Book 5: JavaScript - Arrays

Book for Weeks 10-11:JavaScript Arrays

Site:	Insight2 @ CCSF
Course:	CNIT133-META-Interactive Web Pages-Rubin-JavaScri-Spring 2014
Book:	Book 5: JavaScript - Arrays
Printed by:	Thuong Ho
Date:	Tuesday, March 25, 2014, 11:54 PM

Table of contents

1 Arrays

Textbook References

Flanagan book: Ch. 7

Gosselin book: p. 136-140, 407-421, Ch. 5

McDuffie book: Chapter 7, 13

Deitel book: Chapter 10

An array is a collection of data. In JavaScript, an ARRAY is also an object.

Think of an array like a chest of drawers. Each drawer in the chest is numbered, and each drawer may contain data, whether that data be a string, a number, a boolean, an object, or another array. You access the data in the "drawers" of an array via the "drawer" number.

Here is a conceptual picture of an array:

array image

In the above illustration, I have created an array named fred. The array, fred, has four "drawers" or positions, numbered sequentially starting from 0 (array positions always start at 0). In the 0 position (the first drawer) of fred, there is a string, "Hi". In the 1 position (the second drawer) of fred, there is a number, 127. In the 2 position (the third drawer) of fred, there is a boolean value, true. In the 3 position (the fourth drawer) of fred, there is another string, "Wow".

To declare fred in JavaScript, I must instanciate an Array object using the Array() constructor.

Example:

```
var fred = new Array();
```

There are several ways that I could then initialize the contents of the positions in fred. The most characteristic means of doing so is by using the ARRAY ACCESS OPERATOR (**[]**).

Example:

```
var fred = new Array();
fred[0] = "Hi";
fred[1] = 127;
fred[2] = true;
fred[3] = "Wow";
```

I place the array access operator (**[]**) next to the name of the pertinent array (fred), and put the number of the array position that I want between the square-braces of the operator. I use the "gets" or "assignment" operator to put information into each of the array positions. Again, any type of data may be placed into an array position, including objects or other arrays.

I could then PULL or EXTRACT information from a position in the array using the same array access operator (**[]**).

Example (abbreviated)

```
var myData = fred[0];
```

In the above example, I have pulled information from the array fred at its 0 position, and placed that data into the variable, myData. If this array were the same as the one in the previous example, myData would now contain the string, "Hi".

Note on pronunciation: fred[0] would be referred to as "fred sub zero", **fred[1]** would be referred to as "fred sub one", etc.

There is a second way to initialize arrays which involves passing arguments to the Array() constructor during the declaration of the array.

Example:

```
var fred = new Array("Hi", 127, true, "Wow");
```

As you can see in the above example, I have passed all of my initial values for the array as arguments of the Array constructor; this is a terrific shorthand for initializing arrays quickly and easily.

Arrays are a very powerful data storage mechanism which, like loops, are a whole world unto themselves. A JavaScript programmer will be called upon to create arrays in profusion to store every kind of data imaginable. In fact, virtually every element on an HTML page is stored in some sort of an array, which you may then access via JavaScript.

2 Arrays and Loops

Arrays and Loops

Because arrays hold information in numeric sequence, loop logic is frequently used to read, rewrite, and/or otherwise change the data stored in an array.

There is one property of the Array object that you need to know about in order to use loop logic successfully when accessing arrays of an uncertain size: the length property. The length property of an instance of the Array object returns a number which is the number of positions of data stored in the array. Access the length property using dot syntax attached to the instance of the array object in question.

Example (assuming that fred is an instance of an Array object):

```
fred.length
```

Example (in context):

```
var fred = new Array("Hi", 127, true, "Wow");
alert(fred.length);
```

In the above example, fred.length returns a value of 4, the number of positions in the fred array. The alert method, then, would display a value of 4 because we passed fred.length as an argument to the alert method.

Using loop logic and the length property, we could access all of the information in fred.

Example:

```
var fred = new Array("Hi", 127, true, "Wow");

for (myCounter = 0; myCounter < fred.length; myCounter++) {
 var myData = fred[myCounter];
 alert(myData);
};
```

In the above example, we have set the FOR loop to run as long as myCounter is less than the length of fred, which will cause this particular loop to run 4 times (as long as myCounter is either 0, 1, 2, or 3. Now, since fred.length is equal to 4, as soon as myCounter gets to 4, the condition will become false, and the loop will end).

