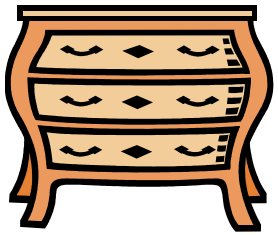
The University Of Sheffield.

COM1004: Web and Internet Technology

Lecture 10: JavaScript: Part 3

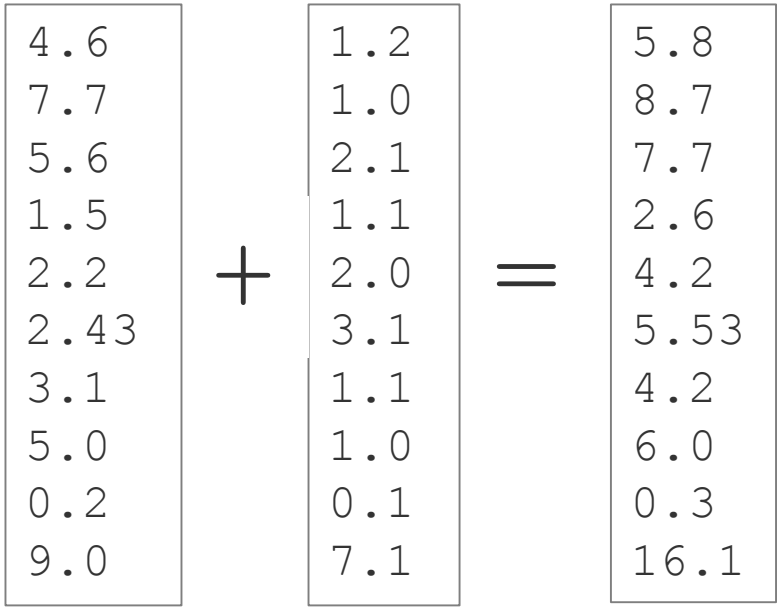


Dr. Steve Maddock
s.maddock@sheffield.ac.uk

1. Introduction

For a Web site:

- Structure using HTML
- Appearance using CSS
- Behaviour using **JavaScript**
  - Although, see recent CSS3 features, e.g. animation
- *So far:* The basics, functions, built-in objects
- *This lecture*: Arrays



2. Arrays

- A structure that can be used to collect together related items
  - E.g. a set of numbers, a set of strings, a set of Boolean values

42	12	67	87	106	43	2	6	78	5

- An indexing approach is then used to access individual items
  - The number 67 is stored at index position 2

0	1	2	3	4	5	6	7	8	9
42	12	67	87	106	43	2	6	78	5

- Arrays are ‘zero indexed’

2. Arrays

- In Java, all the items *must* be the same type, e.g. an array of integers or an array of some other type

0	1	2	3	4	5	6	7	8	9
42	12	67	87	106	43	2	6	78	5

- In JavaScript, the items can be the same type – same as above
- or different types - **This is an important difference from Java**

0	1	2	3	4
42	-89	`'www.dcs.shef.ac.uk'`	3.14	`true`

- An array can contain anything you can assign to a variable, e.g. boolean, number, string, function, object, another Array, ...

3. Creating an array in JavaScript

- There are multiple ways to create an array in JavaScript
- An array literal can be used

```
var data = [42, 12, 67];
```

- The Array object can be used in different ways

```
var data = new Array(3);
data[0] = 42;
data[1] = 12;
data[2] = 67;
```

data

0	1	2
42	12	67

```
var data = new Array(42,12,67);
```

```
var days = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'];
var mixed = [42, 'hello', 67.86, 'a', 'ahoy', -89];
```

3.1 Comparison with Java

JavaScript

```
var data = new Array(3);
data[0] = 42;
data[1] = 21;
data[2] = 67;
```

```
var d = new Array(1,3,5,7,9);
```

Java

```
int numItems = 3;
int[] data = new int[numItems];
data[0] = 42;
data[1] = 21;
data[2] = 67;
```

```
int[] d = {1, 3, 5, 7, 9};
```

4. Using an array

- Items in the array can be altered and used in expressions just like any simple variable

```
var a = 55;
var data = new Array(3);
data[0] = 42;
data[1] = 12;
data[2] = a;
data[1] = data[1] + 2 * data[0]; // data[1] now contains 96
var another = [34, data[0], data[2]*2, 11*3];
```

