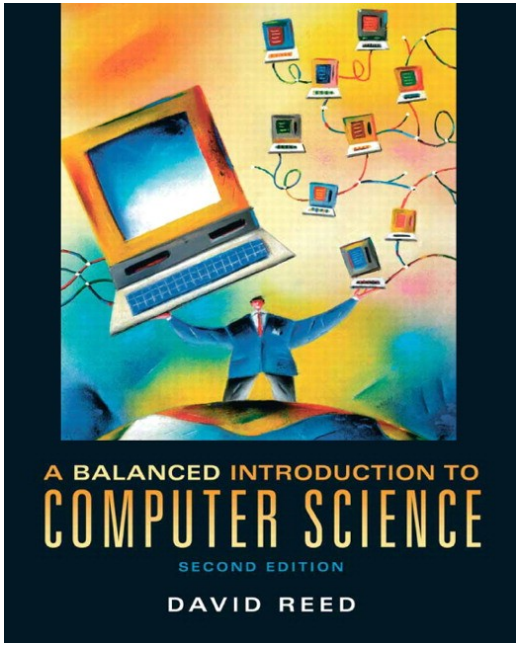
A Balanced Introduction to Computer Science, 2/E

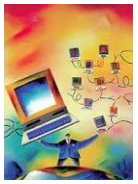
David Reed, Creighton University

ISBN 978-0-13-601722-6



Chapter 4
JavaScript and Dynamic Web Pages

Static vs. Dynamic Pages



recall: a Web page uses HTML tags to identify page content and formatting information

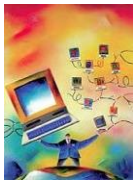
HTML can produce only *static pages*

- static pages look the same and behave in the same manner each time they are loaded into a browser

in 1995, researchers at Netscape developed JavaScript, a language for creating *dynamic pages*

- Web pages with JavaScript can change their appearance:
  - over time (e.g., a different image each time that a page is loaded), or
  - in response to a user's actions (e.g., typing, mouse clicks, and other input methods)

Programming Languages



JavaScript is a *programming language*

- a *programming language* is a language for specifying instructions that a computer can execute
- each *statement* in a programming language specifies a particular action that the computer is to carry out

  (e.g., changing an image or opening a window when a button is clicked)

some programming languages are general-purpose

- popular languages include C++, Java, J#

JavaScript was defined for a specific purpose: *adding dynamic content to Web pages*

- can add JavaScript statements to a Web page using the HTML tags

```
<script type="text/javascript"> . . . </script>
```

- when the browser displays the page, any statements inside the SCRIPT tags are executed and the result is displayed

Simple Dynamic Page

below is a simple Web page with dynamic content

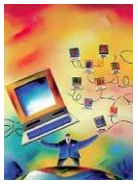
- *note:* dynamic content can be mixed with static HTML content

it demonstrates two types of JavaScript statements

- an `assignment` statement that asks the user for input and stores that input
- a `write` statement that writes text into the HTML page

```
 1. <html>
 2.   <!-- greet.html                                  Dave Reed -->
 3.   <!-- Web page that displays a personalized greeting. -->
 4.   <!-- ================================================== -->
 5.
 6.   <head>
 7.     <title> Greetings </title>
 8.   </head>
 9.
10.   <body>
11.     <script type="text/javascript">
12.       firstName = prompt("Please enter your name", "");
13.
14.       document.write("<p>Hello " + firstName + ", welcome to my Web page.</p>");
15.     </script>
16.
17.     <p>
18.     Whatever else you want to appear in your Web page...
19.     </p>
20.   </body>
21. </html>
```