Inside the loop, I am using myCounter as a record of the position in the array that I am interested in. The first time through the loop, myCounter is equal to 0, so I will be pulling information from fred[0] and putting that into myData. I am then passing the contents of myData ("Hi") to the alert method, which will say "Hi". The next time through the loop, myCounter will be equal to 1, so I will be pulling information in my first line of code from fred[1] (the number 127), and putting that number into myData, etc etc.

We could remove one step from the above example by passing the return value of fred[myCounter] directly as an argument to the alert method, skipping the use of the variable, myData.

Example:

```
var fred = new Array("Hi", 127, true, "Wow");

for (myCounter = 0; myCounter < fred.length; myCounter++) {
 alert(fred[myCounter]);
};
```

Here is an example of this code in action. As an exercise, you may wish to recreate the function from this example, and see if you can get it to work yourself.

Writing your own arrays and loops for real world purposes is fairly difficult and requires a good bit of experience. A good portion of elements on an HTML page are declared, initialized, and stored in JavaScript using arrays. You will be using loops frequently to access pre-existing arrays, and either pulling information out of an array, or affecting the contents of an array in some manner.

Every image on a web page, for instance, is stored, in order of their appearance in the HTML code, in the **images[]** array; the first IMG on the HTML page is stored in **images[0]**, the second IMG on the HTML page is stored in **images[1]**, etc. You can affect the images on an HTML page by accessing their image object instances via this array (there are also other, more direct means, as well). Countless other HTML elements are stored in a similar manner.

3 Form Pulldown Menus Used as Navigation Elements

Form Pulldown Menus Used as Navigation Elements

In this section, I am going to talk about "submitting" a FORM to a JavaScript function, then extracting information from that form using dot syntax and a number of specialized keywords. To demonstrate this process, we are going to look at a FORM pulldown menu used as a navigational element.

Here is an example of what I mean.

You should already know how to create form pulldown menus, and how to use the VALUE attribute of the individual OPTION tags to store URL information.

Example (abbreviated):

```
<option value="http://www.yahoo.com/">Yahoo</option>
```

For the purposes of this demonstration, I am going to name my SELECT tag "destList"; I do NOT need to name my FORM tag.

Example:

```
<form action="">
<select name="destList">
<option value="http://www.yahoo.com/">Yahoo</option>
<option value="http://www.w3.org/">W3C</option>
<option value="http://www.mozilla.org/">Mozilla</option>
<option value="http:/fog.ccsf.org/~srubin">Steve Rubin</option>
</select>
</form>
```

When you "submit" your form to a JavaScript function, you will NOT use the submit INPUT; instead, you will use the generic button INPUT. You will pass the value, this.form, as the argument to your function, which will, essentially, pass a complete copy of the form to the function.

Note: the this keyword, and the value, this.form below. It turns out that, in this particular case, the this keyword refers to the button, and the form property of the button is really a special keyword referring to the entire form attached to the button. The phrase this.form, then, can be passed as an argument to a function; the reference encompasses the entire form enclosed by the FORM tag attached to the given form element.

The following example presumes that the function I'm about to create is called goToNewPage().

Example (abbreviated):

```
<input type="button" value="Go" onclick="goToNewPage(this.form);">
```

Example (in context):

```
<form action="">
<select name="destList">
<option value="http://www.yahoo.com/">Yahoo</option>
<option value="http://www.w3.org/">W3C</option>
<option value="http://www.mozilla.org/">Mozilla</option>
<option value="http://fog.ccsf.org/~srubin">Steve Rubin</option>
</select>

<input type="button" value="Go" onclick="goToNewPage(this.form);">
</form>
```

Now, I'm ready to create my function; the function will have one local argument variable, myForm, which will contain the reference to the form (passed using this.form as an argument from the event handler).

Example:

```
function goToNewPage(myForm) {
// lines of code here...
}
```

In this function, I'm going to need to look at the destList SELECT tag within the form; this can be done using dot syntax.