- *array_var*.**length** returns the length of the array

```
var days = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'];
var i = days.length-1;
alert(days[i]);  // Sun
```

5. JavaScript arrays can grow

- JavaScript will automatically increase the size of the array if new items are added:

```
var data = [];
data[0] = 21;
data[1] = data[1] * 2;
data[2] = 'data[0]';
data[7] = 34;
```

- The last line in the code is an example of what can be done, but probably should not be done
- Instead, the Array methods are preferred when increasing size
  - See later
- In contrast, in Java, array size is fixed when the array is created

example

6. Array methods

- An array is an object, so it has a set of core methods
- Some examples:

Method	Description
concat	Joins multiple arrays
forEach	Executes a specified function for each element of an Array
pop	Removes and returns the last item in the Array
push	**Adds an item to the end of the Array**
shift	Removes and returns the first item in the Array
slice	Returns a new copy of the Array using the specified index and length
sort	Sort the Array alphabetically or using the supplied function
unshift	Adds an item to the beginning of the Array

6. Array methods

- push – add one or more items on the end

```
var properties = ['red','14px','Arial'];
properties[properties.length] = 'bold';
                                // red, 14px, Arial, bold
properties.push('italic', 'underlined');
                                // red, 14px, Arial, bold, italic, underlined
```

```
var data = [];
var number;

for (var i=0; i<10; ++i) {
  number = Math.random()*10;
  data.push(number)
}
```

7. Example

- Write a JavaScript program that stores 10 random numbers in each of two arrays. A third array should then be constructed that stores the result of adding together the respective numbers in each array, as illustrated below

4.6		1.2		5.8
7.7		1.0		8.7
5.6		2.1		7.7
1.5		1.1		2.6
2.2	+	2.0	=	4.2
2.43		3.1		5.53
3.1		1.1		4.2
5.0		1.0		6.0
0.2		0.1		0.3
9.0		7.1		16.1

demo

7. Example

Note: I am using document.write() for output to simplify the code in the examples.

```
const NUM_ITEMS = 10;
var firstArray = [], secondArray = [], thirdArray = [];
var number;

// fill first array
for (var i=0; i<NUM_ITEMS; i++) {
  number = Math.random()*20;
  firstArray.push(number);
}

document.write("first Array: ");
for (var i=0; i<NUM_ITEMS; i++) {
  document.write(firstArray[i].toFixed(2) + " ");
}
document.write("<br />");
```

Fill array with data

Display array contents

first Array: 9.55 4.30 6.98 9.09 18.49 19.37 11.16 8.98 3.08 0.19

7. Example

```
//  fill second array
for (var i=0; i<NUM_ITEMS; i++) {
  number = Math.random()*20;
  secondArray.push(number);
}

document.write("second Array: ");
for (var i=0; i<NUM_ITEMS; i++) {
  document.write(secondArray[i].toFixed(2) + " ");
}
document.write("<br />");
```

Fill array with data

Display array contents

second Array: 1.90 8.56 3.99 14.65 11.94 0.92 9.40 17.18 1.84 1.50

7. Example (solution)

first Array: 9.55 4.30 6.98 9.09 18.49 19.37 11.16 8.98 3.08 0.19

second Array: 1.90 8.56 3.99 14.65 11.94 0.92 9.40 17.18 1.84 1.50

```
// calculateResults
for (var i=0; i<NUM_ITEMS; i++) {
  thirdArray[i] = firstArray[i] + secondArray[i];
}

document.write("third Array: ");
for (var i=0; i<NUM_ITEMS; i++) {
  document.write(thirdArray[i].toFixed(2) + " ");
}
document.write("<br /><br />");
```