Assignment Statement

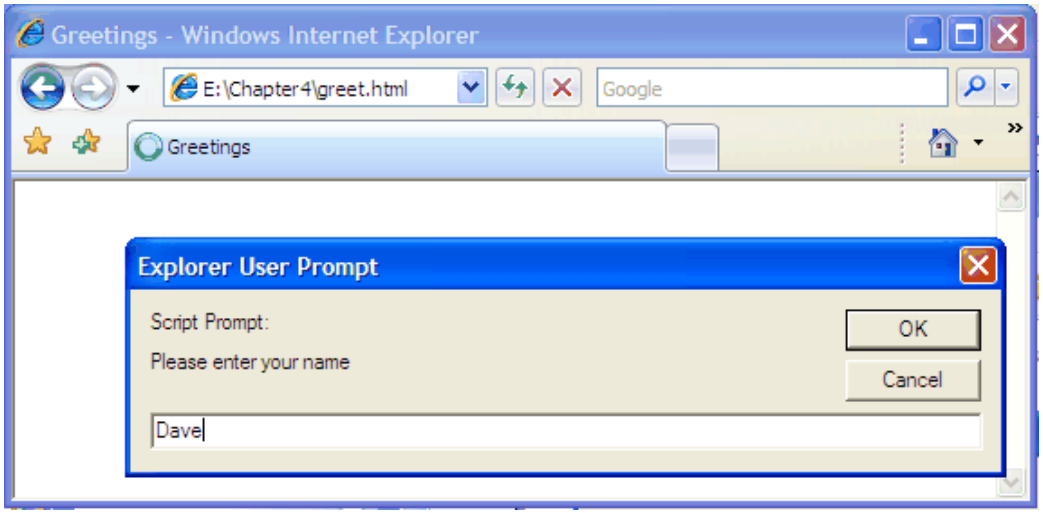


when an assignment statement involving `prompt` is executed by the browser

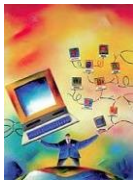
- a separate window is opened with a text box for the user to enter text
- when the user is done typing, he/she can click on the OK button

```
firstName = prompt("Please enter your name", "");
```

- when OK is clicked, the text entered is assigned to a variable
  - a *variable* is a name used to symbolize a dynamic value
  - here, the variable `firstname` is used to store the text entered by the user



Assignment Statement (cont.)

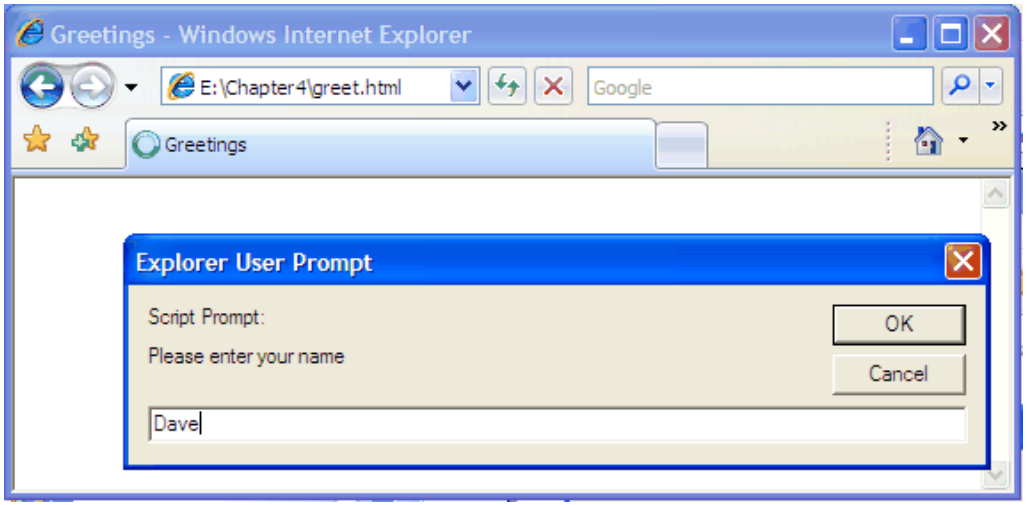


the general form of an assignment statement using a prompt is

```
VARIABLE = prompt("PROMPT MESSAGE", "");
```

- the variable name can vary depending on the task at hand
  - here, the variable is used to store the user's first name, so `firstName` is a meaningful name

- the prompt message that appears in the window can likewise change
  - here, the message "Please enter your name" tells the user what is expected



