Phil Ballard

Seventh Edition

Sams Teach Yourself

JavaScript™ in 24 Hours

Pearson

Phil Ballard

Sams **Teach Yourself**

JavaScript

Seventh Edition

in 24 **Hours**

Pearson

Combining Multiple Arrays

Suppose you have this array:

```
var array1 = ['apple', 'banana', 'pear'];
```

You want to declare another array that includes but extends `array1`. It would be straightforward to declare your new array like this:

```
var array2 = ['orange', 'cherry', 'fig'];
```

Then you would combine the two arrays with a method such as `concat()`, which you've already seen. But if you want to position the second array *at a specific point* in the first, you then have more work to do. Instead, you can simply do this:

```
var array2 = ['orange', ...array1, 'cherry', 'fig'];
```

When you examine `array2`, you'll find it now contains

```
['orange', 'apple', 'banana', 'pear', 'cherry', 'fig'];
```

Calling Functions with an Array of Arguments

You can use the same notation to divide an array into a list of arguments to pass to a function or method.

Let's suppose you want to find the minimum value in a numeric array. (In this case, we use the `min()` method of JavaScript's `Math` object, though the principle works in just the same way with any other function, including the ones you define yourself.)

```
var myArray = [91, 35, 17, 101, 29, 77];
alert(Math.min(...myArray));  // alerts 17
```

Collecting Values

Finally, let's see how the three-dots notation can be used to collect values into an array.

Suppose you declare an array like this:

```
var [a, b, ...c] = [1, 2, 3, 4, 5, 6, 7, 8, 9];
```

In this case, JavaScript will allocate values for `a` and `b`, and use variable `c` to form an array of those remaining, so that `a=1`, `b=2`, and `c=[3, 4, 5, 6, 7, 8, 9]`.

Summary

An array is a convenient means of storing multiple values in a single variable.

In this lesson you learned about some of the methods of creating JavaScript array objects and manipulating them using the language's array methods.

Finally, you learned about three-dots notation and how it can be used to collapse data into arrays or expand data out of an array.

Q&A

Q. Does JavaScript allow associative arrays?

A. JavaScript does not directly support associative arrays (arrays with named indexes). However, there are ways to simulate their behavior by using objects. You'll see examples of this later in the book.

Q. Can I create a multidimensional array in JavaScript?

A. You can create an array of arrays, which amounts to the same thing:

```
var myArray = [[1,2], [3,4], [5,6]];

alert(myArray[1][0]); // alerts '3'
```

Workshop

Quiz questions and exercises to test your understanding and to stretch your skills.

Quiz

1. If the element with highest index in array `Foo` is `Foo[8]`, what value will be returned by `Foo.length`?

   a. 7

   b. 8

   c. 9

2. You have an array called `monthNames` containing the names of all the months of the year. How would you use `join()` to create a string name containing all these month names with a single space between names?

   a. `var names = monthNames.join();`

   b. `var names = monthNames.join(" ");`

   c. `var names = monthNames.join(\s);`

3. What value will be returned by `indexOf()` if it is passed a value that does not appear in the array to which it is applied?

   a. `null`

   b. `undefined`

   c. −1

4. What array method would you use to delete a particular index from an array?
   - a. `indexOf()`
   - b. `slice()`
   - c. `splice()`
5. What is returned when the `toString()` method is used on an array?
   - a. A single string composed of the array elements joined by commas
   - b. A single string composed of the array elements joined by spaces
   - c. A single string composed of the array elements joined with no separating characters

TIP

Register Your Book

Just a reminder for those of you reading these words in the print or e-book edition of this book: If you go to www.informit.com/register and register this book (using ISBN 978-0-672-33809-0), you can receive free access to an online Web Edition that not only contains the complete text of this book but also features an interactive version of this quiz.

Answers

1. c. `Foo.length` will return 9.
2. b. `var names = monthNames.join(" ");`
3. c. It will return –1.
4. c. `splice()`
5. a. A single string composed of the array elements joined by commas

Exercises

Review the array and string methods that share a method name. Familiarize yourself with how the syntax and operation change depending on whether these methods are applied to a string or an array.