Calculate results

Display array contents

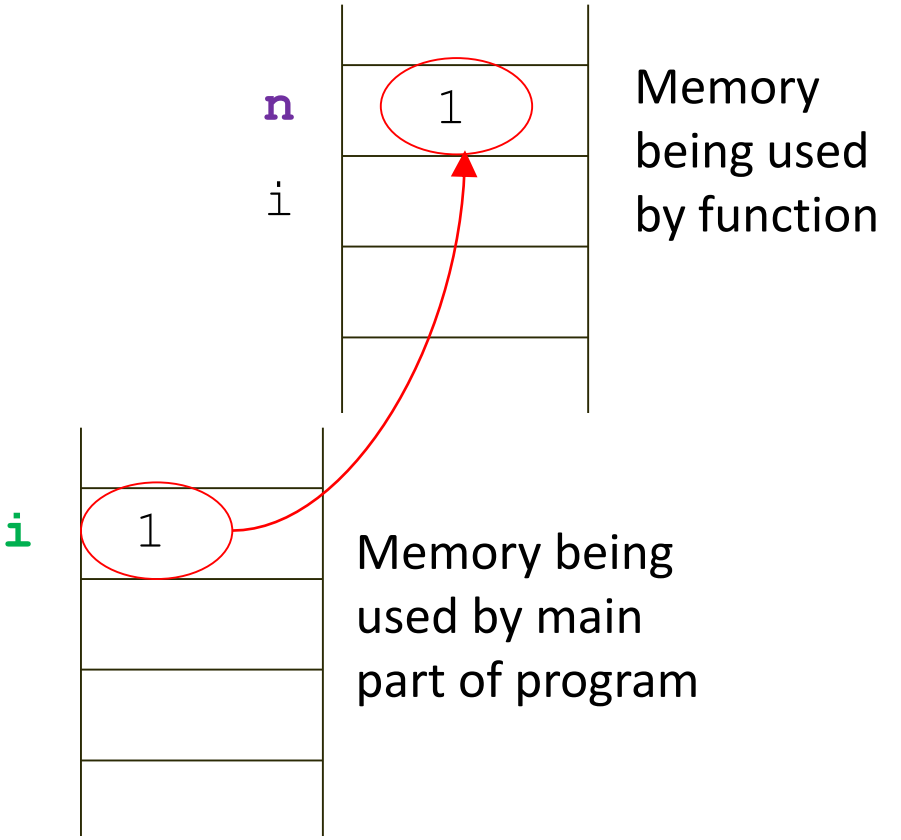
third Array: 11.45 12.86 10.96 23.75 30.43 20.29 20.55 26.16 4.92 1.69

8. Arrays and functions

- Simple types are passed *by value*
  - The value of the actual parameter is copied into the formal parameter
  - Any changes to the value stored in the formal parameter do not affect the actual parameter

```
function printStars(n) {
  for (var i=0; i<n; ++i)
    document.write("*");
  n = 1000000;  // bizarre
}

for (var i=1; i<=4; i++) {
  printStars(i);
  document.write("<br />");
}
```