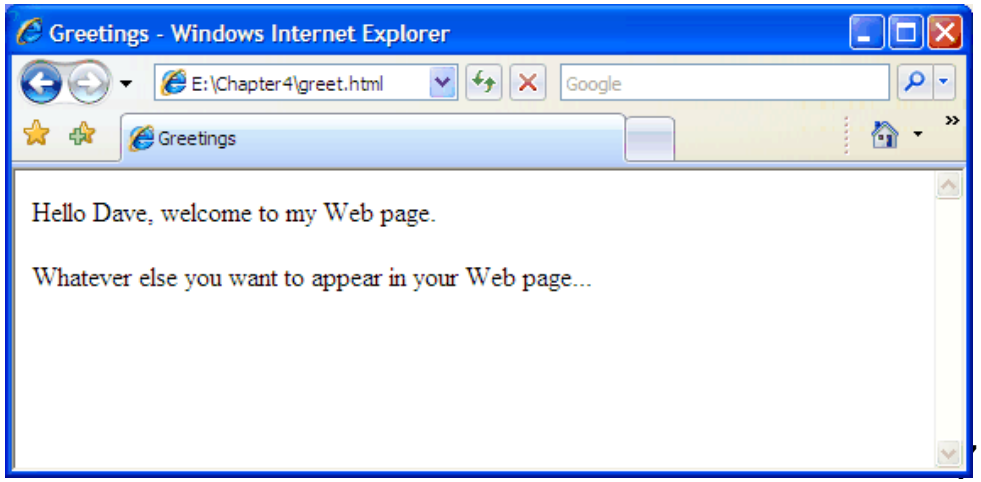
Write Statement

when a write statement is executed by the browser

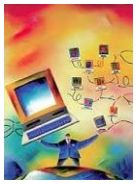
- the message specified in the statement is written into the HTML page
- a message can include
  - a string literal – text enclosed in quotes
  - a variable
  - a combination of strings and variables, connected via '+'

```
document.write("<p>Hello " + firstName +
               ", welcome to my Web page.</p>");
```

- when a variable is encountered, the browser substitutes the value currently assigned to that variable



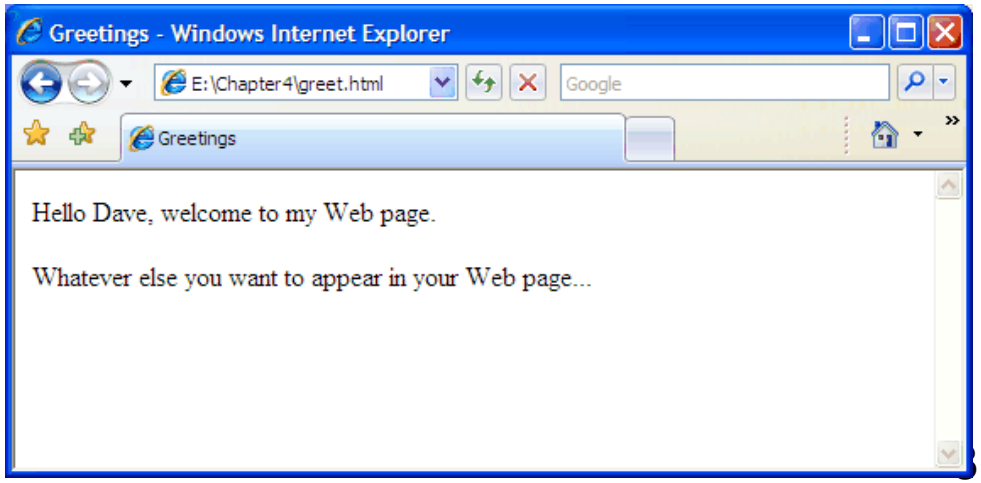
Write Statement (cont.)



