Array

The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Syntax-

Use the following syntax to create an **Array** object −

```
var fruits = new Array( "apple", "orange", "mango" );
```

The **Array** parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

You can create array by simply assigning values as follows −

```
var fruits = [ "apple", "orange", "mango" ];
```

You will use ordinal numbers to access and to set values inside an array as follows.

```
fruits[0] is the first element
fruits[1] is the second element
fruits[2] is the third element
```

Array Properties-

Here is a list of the properties of the Array object along with their description.

1. **Constructor**- Returns a reference to the array function that created the object.
2. **Index-** The property represents the zero-based index of the match in the string
3. **Input-** This property is only present in arrays created by regular expression matches.
4. **Length**- Reflects the number of elements in an array.
5. **Prototype**- The prototype property allows you to add properties and methods to an object.

Array Methods-

Here is a list of the methods of the Array object along with their description.

1. **concat()**- Returns a new array comprised of this array joined with other array(s) and/or value(s).
2. **every()**- Returns true if every element in this array satisfies the provided testing function.

3. **filter()**- Creates a new array with all of the elements of this array for which the provided filtering function returns true.
4. **forEach()**- Calls a function for each element in the array.
5. **indexOf()**- Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
6. **join()**- Joins all elements of an array into a string.
7. **lastIndexOf()**- Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
8. **map()**- Creates a new array with the results of calling a provided function on every element in this array.
9. **pop()**- Removes the last element from an array and returns that element.
10. **push()**- Adds one or more elements to the end of an array and returns the new length of the array.
11. **reduce()**- Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
12. **reduceRight()**- Apply a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value.
13. **reverse()**- Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
14. **shift()**- Removes the first element from an array and returns that element.
15. **slice()**- Extracts a section of an array and returns a new array.
16. **some()**- Returns true if at least one element in this array satisfies the provided testing function.
17. **toSource()**- Represents the source code of an object
18. **sort()**- Sorts the elements of an array
19. **splice()**- Adds and/or removes elements from an array.
20. **toString()**- Returns a string representing the array and its elements.
21. **unshift()**- Adds one or more elements to the front of an array and returns the new length of the array.