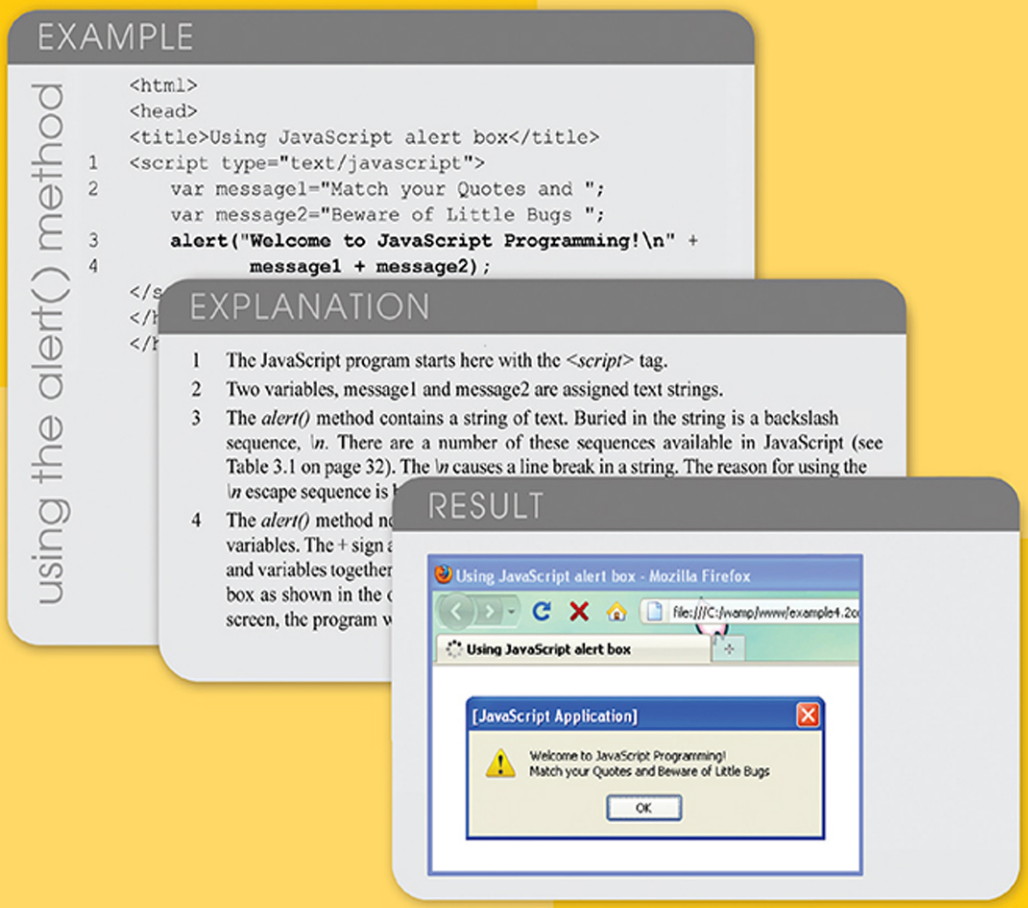
JavaScript *by* Example

Second Edition



Ellie Quigley

JavaScript by Example

Second Edition

Table 9.9 *String* Object (HTML) Methods (continued)

Method	***Formats as HTML***
String.bold()	*String*
String.fixed()	*<tt>String</tt>*
String.fontcolor(color)	*String* e.g., *String*
String.fontsize(size)	*String* e.g., *String*
String.italics()	*String*
String.link(URL)	*String* e.g., *String*
String.small()	*String*
String.strike()	*<strike>String</strike>* (puts a line through the text)
String.sub()	*_{String}* (creates a subscript)
String.sup()	*^{String}* (creates a superscript)

EXAMPLE 9.26

```
  <html>
    <head><title>String object</title></head>
    <body bgcolor="yellow">
      <big>
      <font face="arial">
      <h2>Working with String Objects:</h2>
      <script type="text/javascript">
1       var str1 = new String("Hello world!");
        // Use a String constructor
2       var str2 = "It's a beautiful day today.";
        document.write(str1) + "<br />";
3       document.write(str1.fontcolor("blue")+"<br />");
4       document.write(str1.fontsize(8).fontcolor("red").bold()+
                       "<br />");
5       document.write(str1.big()+ "<br />");
6       document.write("Good-bye, ".italics().bold().big() +
                        str2 + "<br />");
      </script>
      </font>
      </big>
    </body>
  </html>
```

EXPLANATION

1 A *String* object is created with the *String()* constructor.
2 A string primitive is created the literal way.
3 The *fontcolor()* method is used to change the color of the string to blue. This method emulates the HTML tag, <*font color="blue"*>.
4 The *fontsize()*, *fontcolor()*, and *bold()* methods are used as properties of the string.
5, 6 The HTML method is concatenated to the string *"Good-bye, "* causing it to be displayed in italic, bold, large text (see Figure 9.30).