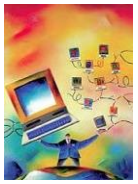
the general form of a write statement is

```
document.write("MESSAGE TO BE DISPLAYED " + VARIABLE +
               " MORE MESSAGE" + ...);
```

- note that the statement can be broken across lines, as long as no string literal is split (i.e., the beginning and ending quotes of a string must be on same line)
- the pieces of the message are displayed in sequence, with no spaces in between
  - if you want spaces, you have to enter them in the text



Formatted Output



the output produced by a write statement is embedded in the page

- the browser displays this output just as it does any other text
- if the text contains HTML tags, the browser will interpret the tags and format the text accordingly

```
document.write("<p>Hello <i>" + firstName +
               "</i>, welcome to my Web page.</p>");
```

assuming the variable `firstName` has been assigned `"Dave"`, the browser would execute the statement to produce

```
<p>Hello <i>Dave</i>, welcome to my Web page.</p>
```

which would be displayed by the browser as

Hello *Dave*, welcome to my Web page.

Syntax Errors

an error in the format of an HTML or JavaScript statements is known as a *syntax error*

- some syntax errors are ignored by the browser
  - e.g., misspelling an HTML tag name
- most JavaScript syntax errors will generate an error message

```
document.write("This example is illegal since the
                string is broken across lines");
```

yields: **Error: unterminated string literal**

```
document.write("The value of x is " x);
```

yields: **Error: missing) after argument list**

JavaScript Variables

a variable name can be any sequence of letters, digits, and underscores (but must start with a letter)

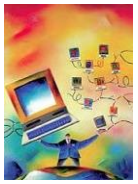
- valid: `tempInFahr` `SUM` `current_age` `Sum2Date` `x`
- invalid: `2hotforU` `salary$` `two words` `"sum_to_date"`

variable names are case sensitive, so `Sum` and `SUM` are treated as different variables

Reserved Words That Shouldn't Be Used as Variable Names

abstract	document	if	package	throw
boolean	double	implements	parent	throws
break	else	import	private	top
byte	enum	in	protected	transient
case	export	instanceof	public	true
catch	extends	int	return	try
char	false	interface	screen	typeof
class	final	length	self	var
const	finally	location	short	void
continue	float	long	static	volatile
debugger	for	name	super	while
default	function	native	switch	window
delete	goto	new	synchronized	with
do	history	null	this	

Variables & Memory Cells



computers keep track of the values that variables represent by associating each variable with a specific piece of memory, known as a *memory cell*

- when a JavaScript assignment is executed,

```
firstName = prompt("Please enter your name", "");
```

- the value entered by the user (e.g., `"Dave"`) is stored in a memory cell associated with the variable `firstName`

- any future reference to the variable name evaluates to the value stored in its associated memory cell

Another Example

once you create a variable, you can repeatedly assign values to it

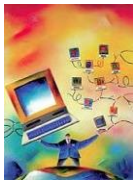
- only the most recent value is retained in memory

EXAMPLE: suppose we want to prompt the user for two different foods

- if only one food is needed at a time, we can reuse the same variable

```
1.  <html>
2.    <!-- food.html                                          Dave Reed -->
3.    <!-- Web page that prompts for and displays food preferences. -->
4.    <!-- ================================================== -->
5.
6.    <head>
7.      <title> Who's Hungry? </title>
8.    </head>
9.
10.   <body>
11.     <script type="text/javascript">
12.       food = prompt("What is your favorite food?", "");
13.       document.write("<p>Your favorite food is " + food + ".</p>");
14.
15.       food = prompt("What is your least favorite food?", "");
16.       document.write("<p>Your least favorite food is " + food + ".</p>");
17.     </script>
18.   </body>
19. </html>
```

