# 23. JavaScript Array Object

The Array object is used to store multiple values in a single variable.

## Example

Create an array, and assign values to it:

```
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
```

## 23.1. What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1="Saab";
var car2="Volvo";
var car3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

## 23.2. Create an Array

An array can be created in three ways.

The following code creates an Array object called myCars:

1: Regular:

```
var myCars=new Array();
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

2: Condensed:

```
var myCars=new Array("Saab","Volvo","BMW");
```

3: Literal:

```
var myCars=["Saab","Volvo","BMW"];
```

## 23.3. Access an Array

You refer to an element in an array by referring to the **index** number.

This statement access the value of the first element in myCars:

```
var name=myCars[0];
```

This statement modifies the first element in myCars:

```
myCars[0]="Opel";
```

[0] is the first element in an array. [1] is the second . . . . . (indexes start with 0)

## 23.4. You Can Have Different Objects in One Array

All JavaScript variables are objects. Array elements are objects. Functions are objects.

Because of this, you can have variables of different types in the same Array.

You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

```
myArray[0]=Date.now;
myArray[1]=myFunction;
myArray[2]=myCars;
```

## 23.5. Array Methods and Properties

The Array object has predefined properties and methods:

```
var x=myCars.length              // the number of
elements in myCars
var y=myCars.indexOf("Volvo")    // the index
```

```
    position of "Volvo"
```

## 23.6. Create New Methods

Prototype is a global constructor in JavaScript. It can construct new properties and methods for any JavaScript Objects.

## Example: Make a new Array method.

```
    Array.prototype.ucase=function()
    {
      for (i=0;i<this.length;i++)
      {this[i]=this[i].toUpperCase();}
    }
```

The example above makes a new array method that transforms array values into upper case.

## 23.7. More Examples

**a) Join two arrays – concat()**

```
    <!DOCTYPE html>
    <html>
    <body>

    <p id="demo">Click the button to join three
    arrays.</p>

    <button onclick="myFunction()">Try it</button>

    <script>
    function myFunction()
    {
    var hege = ["Cecilie", "Lone"];
    var stale = ["Emil", "Tobias", "Linus"];
    var kai = ["Robin"];
    var children = hege.concat(stale,kai);
    var x=document.getElementById("demo");
    x.innerHTML=children;
    }
```

```
</script>

</body>
</html>
```

**b) Join three arrays – concat()**

```
<!DOCTYPE html>
<html>
<body>

<script>

var parents = ["Jani", "Tove"];
var brothers = ["Stale", "Kai Jim", "Borge"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(brothers, children);
document.write(family);

</script>

</body>
</html>
```

**c) Join all elements of an array into a string – join()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to join the array
elements into a string.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var x=document.getElementById("demo");
x.innerHTML=fruits.join();
}
```

```
</script>

</body>
</html>
```

**e) Add new elements to the end of an array – push()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to add a new element
to the array.</p>

<button onclick="myFunction()">Try it</button>

<script>
var fruits = ["Banana", "Orange", "Apple",
"Mango"];

function myFunction()
{
fruits.push("Kiwi")
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

</body>
</html>
```

**f) Reverse the order of the elements in an array – reverse()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to reverse the order
of the elements in the array.</p>

<button onclick="myFunction()">Try it</button>

<script>
```

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];

function myFunction()
{
fruits.reverse();
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

</body>
</html>
```

**g) Remove the first element of an array – shift()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to remove the first
element of the array.</p>

<button onclick="myFunction()">Try it</button>

<script>
var fruits = ["Banana", "Orange", "Apple",
"Mango"];

function myFunction()
{
fruits.shift();
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

</body>
</html>
```

**h) Select elements from an array – slice()**

```
<!DOCTYPE html>
<html>
```

```
<body>

<p id="demo">Click the button to extract the second
and the third elements from the array.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var fruits = ["Banana", "Orange", "Lemon", "Apple",
"Mango"];
var citrus = fruits.slice(1,3);
var x=document.getElementById("demo");
x.innerHTML=citrus;
}
</script>

</body>
</html>
```

**i) Sort an array (alphabetically and ascending) – sort()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to sort the
array.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.sort();
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

</body>
</html>
```

### j) Sort numbers (numerically and ascending) – sort()

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to sort the
array.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var points = [40,100,1,5,25,10];
points.sort(function(a,b){return a-b});
var x=document.getElementById("demo");
x.innerHTML=points;
}
</script>

</body>
</html>
```

### k) Sort numbers (numerically and descending) – sort()

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to sort the
array.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var points = [40,100,1,5,25,10];
points.sort(function(a,b){return b-a});
var x=document.getElementById("demo");
x.innerHTML=points;
}
</script>
```

```
</body>
</html>
```

**l) Add an element to position 2 in an array – splice()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to add elements to
the array.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.splice(2,0,"Lemon","Kiwi");
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

</body>
</html>
```

**m) Convert an array to a string – toString()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to convert the array
into a String.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
```

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.toString();
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

</body>
</html>
```

**n) Add new elements to the beginning of an array – unshift()**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to add elements to
the array.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.unshift("Lemon","Pineapple");
var x=document.getElementById("demo");
x.innerHTML=fruits;
}
</script>

<p><b>Note:</b> The unshift() method does not work
properly in Internet Explorer 8 and earlier, the
values will be inserted, but the return value will
be <em>undefined</em>.</p>

</body>
</html>
```

# 23.8. Complete Array Object Reference

For a complete reference of all properties and methods, go to our complete Array object reference.

The reference contains a description of all Array properties and methods.

## Array Object Properties

| Property | Description |
|---|---|
| constructor | Returns the function that created the Array object's prototype |
| length | Sets or returns the number of elements in an array |
| prototype | Allows you to add properties and methods to an Array object |

## Array Object Methods

| Method | Description |
|---|---|
| concat() | Joins two or more arrays, and returns a copy of the joined arrays |
| indexOf() | Search the array for an element and returns its position |
| join() | Joins all elements of an array into a string |
| lastIndexOf() | Search the array for an element, starting at the end, and returns its position |
| pop() | Removes the last element of an array, and returns that element |
| push() | Adds new elements to the end of an array, and returns the new length |
| reverse() | Reverses the order of the elements in an array |
| shift() | Removes the first element of an array, and returns that element |
| slice() | Selects a part of an array, and returns the new array |
| sort() | Sorts the elements of an array |
| splice() | Adds/Removes elements from an array |
| toString() | Converts an array to a string, and returns the result |
| unshift() | Adds new elements to the beginning of an array, and returns the new length |
| valueOf() | Returns the primitive value of an array |