Figure 9.30 Properties of the *String* object are used to change its appearance and determine its size.

There are a number of methods (see Table 9.10) provided to manipulate a string.

Table 9.10 Methods for String Manipulation

Method	***What It Does***
charAt(index)	Returns the character at a specified index position.
charCodeAt(index)	Returns the Unicode encoding of the character at a specified index position.
concat(string1, ..., stringn)	Concatenates string arguments to the string on which the method was invoked.
fromCharCode(codes)	Creates a string from a comma-separated sequence of character codes.
indexOf(substr, startpos)	Searches for the first occurrence of *substr* starting at *startpos* and returns the *startpos (index value)* of *substr*.
lastIndexOf(substr, startpos)	Searches for the last occurrence of *substr* starting at *startpos* and returns the *startpos (index value)* of *substr*.
replace(searchValue, replaceValue)	Replaces *searchValue* with *replaceValue*.

Continues

Table 9.10 Methods for String Manipulation (continued)

Method	*What It Does*
search(regexp)	Searches for the regular expression and returns the index of where it was found.
slice(startpos, endpos)	Returns string containing the part of the string from *startpos* to *endpos*.
split(delimiter)	Splits a string into an array of words based on *delimiter*.
substr(startpos, endpos)	Returns a subset of string starting at *startpos* up to, but not including, *endpos*.
toLocaleLowerCase()	Returns a copy of the string converted to lowercase.
toLocaleUpperCase()	Returns a copy of the string converted to uppercase.
toLowerCase()	Converts all characters in a string to lowercase letters.
toString()	Returns the same string as the source string.
toUpperCase()	Converts all characters in a string to uppercase letters.
valueOf	Returns the string value of the object.

Methods That Find a Position in a String. A substring is a piece of an already existing string; thus *eat* is a substring of both *create* and *upbeat*, and *Java* is a substring of *JavaScript*. When a user enters information, you want to see if a certain pattern of characters exist, such as the @ in an e-mail address or a zip code in an address. JavaScript provides a number of methods to assist you in finding substrings.

The *indexOf()* and the *lastIndexOf()* methods are used to find the first instance or the last instance of a substring within a larger string. They are both case sensitive. The first character in a string is at index value 0, just like array indexes. If either of the methods finds the substring, it returns the position of the first letter in the substring. If either method can't find the pattern in the string, then a –1 is returned.

EXAMPLE 9.27

```
  <html>
    <head><title>Substrings</title></head>
    <body bgcolor="lightgreen">
      <font face="arial"
      <big>
      Searching for an @ sign
      <script type="text/javascript">
1       var email_addr=prompt("What is your email address? ","");
```

EXAMPLE 9.27 (CONTINUED)

```
2          while(email_addr.indexOf("@") == -1 ){
3             alert( "Invalid email address.");
              email_addr=prompt("What is your email address? ","");
            }
            document.write("<br />OK.<br />");
        </script>
        </big>
        </font>
      </body>
    </html>
```

EXPLANATION

1 The user is prompted for his or her e-mail address and the input is assigned to a string called *email_addr*.

2 The loop expression uses the *indexOf() String* method to see if there is an @ symbol in the e-mail address. If there isn't, the *indexOf()* method returns –1 and the body of the loop is executed.

3 If the *indexOf()* method didn't find the @ substring, the alert box appears and the user is prompted again (see Figures 9.31 and 9.32). The loop terminates when the user enters an e-mail address containing an @ sign. Of course, this is just a simple test for validating an e-mail address; more elaborate methods of validation are discussed in Chapter 17, "Regular Expressions and Pattern Matching."

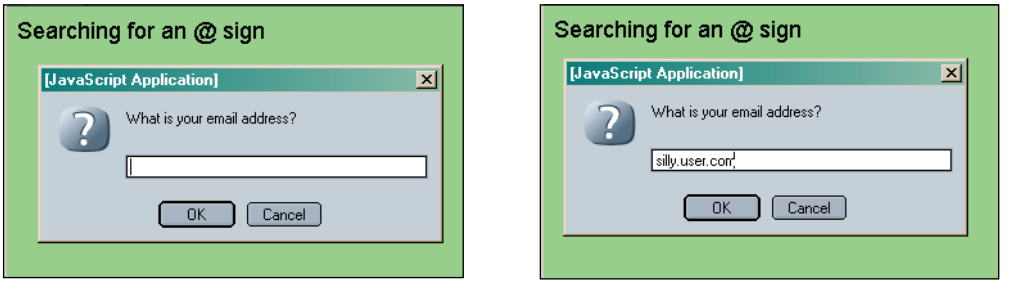


Figure 9.31 Using the *indexOf() String* method.