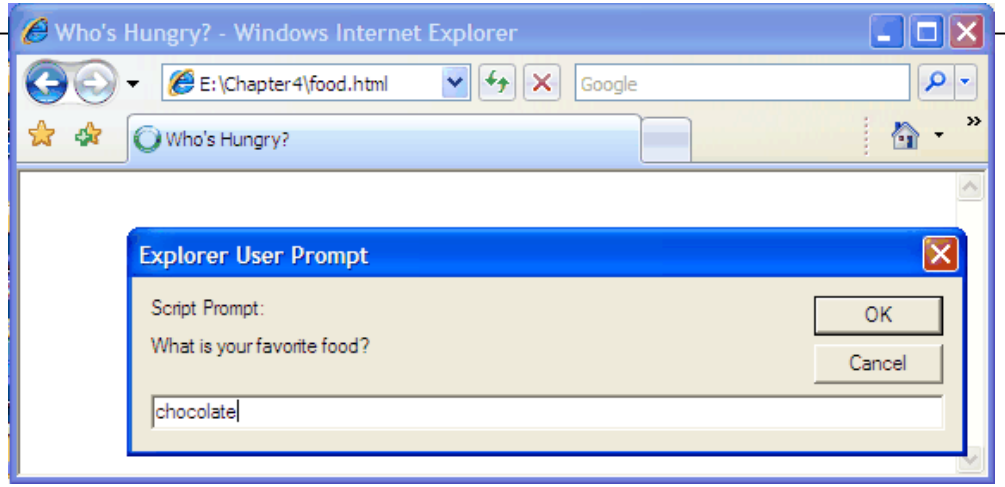
Reusing Variables

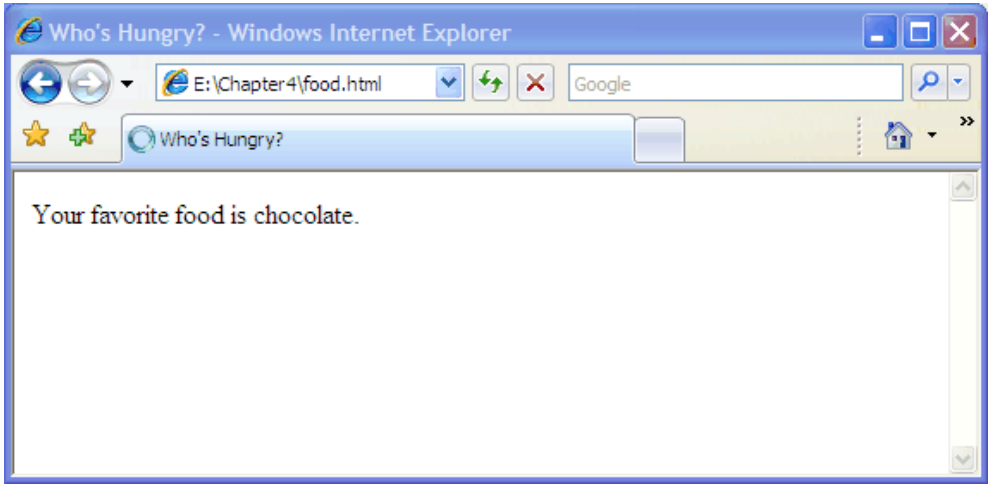


```
food = prompt("What is your favorite food?", "");
document.write("<p>Your favorite food is " + food + "</p>");
```

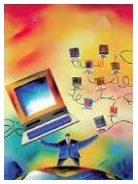
the first pair of statements

- stores the user's favorite food
- displays that food in the page





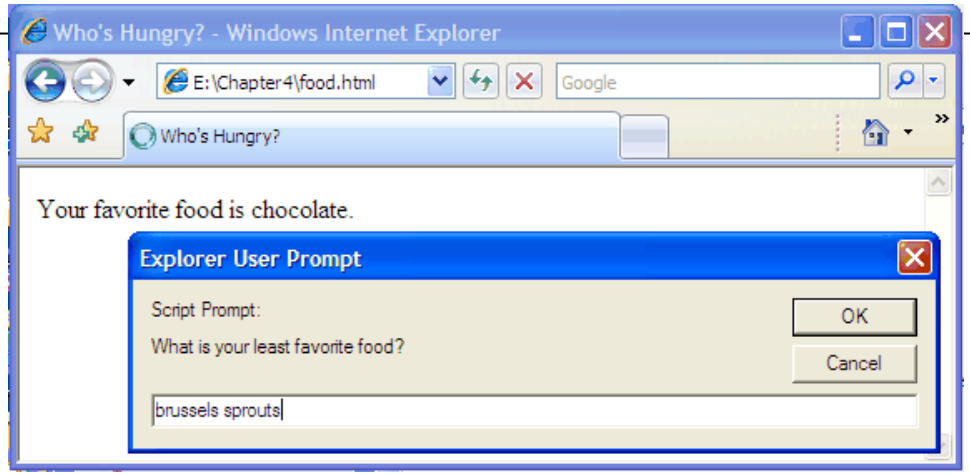
Reusing Variables (cont.)