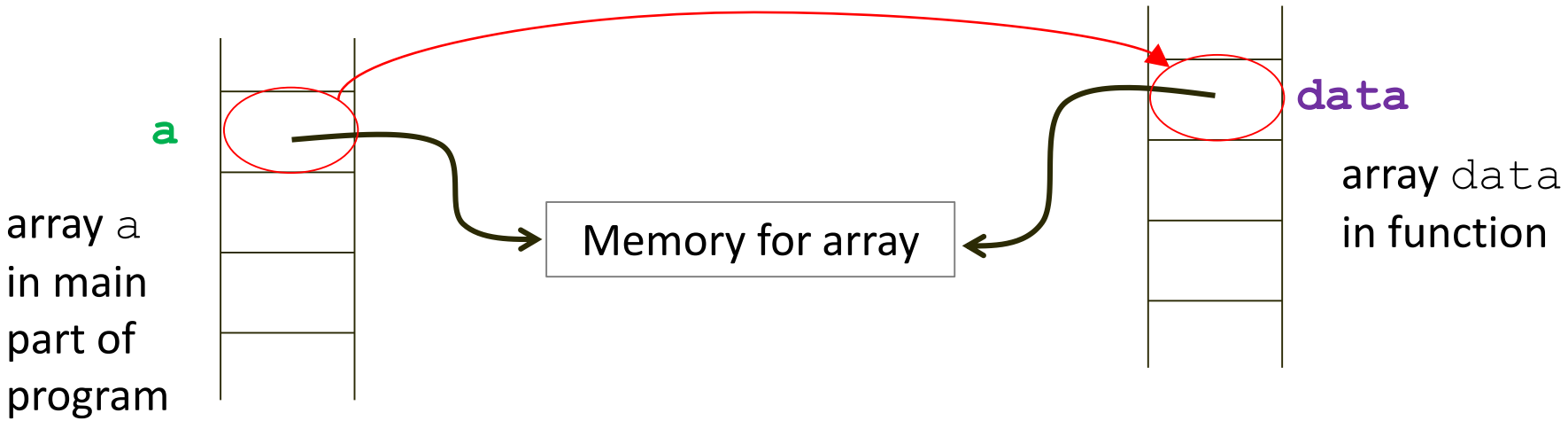
8. Arrays and functions

- Arrays are passed *by reference* to functions
  - A modification of the contents of the object referred to by the formal parameter will also affect the contents of the object referred to by the actual parameter

```
var a = [10, 20];

function f(data) {
  // change data
  data[0] = 42;
}

displayArray(a);   // 10, 20
f(a);
displayArray(a);   // 42, 20
```



8. Arrays and functions

```
const NUM_ITEMS = 10;
var firstArray = [], secondArray = [], thirdArray = [];
var number;

// fill first array
for (var i=0; i<NUM_ITEMS; i++) {
  number = Math.random()*20;
  firstArray.push(number);
}

document.write("first Array: ");
for (var i=0; i<NUM_ITEMS; i++) {
  document.write(firstArray[i].toFixed(2)
}
document.write("<br />");
```

