Julie C. Meloni

SIXTH EDITION

Sams Teach Yourself

PHP, MySQL & JavaScript All in One

Pearson

Julie C. Meloni

Sams **Teach Yourself**

PHP, MySQL & JavaScript

All in One

SIXTH EDITION

Pearson

The value in the parentheses specifies a character to separate the parts of the array. In this case, a space is used, resulting in the final string `John Q. Public`. If you do not specify a character, commas are used.

Sorting a String Array

JavaScript also includes a `sort` method for arrays that returns an alphabetically sorted version of the array. For example, the following statements initialize an array of four names and sort it:

```
names[0] = "Public, John Q.";
names[1] = "Doe, Jane";
names[2] = "Duck, Daisy";
names[3] = "Mouse, Mickey";
sortednames = names.sort();
```

The last statement sorts the `names` array and stores the result in a new array, `sortednames`.

Sorting a Numeric Array

Because the `sort` method sorts alphabetically, it won't work with a numeric array—at least not the way you'd expect. If an array contains the numbers 4, 10, 30, and 200, for example, it would sort them as 10, 200, 30, 4—not even close. Fortunately, there's a solution: You can specify a function in the `sort` method's parameters, and that function is used to compare the numbers. The following code sorts a numeric array correctly:

```
function numbercompare(a,b) {
   return a-b;
}
numbers = new Array(30, 10, 200, 4);
sortednumbers = numbers.sort(numbercompare);
```

This example defines a simple function, `numbercompare`, that subtracts the two numbers. After you specify this function in the `sort` method, the array is sorted in the correct numeric order: 4, 10, 30, 200.

NOTE

JavaScript expects the comparison function to return a negative number if `a` belongs before `b`, `0` if they are the same, or a positive number if `a` belongs after `b`. This is why `a-b` is all you need for the function to sort numerically.

▼ TRY IT YOURSELF

Sorting and Displaying Names

To gain more experience working with JavaScript's string and array features, you can create a script that enables the user to enter a list of names and then displays the list in sorted form.

Because this will be a larger script, you will create separate HTML and JavaScript files. First, the `sort.html` file will contain the HTML structure and form fields for the script to work with. Listing 7.2 shows the HTML document.

LISTING 7.2 The HTML Document for the Sorting Example

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <title>Array Sorting Example</title>
    <script type="text/javascript" src="sort.js"></script>
  </head>

  <body>
     <h1>Sorting String Arrays</h1>
     <p>Enter two or more names in the field below,
     and the sorted list of names will appear in the
     textarea.</p>
     <form name="theform">
     Name:
     <input type="text" name="newname" size="20">
     <input type="button" name="addname" value="Add"
     onclick="SortNames();">
     <br/>
     <h2>Sorted Names</h2>
     <textarea cols="60" rows="10" name="sorted">
     The sorted names will appear here.
     </textarea>
     </form>
  </body>
</html>
```

Because the script will be in a separate document, the `<script>` tag in the header of this document uses the `src` attribute to include a JavaScript file called `sort.js`. You will create this file next.

This document defines a form named `theform`, a text field named `newname`, an `addname` button, and a text area named `sorted`. Your script will use these form fields as its user interface.

Listing 7.3 provides the JavaScript necessary for the sorting process.

LISTING 7.3 The JavaScript File for the Sorting Example

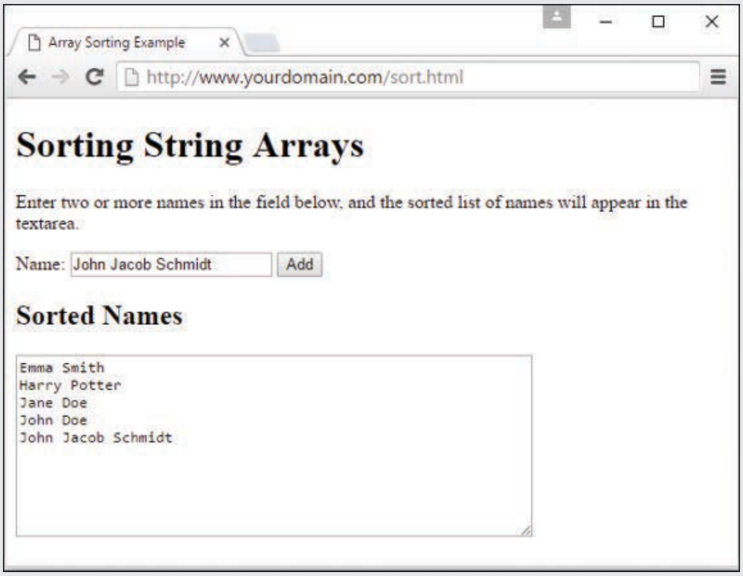
```
// initialize the counter and the array
var numbernames=0;
var names = new Array();
function SortNames() {
   // Get the name from the text field
   thename=document.theform.newname.value;
   // Add the name to the array
   names[numbernames]=thename;
   // Increment the counter
   numbernames++;
   // Sort the array
   names.sort();
   document.theform.sorted.value=names.join("\n");
}
```

The script begins by defining two variables with the `var` keyword: `numbernames` is a counter that increments as each name is added, and the `names` array stores the names.

When you type a name into the text field and click the button, the `onclick` event handler calls the `SortNames` function. This function stores the text field value in a variable, `thename`, and then adds the name to the `names` array using `numbernames` as the index. It then increments `numbernames` to prepare for the next name.

The final section of the script sorts the names and displays them. First, the `sort()` method is used to sort the `names` array. Next, the `join()` method is used to combine the names, separating them with line breaks, and display them in the text area.

To test the script, save it as `sort.js` and then load the `sort.html` file you created previously into a browser. You can then add some names and test the script. Figure 7.2 shows the result after several names have been sorted.



Sorting String Arrays

Enter two or more names in the field below, and the sorted list of names will appear in the textarea.

Name: John Jacob Schmidt [Add]

Sorted Names

