Arrays

An array is a java datatype (like *number* or *boolean*).

- Arrays represent *collections* of values. Each array contains one or more distinct values.
  - Each *element* (individual value) of the array is accessed using an *index*. The indices of the array start at zero (just like strings).

To create an array you should use the *array constructor*:

```javascript
var words = new Array(5);
```

(this creates a new array that contains five elements and assigns it to the variable “words”. The number in the brackets indicates the array size, but in javascript it is usual to leave this out, which creates an array with no elements. Since Javascript allows us to add new elements any time this is easiest).
Arrays

An array is a java datatype (like *number* or *boolean*).

- Arrays represent *collections* of values. Each array contains one or more distinct values.

- Each *element* (individual value) of the array is accessed using an *index*. The indices of the array start at zero (just like strings).

To create an array you should use the *array constructor*:

```javascript
var words = new Array(5);
```

(this creates a new array that contains five elements and assigns it to the variable “words”. The number in the brackets indicates the array size, but in javascript it is usual to leave this out, which creates an array with no elements. Since Javascript allows us to add new elements any time this is easiest).
Accessing and Updating array elements

Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] );
var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
Accessing and Updating array elements

Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] ); var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] ); var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
Accessing and Updating array elements

Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] ); var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
Accessing and Updating array elements

Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] ); var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
Accessing and Updating array elements

Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] ); var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
Accessing and Updating array elements

Each element of the array is a variable; we find them using *indices*, which are placed after the variable name thus:

```javascript
words[0] = "Darwin";
words[1] = "Huxley";
words[2] = "Haeckel";
words[3] = "Mencken";
```

Each number is an index, each element of "words" is a variable. Array elements can also be accessed using the index:

```javascript
document.write( words[1] ); var s = words[0+2];
```

As an alternative we can assign values when the array is created:

```javascript
var words = new Array("Darwin", "Huxley", "Haeckel", "Mencken");
```
We must take care that the index has had a value assigned to it – otherwise the value will be “undefined”:

```javascript
document.write( words[123] ); // Undefined at this point!
```

Like a string an array has a special property, called `length` which can be used to find out how many elements it contains:

```javascript
var size = words.length;
document.write("I know "+size+" words!");
```
Arrays are great when we want to store lots of values which are related in some way. Instead of creating several variables (called “word1”, “word2” and so on) we create an array. This allows us to compute which variable we want.

Here is a function that could be used to replace the get_secret function in the hangman program.

function get_secret(){
  var words = new Array("fish", "banana", "horsefly", "oolong", "clouded", "monkey", "trucker");

  var index = Math.round( Math.random() * words.length );
  return words[index];
}
For loops

We have seen one kind of loop structure in Javascript – the while loop. The for loop is another.

for loops go great with arrays! A for loop is best used when you want to repeat a block of code a certain number of times using a counter. As well as a block of code each for loop contains:

- An initialisation section that sets the loop counter to it’s first value
- A condition that will be checked each time around the loop
- A statement that will be executed each time (usually used to increment or decrement the loop counter).

```
for( initialisation , test , statement ){
    block
}
```
Examples

// Countdown!
for( i=10; i>0; i-- ){
    alert(i);
}

// Print the even numbers from 2 to 100
for( i=2; i<=100; i=i+2 ){
    document.write(i);
}

// Print out all the words in an array
for( i=0; i<words.length; i++){
    document.write( words[i] );
}
What the ...! i++ ? i--

We see in the previous examples use of the following shorthands:

- $i--$ is short for $i = i - 1$
- $i++$ is short for $i = i + 1$

*Note:* this does not extend to multiplication or division.

The following shorthands work for all four arithmetic operations:

- $n += e$ is short for $n = n + e$
- $n -= e$ is short for $n = n - e$
- $n *= e$ is short for $n = n * e$
- $n /= e$ is short for $n = n / e$