Fill array with data

Replace with a function

Display array contents

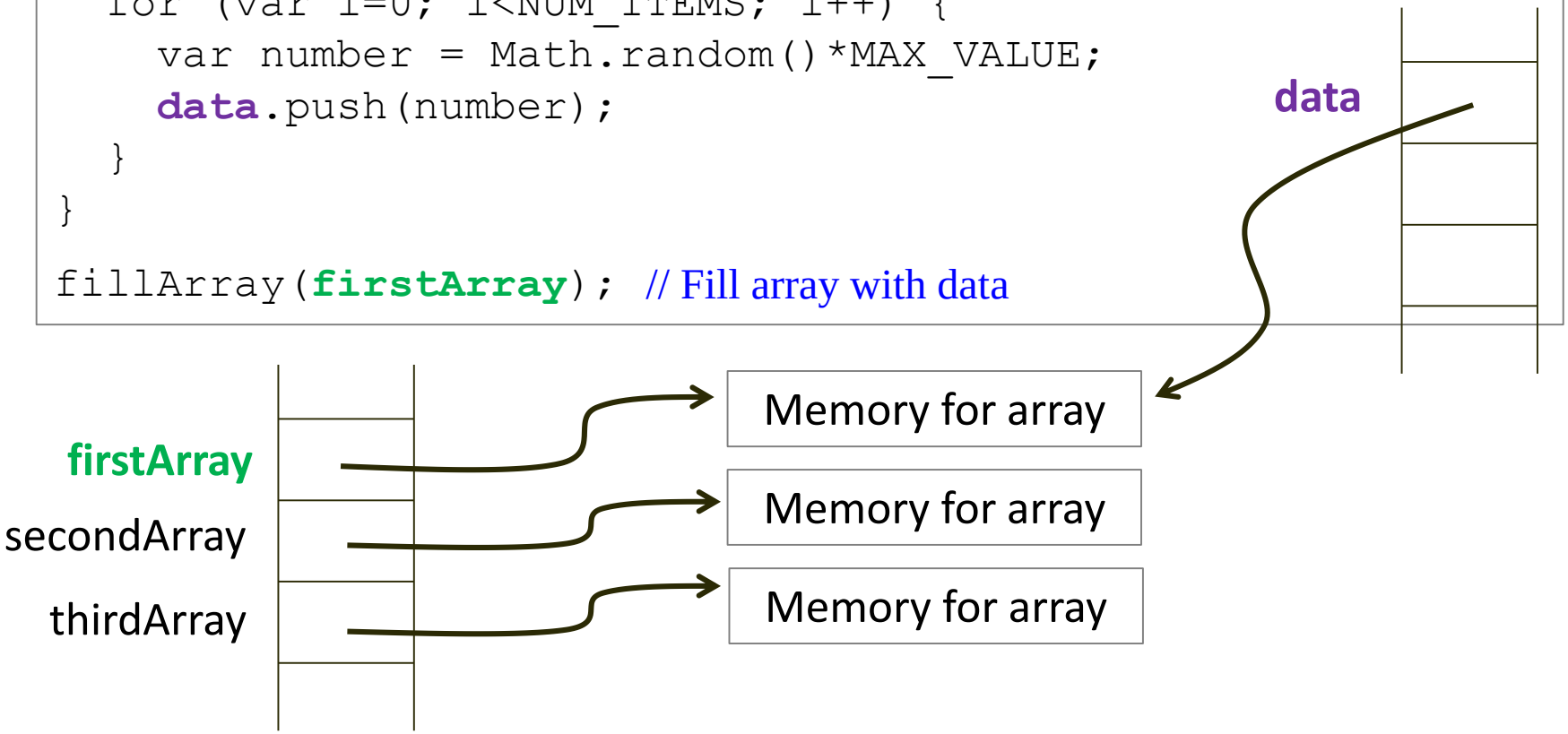
Replace with a function

8. Arrays and functions

```
const MAX_VALUE = 20;
const NUM_ITEMS = 10;
var firstArray = [], secondArray = [], thirdArray = [];

// an array is a specialised type of object , so this is pass by reference
function fillArray(data) {
  for (var i=0; i<NUM_ITEMS; i++) {
    var number = Math.random()*MAX_VALUE;
    data.push(number);
  }
}

fillArray(firstArray); // Fill array with data
```



8. Arrays and functions

```
const NUM_ITEMS = 10;
var firstArray = [], secondArray = [], thirdArray = [];

function fillArray(data) {
  for (var i=0; i<NUM_ITEMS; i++) {
    var number = Math.random()*MAX_VALUE;
    data.push(number);
  }
}

function displayArray(data) {
  for (var i=0; i<data.length; i++) {
    document.write(data[i].toFixed(2) + " ");
  }
}

fillArray(firstArray);
document.write("first Array: ");
displayArray(firstArray);     // Display array contents
document.write("<br />");
```

8. Arrays and functions

```
// calculateResults
for (var i=0; i<NUM_ITEMS; i++) {
  thirdArray[i] = firstArray[i] + secondArray[i];
}
```

Calculate results

```
function addArrays(a, b) {
  var c  = [];
  for (var i=0; i<a.length; i++) {  // assumes arrays are same length
    c[i] = a[i] + b[i];
  }
  return c;
}

// calculateResults
thirdArray = addArrays(firstArray, secondArray);
document.write("third Array: ");
displayArray(thirdArray);
document.write("<br />");
```

example

9. Returning multiple values from a function

- Use an array to return multiple values from a function

```
function getTwoRandomNumbers(n) {
  return [Math.random()*n, Math.random()*n];
}

var a = getTwoRandomNumbers(10);
console.log(a[0], a[1]);
```

- *Alternative*: return a new object from the function – see a later lecture

```
function getTwoRandomNumbers(n) {
  return {x:Math.random()*n, y:Math.random()*n};
}

var a = getTwoRandomNumbers(10);
console.log(a.x, a.y);
```

example

10. Array methods

- The sort() method will sort the array elements alphabetically by default

```
function fillArray(d) {
  for (var i=0; i<10; i++) {
    var number = Math.floor(Math.random()*50);
    d.push(number)
  }
}

function displayArray(d) {
  document.write(d);
  document.write("<br \/>");
}

var data = [];
fillArray(data);
document.write("unsorted: ");
displayArray(data);
data.sort();
document.write("sorted?: ");
displayArray(data);
```

example

11. Array methods

- Instead, a function may be passed as a parameter which will be used as the comparison operator whilst sorting

```
function cmp(x,y) {
  return x < y ? -1 : x == y ? 0 : 1;
}

function fillArray(d) {
  for (var i=0; i<10; i++) {
    var number = Math.floor(Math.random()*50);
    d.push(number)
  }
}

function displayArray(d) {
  document.write(d);
  document.write("<br \/>");
}

var data = [];
fillArray(data);
document.write("unsorted: ");
displayArray(data);
data.sort(cmp);
document.write("sorted: ");
displayArray(data);
```

