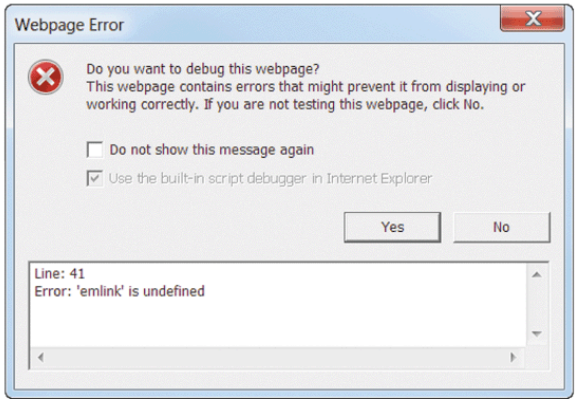
Figure 10-27 Adding comments to the showEM() function

```
<script type="text/javascript">
   function showEM(userName,emServer) {
      /*
         The showEM() function displays a link to the user's
         e-mail address.
         The text of the user and e-mail server names are entered in
         reverse order to thwart e-mail harvesters.
      */

      userName = stringReverse(userName); // reverse the text of the userName parameter
      emServer = stringReverse(emServer); // reverse the text of the emServer parameter

      var emLink = userName + "@"+emServer; // combine the text of userName and emServer
      document.write("<a href='mailto:" + emLink + "'>");
      document.write(emLink);
      document.write("</a>");
   }
</script>
```

Debugging Your JavaScript Programs

- **Debugging** is the process of searching code to locate a source of trouble
- There are three types of errors:
  - Load-time errors
  - Run-time errors
  - Logical errors

Figure 10-29 Reporting a run-time error in Internet Explorer

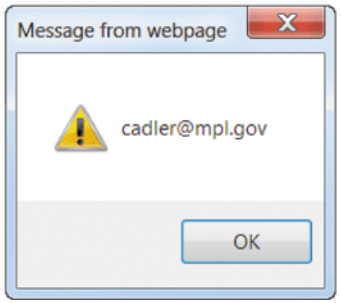


Debugging Tools and Techniques

- Modular code entails breaking up a program's different tasks into smaller, more manageable chunks
- An alert dialog box is a dialog box generated by JavaScript that displays a text message with an OK button

```
alert(text);
```

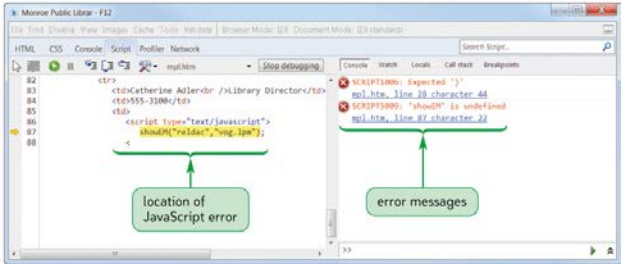
Figure 10-31 Alert dialog box



Debugging Tools and Techniques

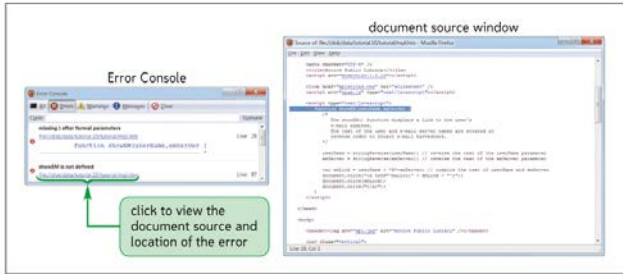
Internet Explorer Developer Tools

Figure 10-32 Internet Explorer Developer Tools window



Firefox Error Console and Document Source Window

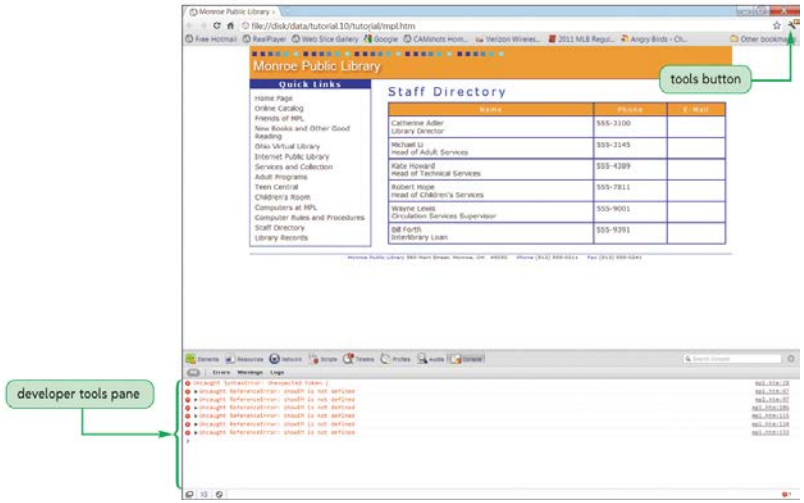
Figure 10-33 Firefox Error Console and document source window



Debugging Tools and Techniques

Google Chrome Developer Tools Pane

Figure 10-34 Google Chrome developer tools pane



Safari Web Inspector Window

Figure 10-35 Safari Web Inspector window