```
Emma Smith
Harry Potter
Jane Doe
John Doe
John Jacob Schmidt
```

FIGURE 7.2
The output of the name-sorting example.

Summary

In this chapter, the lessons focused on variables and how JavaScript handles them. You learned how to name variables, how to declare them, and the differences between local and global variables. You also explored the data types supported by JavaScript and how to convert between them.

You also learned about JavaScript's more complex variable types—strings and arrays—and looked at the features that enable you to perform operations on them, such as converting strings to uppercase and sorting arrays. Not only is all of the information in this chapter useful as foundational JavaScript knowledge, but the topics covered are conceptually similar to those you'll learn and practice in the foundational PHP chapters later in this book.

In the next chapter, you'll continue your foundational JavaScript education by learning more about three additional key features: functions, objects, and flow control.

Q&A

Q. What is the importance of the `var` keyword? Should I always use it to declare variables?

A. You only need to use `var` to define a local variable in a function. However, if you're unsure at all, it's always safe to use `var`. Using it consistently will help you keep your scripts organized and error free.

Q. Is there any reason I would want to use the `var` keyword to create a local variable with the same name as a global one?

A. Not on purpose. The main reason to use `var` is to avoid conflicts with global variables you might not know about. For example, you might add a global variable in the future, or you might add another script to the page that uses a similar variable name. This is more of an issue with large, complex scripts.

Q. What good are Boolean variables?

A. Often in scripts you'll need a variable to indicate whether something has happened—for example, whether a phone number the user has entered is in the right format. Boolean variables are ideal for this; they're also useful in working with conditions, as you'll see in Chapter 8.

Q. Can I store other types of data in an array? For example, can I have an array of dates?

A. Absolutely. JavaScript enables you to store any data type in an array.

Q. What about two-dimensional arrays?

A. These are arrays with two indexes (such as columns and rows). JavaScript does not directly support this type of array, but you can use objects to store more complex data.

Workshop

The Workshop contains quiz questions and exercises to help you solidify your understanding of the material covered. Try to answer all questions before looking at the "Answers" section that follows.

Quiz

1. Which of the following is *not* a valid JavaScript variable name?

   A. `2names`

   B. `first_and_last_names`

   C. `FirstAndLast`

2. If the statement `var fig=2` appears in a function, which type of variable does it declare?
   - **A.** A global variable
   - **B.** A local variable
   - **C.** A constant variable
3. If the string `test` contains the value `The eagle has landed.`, what would be the value of `test.length`?
   - **A.** `4`
   - **B.** `21`
   - **C.** `The`
4. Using the same sample string, which of these statements would return the word `eagle`?
   - **A.** `test.substring(4,9)`
   - **B.** `test.substring(5,9)`
   - **C.** `test.substring("eagle")`
5. What will be the result of the JavaScript expression `31 + " angry polar bears"`?
   - **A.** An error message
   - **B.** 32
   - **C.** "31 angry polar bears"

Answers

1. **A.** `2names` is an invalid JavaScript variable name because it begins with a number. The others are valid, although they're probably not ideal choices for names.
2. **B.** Because the variable is declared in a function, it is a local variable. The `var` keyword ensures that a local variable is created.
3. **B.** The length of the string is 21 characters.
4. **A.** The correct statement is `test.substring(4,9)`. Remember that the indexes start with `0` and that the second index is noninclusive.
5. **C.** JavaScript converts the whole expression to the string `"31 angry polar bears"`. (No offense to polar bears, who are seldom angry and rarely seen in groups this large.)

Exercises

- Modify the sorting example in Listing 7.3 to convert the names to all uppercase before sorting and displaying them.
- Modify Listing 7.3 to display a numbered list of names in the textarea.