Syntax of sort method:
arr.sort([*compareFunction*])

example

12. Multidimensional arrays in JavaScript

- Can be declared in multiple ways

```
var numbers = new Array(new Array(1,2,3),
                        new Array(4,5,6),
                        new Array(7,8,9));
var questions =
  [ ['How many moons does Earth have?', 1],
    ['How many moons does Mars have?', 2]],
    ['How many moons does Saturn have?', 61] ];

\\ alternative layout:
var questions2 = [
  ['How many moons does Earth have?', 1],
  ['How many moons does Mars have?', 2]],
  ['How many moons does Saturn have?', 61]
];
```

12.1 Example

- Write a program that calculates the minimum number of coins required to make up a required amount of money given in pence, e.g. 457 pence is two £2 coins, one 50p coin, one 5p coin and one 2p coin. Only output coins that are part of the change.
- Solution

```
var pence=prompt('Amount in pence? ');
document.write('<p>Amount in pence: '+pence+'</p>');
var coins = Math.floor(pence/200);
if (coins!=0)
  document.write('<p>&pound;2 coins: '+coins+'</p>');
pence = pence%200;
coins = Math.floor(pence/100);
if (coins!=0)
  document.write('<p>&pound;1 coins: '+coins+'</p>');
```

```
pence = pence%100;
coins = Math.floor(pence/50);
if (coins!=0)
  document.write('<p>50p coins: '+coins+'</p>');
pence = pence%50;
coins = Math.floor(pence/20);
if (coins!=0)
  document.write('<p>20p coins: '+coins+'</p>');
pence = pence%20;
coins = Math.floor(pence/10);
if (coins!=0)
  document.write('<p>10p coins: '+coins+'</p>');
pence = pence%10;
coins = Math.floor(pence/5);
if (coins!=0)
  document.write('<p>5p coins: '+coins+'</p>');
pence = pence%5;
coins = Math.floor(pence/2);
if (coins!=0)
  document.write('<p>2p coins: '+coins+'</p>');
pence = pence%2;
coins = Math.floor(pence);
if (coins!=0)
  document.write('<p>1p coins: '+coins+'</p>');
```

12.1 Example

- Solution

```
var pence = prompt('Amount in pence? ');
document.write('<p>Amount in pence: '+pence+'</p>');

var coins = [['&pound;2',200],['&pound;1',100],
             ['50p',50],['20p',20],['10p',10],
             ['5p',5],['2p',2],['1p',1]];

for (var i=0; i<coins.length; i++) {
  var numCoins = Math.floor(pence/coins[i][1]);
  if (numCoins!=0){
    document.write('<p>'+coins[i][0]
                   +' coins: '+numCoins+'</p>');
  }
  pence = pence%coins[i][1];
}
```

13. Summary of JavaScript arrays

- A structure that can be used to collect together a group of items
- Array items can be all the same type or can be a mix of different types
- Built-in Array object contains lots of useful methods to manipulate arrays and their items
- Arrays can grow and shrink dynamically during program execution
- Using arrays is one technique to return multiple values from functions
- Arrays are passed *by reference* to functions
- Multidimensional arrays give even more possibilities to group related information, as well as opening up possibilities for representing such things as two-dimensional spaces such as game boards

Appendix A

- Using the DOM manipulation methods

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>JavaScript examples</title>
</head>

<body>
  <h1>Array example</h1>
  <p id="answer">Answer here</p>
  <script src="./js/getTwo.js"></script>
</body>

</html>
```

Appendix A

```
function aGetTwoRandomNumbers(n) {
  return [Math.random()*n, Math.random()*n];
}

var a = aGetTwoRandomNumbers(10);
console.log(a[0], a[1]);

var element = document.getElementById('answer');

var message = "<p>a= " + a + "</p>";
element.innerHTML = message;

a = aGetTwoRandomNumbers(10);  // get two more random numbers
console.log(a[0], a[1]);

var message = "<p>a= " + a + "</p>";
element.innerHTML += message;  // add more output to the element
```

Note: when writing an array, JavaScript automatically writes each element in a comma-separated list