Example:

```
myForm.destList
```

Now, I'm going to need to look at the options[] array within the destList SELECT tag. All of the OPTION tags in a SELECT tag are stored in an options[] array. This, too, will be accessed using dot syntax.

Example:

```
myForm.destList.options[]
```

What I really need to see, however, is the value property of the desired option in the options[] array. The value property of the option contains the URL string that we've placed in the VALUE attribute.

Example:

```
myForm.destList.options[].value
```

The problem with this syntax is that we do not know WHICH option we want!

In the options[] array, as you might expect, the FIRST option (in this case, the option for Yahoo) is stored in options[0], the SECOND option (the option for the W3C) is stored in options[1], etc. If I want to get the URL value for Mozilla, then, I would need to look at options[2] in this example.

Example:

```
myForm.destList.options[2].value
```

The above example will return the string value, "http://www.mozilla.org/".

Alright, this hard-coding is good enough, in its way, but HOW do I find out WHICH option the USER has selected?

There is a special keyword of the SELECT tag, selectedIndex, which returns the number of the option which the user has selected.

Example:

```
myForm.destList.selectedIndex
```

I need to put the above code INTO the array access operator ([]) for the options[] array to access the selected option.

Example:

```
myForm.destList.options[myForm.destList.selectedIndex].value
```

I would then place this extracted information into some sort of local variable.

Example:

```
var mydest = myForm.destList.options[myForm.destList.selectedIndex].value;
```

Once I have extracted the URL string value that I require, I will need to ASSIGN that value to the browser window's location object. The location property of the window instance of the Window object gives us access to the location bar, and will advance the page in the web browser when assigned to the desired URL.

Example:

```
window.location = "http://www.mozilla.org/";
```

Example:

```
var mydest = myForm.destList.options[myForm.destList.selectedIndex].value;
window.location = mydest;
```

Example (complete function):

```
function goToNewPage(myForm) {
var mydest = myForm.destList.options[myForm.destList.selectedIndex].value;
window.location = mydest;
}
```

In the example cited earlier, there was also a pulldown menu which changed the page location WITHOUT hitting a "Go" button. This functionality was achieved using the onchange event handler, which is triggered when the state of a form element is changed by the user; the onchange event handler would be added to the SELECT tag for the pulldown menu.

Example:

```
<select name="destList" onchange="goToNewPage(this.form);">
<option value="http://www.yahoo.com/">Yahoo</option>
<option value="http://www.w3.org/">W3C</option>
<option value="http://www.mozilla.org/">Mozilla</option>
<option value="http://fog.ccsf.org/~srubin">Steve Rubin</option>
</select>
```

In the above example, the Yahoo option is pre-selected. If the user wishes to go to Yahoo, then, they will NOT be able to do so, because the onchange event handler is ONLY called when the state of the pulldown menu is CHANGED.

To get around this problem, web programmers add some sort of blank option to the pulldown menu; this blank option usually tells the user what they ought to do with the pulldown menu.

Example:

```
<select name="destList" onchange="goToNewPage(this.form);">
```

```
<option>Choose dest:</option>
<option value="http://www.yahoo.com/">Yahoo</option>
<option value="http://www.w3.org/">W3C</option>
<option value="http://www.mozilla.org/">Mozilla</option>
<option value="http://fog.ccsf.org/~srubin">Steve Rubin</option>
</select>
```

That's it! Here's the example page again. I strongly urge you to try typing out this code for yourself, by hand, to get a feel for the process.

4 Web Tutorials and Examples - Arrays

pulldown menus

- http://blazonry.com/javascript/js_menu.php

Form Validation

- http://corpocrat.com/2009/07/15/quick-easy-form-validation-tutorial-with-jquery/
- http:// www.elated.com/articles/form-validation-with-javascript/

arrays

- http:// www.pageresource.com/jscript/jarray.htm
- http://www.w3schools.com/js/js_obj_array.asp
- http:// wsabstract.com/javatutors/loop1.shtml

two-dimensional arrays

- http://www.javascriptkit.com/javatutors/twoarray1.shtml

5 Short Exercises - Arrays

1. Fill in the blanks in each of the following statements:

a) JavaScript stores lists of values in _____.