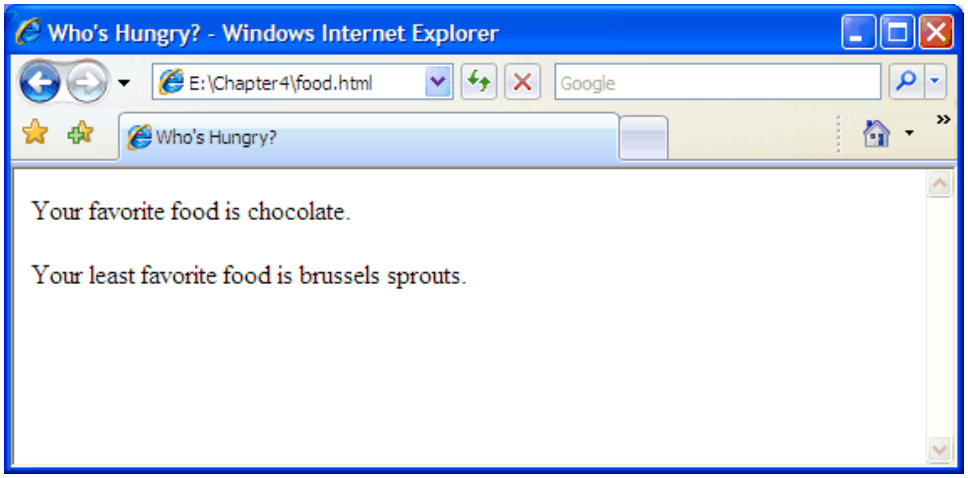


```
food = prompt("What is your least favorite food?", "");
document.write("<p>Your least favorite food is " + food + "</p>");
```

the second pair of statements

- stores the user's least favorite food (overwriting the old value)
- displays that food in the page





Prompts with Defaults

so far, all prompts have been of the form

```
VARIABLE = prompt("PROMPT MESSAGE", "");
```

sometimes it makes sense to provide default values for prompts

- can specify a string literal instead of ""
- this string will appear in the prompt box when it appears
  - if the user wants to accept the default value, can just click OK

EXAMPLE: suppose we wanted to create a page that displays a verse of the children's song, *Old MacDonald had a Farm*

- the page should be able to display any verse
- can accomplish this by prompting the user for the animal and sound
- can specify default values so that it is easy to display a common verse

```
animal = prompt("Enter a kind of animal:", "cow");
sound = prompt("What kind of sound does it make?", "moo");
```

Old MacDonald

this page prompts the user for the animal and sound ("cow" and "moo", by default), then displays a verse using those values

- `<br />` tags are embedded to break the output onto separate lines

```
 1. <html>
 2.   <!-- oldmac.html                                        Dave Reed -->
 3.   <!-- Web page that displays a verse of Old MacDonald.           -->
 4.   <!-- ========================================================== -->
 5.
 6.   <head>
 7.     <title> Old MacDonald </title>
 8.   </head>
 9.
10.   <body>
11.     <h3 style="text-align:center">Old MacDonald Had a Farm</h3>
12.
13.     <script type="text/javascript">
14.       animal = prompt("Enter a kind of animal:", "cow");
15.       sound = prompt("What kind of sound does it make?", "moo");
16.
17.       document.write("<p>Old MacDonald had a farm, E-I-E-I-0.<br />");
18.       document.write("And on that farm he had a " + animal + ", E-I-E-I-0.<br />");
19.       document.write("With a " + sound + "-" + sound + " here, and a " +
20.                      sound + "-" + sound + " there,<br />");
21.       document.write(" here a " + sound + ", there a " + sound +
22.                      ", everywhere a " + sound + "-" + sound + ".<br />");
23.       document.write("Old MacDonald had a farm, E-I-E-I-0.</p>");
24.     </script>
25.   </body>
26. </html>
```