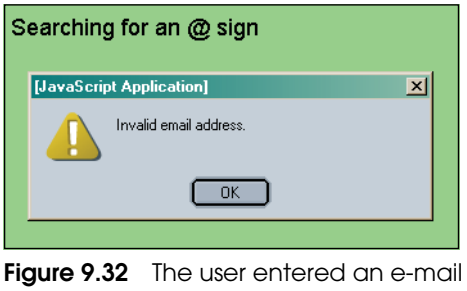


Figure 9.32 The user entered an e-mail address without the @ symbol.

EXAMPLE 9.28

```
<html>
  <head><title>String Manipulation</title></head>
  <body>
    <h2>Working with String Manipulation Methods</h2>
    <script type="text/javascript">
      function break_tag(){
        document.write("<br />");
      }
      document.write("<h3>");
1     var str1 = new String("The merry, merry month of June...");
      document.write("In the string:<em> "+ str1 );
2     document.write("</em> the first 'm' is at position " +
                     str1.indexOf("m"));
      break_tag();
3     document.write("The last 'm' is at position " +
                     str1.lastIndexOf("m"));
      break_tag();
4     document.write("<em>str1.substr(4,5)</em> returns<em> " +
                     str1.substr(4,5));
      break_tag();
      document.write(str1.toUpperCase());
      document.write("</h3>");
    </script>
  </body>
</html>
```

EXPLANATION

1 A new *String* object is created with the *String()* constructor.

2 The *indexOf() String* method returns the index value where "m" is first encountered in the string starting at the first character, position 0.

3 The *lastIndexOf()* method returns the index position of the last occurrence of "m" in the string starting from the left from position 0.

4 Starting at position 4, the *Substr* method returns 5 characters.

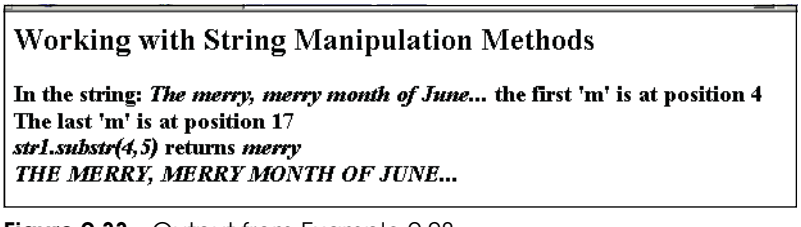


Figure 9.33 Output from Example 9.28.

Methods That Extract Substrings from a String. You might have to do more than just find a substring within a string; you might need to extract that substring. For example, we found the @ in the e-mail address, now we might want to get just the user name or the server name or domain name. To do this, JavaScript provides methods such as *splice()*, *split()*, *charAt()*, *substr()*, and *substring()*.

EXAMPLE 9.29

```
<html>
  <head><title>Extracting Substrings</title></head>
  <body bgcolor=lightgreen>
    <font face="arial">
    <big>Extracting substrings</big>
    <small>
    <script type="text/javascript">
1     var straddr = "DanielSavage@dadserver.org";
      document.write("<br />His name is<em> " +
2     straddr.substr(0,6) + "</em>.<br />");
3     var namesarr = straddr.split("@" );
4     document.write( "The user name is<em> " +  namesarr[0] +
                      "</em>.<br />");
5     document.write( "and the mail server is<em> " +  namesarr[1]
                      + "</em>.<br />");
6     document.write( "The first character in the string is <em>"
                      + straddr.charAt(0)+ "</em>.<br />");
7     document.write( "and the last character in the string is <em>"
                       + straddr.charAt(straddr.length - 1)
                       + "</em>.<br />");
    </script>
    </small>
    </font>
  </body>
</html>
```

EXPLANATION

1 A string is assigned an e-mail address.

2 The *substr()* starts at the first character at position 0, and yanks 6 characters from the starting position. The substring is *Daniel*.

3 The *split()* method creates an array, called *namesarr*, by splitting up a string into substrings based on some delimiter that marks where the string is split. This string is split using the @ sign as its delimiter.

4 The first element of the array, *namesarr[0]*, that is created by the *split()* method is *DanielSavage*, the user name portion of the e-mail address.

5 The second element of the array, *namesarr[1]*, that is created by the *split()* method is *dadserver.org*, the mail server and domain portion of the e-mail address.

Continues