Ans: arrays.

b) The names the four elements of array p are ___, ___,
_____, and ______.

Ans: p[0], p[1], p[2], p[3].

c) In a two-dimensional array, the first subscript identifies
the _____ of an element, and the second subscript identifies the
of an element.

Ans: row, column.

d) An m-by-n array contains ___ rows, ___ columns and
_____elements.

Ans: m, n, m * n.

e) The name of the element in row 3 and column 5 of
array d is ______.

Ans: d[3][5].

f) The name of the element in the third row and fifth column of
array d is ______.

Ans: d[2][4]

2. State whether each of the following is true
or false. If false, explain why.

a) To refer to a particular location or element within an array, we
specify the name of the array and the value of the particular element.

Ans: False, we specify the name of the array
and the indexed location to refer to a specific
element.

b) A variable declaration reserves space for an array.

Ans: True.

c) To indicate that 100 locations should be reserved for
integer array p, the programmer
should write the declaration
p[100];

Ans: False. var p = new Array[100];

d) A JavaScript program that initializes the elements of a 15-element array to zero must contain at least one for statement.

Ans: False. Arrays can be initialized upon declaration.

e) A JavaScript program that totals the elements of a two-dimensional array must contain nested for statements.

Ans: True.

3. Write JavaScript statements to accomplish each of the
following tasks:

a) Display the value of the seventh element of array f.

Ans: document.write(f[6]);

b) Initialize each of the five elements of one-dimensional array g to 8.

Ans: for (var i = 0; i < 5; i++)
g[i] = 8;

c) Total the elements of array c, which contains 100 numeric elements.

Ans: for (var i = 0; i < 100; i++)
total += parseInt(c[i]);

d) Copy 11-element array a into the first portion of array b, which contains 34 elements.

Ans: for (var i = 0; i < a.length; i++)
b[i] = parseInt(a[i]);

e) Determine and print the smallest and largest values
contained in 99-element floatingpoint array w.

Ans: var highestNumber = w[0], lowestNumber = w[0];
for (var i = 1; i < 99; i++) {
if (w[i] > highestNumber)
highestNumber = parseFloat(w[i]);
if (w[i] < lowestNumber)
lowestNumber = parseFloat(w[i]);
}
document.writeln(highestNumber + " " + lowestNumber);

4. Consider a two-by-three array t that will store integers.

a) Write a statement that declares and creates array t.

Ans: var t = new Array(2);
t[0] = new Array(3);
t[1] = new Array(3);

b) How many rows does t have?

Ans: 2.

c) How many columns does t have?

Ans: 3.

d) How many elements does t have?

Ans: 6.

e) Write the names of all the elements in the second row of t.

Ans: t[1][0], t[1][1], t[1][2].

f) Write the names of all the elements in the third column of t.

Ans: t[0][2], t[1][2].

g) Write a single statement that sets the element of t in row 1 and
column 2 to zero.

Ans: t[1][2] = 0;

h) Write a series of statements that initializes
each element of t to zero. Do not use a repetition structure.

Ans: t[0][0] = 0;
t[0][1] = 0;
t[0][2] = 0;
t[1][0] = 0;
t[1][1] = 0;
t[1][2] = 0;

i) Write a nested for statement that initializes each element of t to zero.

Ans: for (var j = 0; j < t.length; j++)
for (var k = 0; k < t[j].length; k++)
t[j][k] = 0;
or
for (var j in t)
for (var k in t[j])
t[j][k] = 0;

j) Write a series of statements that determines and prints the smallest
value in array t.

Ans: // assume small is declared and initialized
for (var x = 0; x < t.length; x++)
for (var y = 0; y < t[x].length; y++)
if (t[x][y] < small)
small = t[x][y];
document.writeln("Smallest is " + small);

k) Write a statement that displays the elements of the first row of t.
Ans: document.writeln(t[0][0] + " " + t[0][1] + " " + t[0][2]);
l) Write a statement that totals the elements of the fourth column of t.

Ans: t does not have a fourth column.