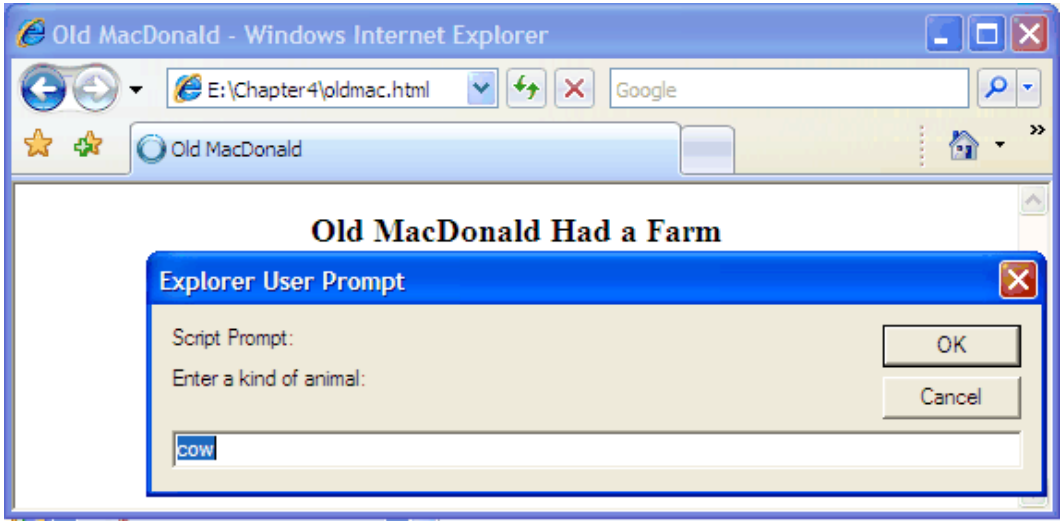
Old MacDonald (cont.)

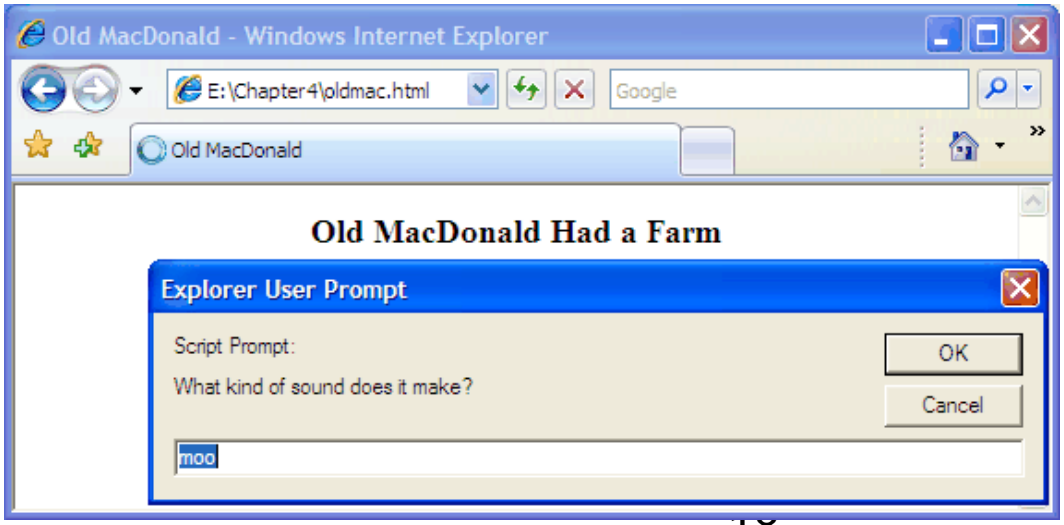
the default values automatically appear in the prompt boxes

