Arrays

Chapter 2
Objectives

At the end of this session you will be able to:

• Work with arrays
Arrays

• Collection of data represented by a single variable name

• Used for groups of related information
  – e.g. names, courses, departments, etc
Declaring and Initialising Arrays

• Declaring works by calling constructor of function `Array()` with number of elements as the argument

    var arrayName = new Array(number);
    e.g.:
    var studentNames = new Array(34);

• Elements of array are indexed and start with zero
Array Elements

Use **index** between square brackets to identify specific elements:

```javascript
studentNames[0] = "Smith, John";  // 1st element
studentNames[1] = "Doe, Jane";    // 2nd element
studentNames[2] = "Jones, Sarah"; // 3rd element
```

Accessing elements:

```javascript
document.write("<p>"+studentNames[0]+"</p>");
```
Array Elements

• In JavaScript you can use different **type** of elements in the same array

```javascript
var hotelReservation = new Array(4);
hotelReservation[0] = "Don Gosselin"; // name
hotelReservation[1] = 2; // number of nights
hotelReservation[2] = 89.95; // price
hotelReservation[3] = true; // boolean
```
Number of Elements

• Declaring number of elements is optional
  ```javascript
  var studentNames = new Array();
  ```

• `studentNames[0]` and `studentNames[1]` automatically created (values will be `undefined`)

• Use `length` property to determine number of elements:
  ```javascript
  var numStudents = studentNames.length 
  ```
Declaring and Initialising

• Condensed form:
  var cars = new Array("car1","car2","car3");

• Literal form:
  var cars = ["car1","car2","car3"];
Manipulating Arrays

• Use various *methods* to manipulate arrays
  – `indexOf()` – return position of an element
  – `sort()` – sort the elements
  – `shift()` – remove first element
  – `push(element)` – add an element to the end
  – `splice(index, howmany, element1, ...elementX)` – remove and insert
  – `toString()` ... etc
References

var x = new Array();

x contains *reference* or pointer to the array, not the array itself (unlike in the case of primitive types and strings)

Therefore:

var y=x;

y[1] = "Hello" // changes x as well!
Comparing Arrays

- Since the variables contain references, and not pointers, comparing arrays (and any other object instance) is not trivial

\[ x == y \text{ or } x === y \]

Only true if points to the same object
Exercise

```javascript
var x = ["Mary","John","Amy"];
x.sort();
x.indexOf("Mary");
x.splice(1,2,"George");
x.push("Alex");
```