6 Coding Exercises - Arrays

Exercise 1 - Simple Array Example:

Using an array, create a webpage that contains a script that uses input text boxes to input four integers and uses a form textarea to display the inputted numbers, how many integers were less than 0 and how many were greater or equal to 0, and the product of the 4 integers. Use jQuery to click/hide the instructions and to validate the input as numeric.

>>Here is the solution.

Exercise 2a - Form Validation - input text boxes:

Create a webpage that contains a script that validates whether a user has entered correct data in form input text boxes.

>>Here is the solution.

Exercise 2b - Form Validation - input text boxes:

Create a webpage that validates whether a user has entered non-blank data in form input text boxes, using the required attribute and an input type of submit. **No script** is used for validation. A submitted form with all non-blank fields will automatically open an email program containing the form data.

>>Here is the solution.

Exercise 3 - Form Validation - radio buttons:

Create a webpage that contains a script that validates whether a user has checked one form radio button.

>>Here is the solution.

Exercise 4 - Form Validation - select items:

Create a webpage that contains a script that validates whether a user has selected a form select item.

>>Here is the solution.

Exercise 5a- Form Validation - jQuery

Create a webpage that contains a script using jQuery that validates whether a user has entered data in a form input text box, clicked a radio button, clicked a checkbox, and selected an option.

>>Here is the solution.

Here is another form using jQuery that validates whether a user has entered data in a form input text box, clicked a radio button, clicked a checkbox, and selected an option.

>>Here is the solution.

Exercise 5b- Form Validation - jQuery

Create a webpage that contains a script using jQuery that validate a form with various elements. This example is taken from JQuery: the Missing Manual - Chapter 9.

>>Here is the solution.

Note that the error messages are in red and are below each form element (in the form.css file, see #signup label.error).

Exercise 6:

Given the following table of data about several oil companies operating in the Bay Area. Create a webpage containing a script that uses a two-dimensional array that enables the user to enter an oil company name (full or abbreviated) and have the unleaded cost per gallon and comment appear in separate form text fields of the page. The user should be able to start the script by pressing the button, "Oil Company Info". Accept lower or upper case values for the oil company abbreviation or name. If the input is invalid then display an error message in the comment text box.

Use jQuery UI for a button that will display the instructions - 'Enter an Oil Company abbreviation or name (lower or upper case is ok) in the field next to the Oil Info button'.

Gas Co. Abbr	Gas Co. Name	Cost per Gal.	Comment
AR	Arco	2.99	cheap gas
BP	British Petroleum	3.15	north sea
EX	Exon	3.37	oil spill
SH	Shell Oil	3.45	rip off

>>Here is the solution.

Exercise 7:

Use a one-dimensional array to solve the following problem: Members of a club pay their dues depending on their gross income. The dues rate is $10 plus 2% of their gross income. For example, a club member whose gross income is $50000 pays $10 plus 2% of $50000, or a total of $1010. Create a webpage with a script (using an array of counters) that obtains the gross income for each member through an HTML or XHTML form and determines how many of the members pay total dues in each of the following ranges (assume that each member's gross salary is truncated to an integer amount):

a) $10-99
b) $100-999
c) $1000-4999
i) $5000 and over

>>Here is the solution.

Exercise 8:

Create a webpage that contains a script, using a one-dimensional array to solve the following problem: Read in 5 numbers via a prompt, each of which is between 1 and 10. As each number is read, print it only if it is not a duplicate number that has already been read. (Note: Print the numbers in a form textarea). Provide for case in which all 5 numbers are different. Use the smallest possible array to solve this problem. Thanks to Michael Ogi for modifying this example to include pushing numbers onto the array and appending the number to the output string.

>>Here is the solution.

Exercise 8b:

Create a webpage that contains a script, using a one-dimensional array to solve the following problem: Read in numbers into an input text box, each of which is between 1 and 10. As each number is read, print it only if it is not a duplicate number that has already been read. (Note: Print the numbers in a form textarea. The amount of numbers read in is unlimited. Thanks to Michael Ogi for creating this example.)

>>Here is the solution.

Exercise 9:

Create a webpage that contains a script, using a one-dimensional array containing the names of 3 images. The page opens with an image that is not in the array. When the user continuously clicks the image, a new image from the array should replace it. After the last image in the array is displayed, alert "End".

>>Here is the solution.