Using the three-dots notation, write a function that takes as its argument an arbitrary number of numeric arrays and returns the sum of the elements from all the arrays. (For the purpose of this exercise, you can assume that type-checking of the input data is not required.)

Test your function in the JavaScript console.

LESSON 9

Handling Events in JavaScript

What You'll Learn in This Lesson:

- ▶ What we mean by events
- ▶ What event handlers are and what they do
- ▶ The different ways of adding event handlers
- ▶ How to use the `event` object
- ▶ Event bubbling and capturing

Some JavaScript programs begin their execution at the first line of code and then carry on through, line by line, oblivious to anything else that may be happening until they complete. Often, though, we want our programs to react to the things that happen in their environment—to the user clicking on a page element, perhaps, or the loading of an image being completed. We call these occurrences *events*, and JavaScript is equipped to detect and act upon a long list of them.

Types of Events

It's easiest to parcel up the possible events into groups. The events you are perhaps most likely to deal with on a regular basis are mouse events, keyboard events, DOM object events, and form events. Tables 9.1 to 9.4 summarize the more commonly used events in each of these categories.

NOTE

Other Types of Events

Many other types of events are available in JavaScript, relating to activities such as drag and drop, clipboard use, printing, animation, and so on. You may see some of these events crop up in other lessons, even if they're not covered here. The basic principles are, of course, just the same.

TABLE 9.1 Mouse Events

Event	Occurs When...
`onclick`	The user clicks on an element
`oncontextmenu`	The user right-clicks on an element to open a context menu
`ondblclick`	The user double-clicks on an element
`onmousedown`	The user presses a mouse button over an element
`onmouseenter`	The pointer is moved onto an element
`onmouseleave`	The pointer is moved out of an element
`onmousemove`	The pointer is moving while it is over an element
`onmouseover`	The pointer is moved onto an element or onto one of its children

TABLE 9.2 Keyboard Events

Event	Occurs When...
`onkeydown`	The user is pressing a key
`onkeypress`	The user presses a key
`onkeyup`	The user releases a key

TABLE 9.3 DOM Object Events

Event	Occurs When...
`onerror`	An error occurs while loading an external file
`onload`	An object has loaded
`onresize`	The document view is resized
`onscroll`	An element's scrollbar is being scrolled

TABLE 9.4 Form Events

Event	Occurs When...
`onblur`	An element loses focus
`onchange`	The content, selection, or checked state has changed
`onfocus`	An element gets focus
`onreset`	A form is reset
`onselect`	The user selects some text
`onsubmit`	A form is submitted

TIP

Get a Complete List of Events

The complete list of events handled by JavaScript is huge, and I don't intend to cover all of them here. Instead, see www.w3schools.com/jsref/dom_obj_event.asp for a more complete list.

Event Handlers

So what do we mean when referring to an *event handler*? Put simply, an event handler is a piece of code that is executed when a specified event is detected by JavaScript. Here are a few possible examples:

- When the user's mouse hovers over the enter button, change the button's color.
- When the user presses the P key, pause the operation of the program.
- When the page finishes loading, make the menu elements visible.

There are several different ways to add event handlers to your programs. Let's look at each of them in turn, starting with the oldest and simplest.

Inline Event Handlers

When JavaScript was first introduced to web pages, event handlers were generally added inline to page elements. Inline event handlers usually take this form:

```
handlername = "JavaScript code"
```

They also are usually inserted in the opening tag of an HTML element. Let's look at an example:

```
<a href="https://www.w3.org/" onclick="alert('hello W3C!')">World Wide Web
Consortium (W3C)</a>
```

TIP

Different Strokes

Different event handlers work with various HTML tags. While `onclick` can be inserted into many HTML tags, handlers like `onload` work only with `<body>` and `<img>` elements.

Event Handlers as Properties of DOM Objects

This method of assigning event handlers works okay for trivial examples, but it has some major disadvantages. The main one is that it mixes up how things *look* (the presentation layer, as it's