Array

1) Array Literals
2) Length
3) Delete
4) Enumeration
5) Confusion
6) Methods
7) Dimensions
8) Array methods

1. Array Literals

```
var empty = [];
var numbers = [
    'zero', 'one', 'two', 'three', 'four',
    'five', 'six', 'seven', 'eight', 'nine'
];
empty[1]; // undefined
numbers[1]; // 'one'
empty.length; // 0
numbers.length; // 10
```

The object literal:

```
var numbers_object = {
    '0': 'zero',
    '1': 'one',
    '2': 'two',
    '3': 'three',
    '4': 'four',
    '5': 'five',
    '6': 'six',
    '7': 'seven',
    '8': 'eight',
    '9': 'nine'
};

numbers_object['6'];

// In most languages, the elements of an array are all required to be of the same type.
// JavaScript allows an array to contain any mixture of values:

var misc = [
    'string', 98.6, true, false, null, undefined,
    ['nested', 'array'], {object: true}, NaN,
    Infinity
];
```

```
misc.length; // 10
```

2. Length

```
var myArray = [];
myArray.length; // 0
myArray[1000000] = true;
myArray.length; // 1000001
// myArray contains one property.
```

3. Delete

```
var numbers = [
    'zero', 'one', 'two', 'three', 'four',
    'five', 'six', 'seven', 'eight', 'nine'
];

delete numbers[2];
// numbers is ['zero', 'one', undefined, 'shi', 'go']

// Unfortunately, that leaves a hole in the array.

numbers.splice(2, 1);
// numbers is ['zero', 'one', 'shi', 'go']
```

4. Enumeration

```
// Since JavaScript arrays are really objects, the for in statement can be used to iterate over
```

all of the properties of an array.

```
var i;

for (i = 0; i < myArray.length; i += 1) {

    document.writeln(myArray[i]);

}
```

5. Confusion

A common error in JavaScript programs is to use an object when an array is required or an array when an object is required. The rule is simple: when the property names are small sequential integers, you should use an array. Otherwise, use an object.

6. Methods

JavaScript provides a set of methods for acting on arrays. The methods are functions stored in Array.prototype.

// For example, suppose we want to add an array method that will allow us to do computation on an array:

```
Function.prototype.method = function(name, func) {

   this.prototype[name] = func;

   return this;

};

Array.method('reduce', function(f, value) {

    var i;

    for (i = 0; i < this.length; i += 1) {

        value = f(this[i], value);

    }
```

```
    return value;
});

// Create an array of numbers.
var data = [4, 8, 15, 16, 23, 42];
// Define two simple functions. One will add two
// numbers. The other will multiply two numbers.
var add = function(a, b) {
    return a + b;
};
var mult = function(a, b) {
    return a * b;
};

// Invoke the data's reduce method, passing in the
// add function.
var sum = data.reduce(add, 0); // sum is 108
// Invoke the reduce method again, this time passing
// in the multiply function.
var product = data.reduce(mult, 1);
// product is 7418880
// Give the data array a total function.
data.total = function( ) {
    return this.reduce(add, 0);
```

```
};

total = data.total( ); // total is 108
```

7. Dimensions

```
Array.dim = function(dimension, initial) {
    var a = [], i;
    for (i = 0; i < dimension; i += 1) {
        a[i] = initial;
    }
    return a;
};
// Make an array containing 10 zeros.
var myArray = Array.dim(10, 0);
console.log(myArray);


// JavaScript does not have arrays of more than one dimension, but like most C languages, it can have arrays of arrays:
var matrix = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8]
];
document.write(matrix[2][1]); // 7
```

```
//
Array.matrix = function(m, n, initial) {
    var a, i, j, mat = [];
    for (i = 0; i < m; i += 1) {
        a = [];
        for (j = 0; j < n; j += 1) {
            a[j] = initial;
        }
        mat[i] = a;
    }
    return mat;
};
// Make a 4 * 4 matrix filled with zeros.
var myMatrix = Array.matrix(4, 4, 0);
console.log(myMatrix);

// Method to make an identity matrix.

Array.identity = function(n) {
    var i, mat = Array.matrix(n, n, 0);
    for (i = 0; i < n; i += 1) {
        mat[i][i] = 1;
    }
```

```
    return mat;
};


myMatrix = Array.identity(4);
document.writeln(myMatrix[3][3]); // 1
```

8. Important Array methods

```
Array.prototype.concat;
Array.prototype.join;
Array.prototype.pop();
Array.prototype.push();
Array.prototype.slice();
Array.prototype.sort();
Array.prototype.splice();
Array.prototype.unshift();
```