- the user can click OK to accept the defaults

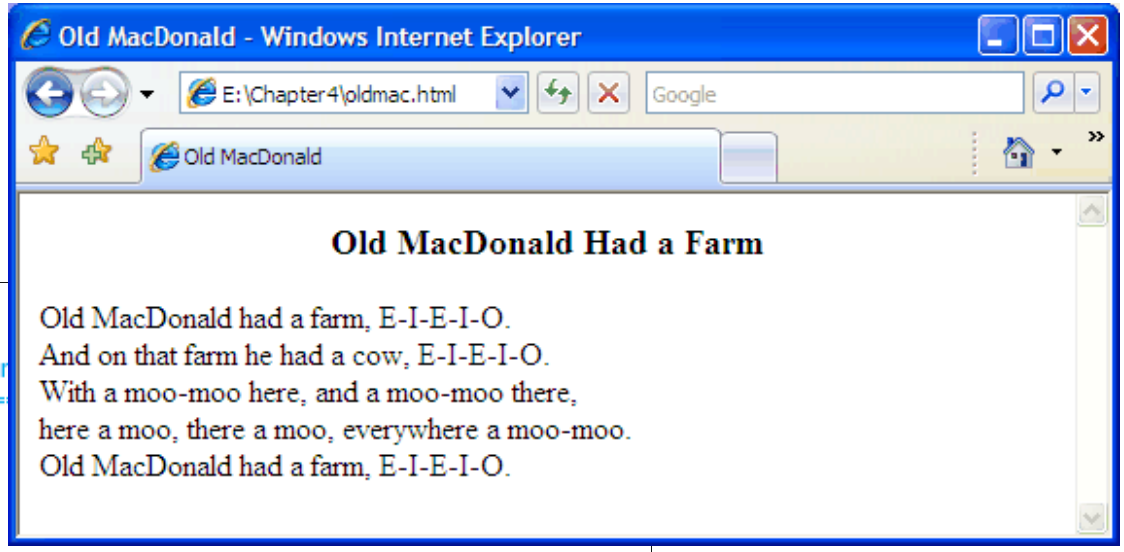
  OR

- type new values into the prompt box





Old MacDonald (cont.)




```
1.  <html>
2.    <!-- oldmac.html
3.    <!-- Web page that displays a ver
4.    <!-- ============================
5.
6.    <head>
7.      <title> Old MacDonald </title>
8.    </head>
9.
10.   <body>
11.     <h3 style="text-align:center">Old MacDonald Had a Farm</h3>
12.
13.     <script type="text/javascript">
14.       animal = prompt("Enter a kind of animal:", "cow");
15.       sound = prompt("What kind of sound does it make?", "moo");
16.
17.       document.write("<p>Old MacDonald had a farm, E-I-E-I-0.<br />");
18.       document.write("And on that farm he had a " + animal + ", E-I-E-I-0.<br />");
19.       document.write("With a " + sound + "-" + sound + " here, and a " +
20.                      sound + "-" + sound + " there,<br />");
21.       document.write(" here a " + sound + ", there a " + sound +
22.                      ", everywhere a " + sound + "-" + sound + ".<br />");
23.       document.write("Old MacDonald had a farm, E-I-E-I-0.</p>");
24.     </script>
25.   </body>
26. </html>
```

Localizing Changes

so far, we have used variables to store values read in via prompts

another common use is to store values used repeatedly in a page

- suppose we wanted to change the spelling of the refrain in Old MacDonald ("E-I-E-I-O" → "Eeyigh-Eeyigh-Oh")
- as is, would need to find and update all occurrences in the verse

- instead, could use a variable to store the refrain

```
refrain = "E-I-E-I-O";
document.write("<p>Old MacDonald had a farm, " + refrain + ".<br />");
document.write("And on that farm he had a " + animal + ", " +
               refrain + ".<br />");
. . .
```

- now, to update the value in the entire verse, simply must change the assignment

```
refrain = "Eeyigh-Eeyigh-Oh";
document.write("<p>Old MacDonald had a farm, " + refrain + ".<br />");
document.write("And on that farm he had a " + animal + ", " +
               refrain + ".<br />");
. . .
```