Looping through an array

The array property ***length*** contains the number of elements in the array. The length property is helpful for looping through an array using a for loop.

Figure 6.8.1: Looping through an array with a for loop.

```
let groceries = ["bread", "milk", "peanut butter"];

// Display all elements in groceries array
for (i = 0; i < groceries.length; i++) {
  console.log(i + " - " + groceries[i]);
}
```

```
0 - bread
1 - milk
2 - peanut butter
```

The ***for-of loop*** is a simplified for loop that loops through an entire array. The array name is placed after the of keyword in a for-of loop. Each time through the loop, the next array element is assigned to the variable in front of the of keyword.

Looping through an array with a for-of loop.

```
let groceries = ["bread", "milk", "peanut butter"];

// Display all elements in groceries array
for (let item of groceries) {
  console.log(item);
}
```

```
bread
milk
peanut butter
```

The Array method ***forEach()*** also loops through an array. The forEach() method takes a function as an argument. The function is called for each array element in order, passing the element and the element index to the function.

Figure 6.8.3: Looping through an array with the forEach() method.

```
let groceries = ["bread", "milk", "peanut butter"];

// Display all elements in groceries array
groceries.forEach(function(item, index) {
```

```
  console.log(index + " - " + item);
});
```

```
0 - bread
1 - milk
2 - peanut butter
```

Searching an array

The array methods ***indexOf()*** and ***lastIndexOf()*** search an array and return the index of the first found value or -1 if the value is not found. indexOf() searches from the beginning of the array to the end. lastIndexOf() searches from the end of the array to the beginning. Both functions take two arguments:

1. searchValue - The value to search for
2. startingPosition - Optional argument that indicates the index at which the search should begin (default is 0
3.

Searching for array elements.

```
let scores = [80, 92, 75, 64, 88, 92];

s = scores.indexOf(92);        // 1
s = scores.indexOf(92, 2);     // 5
s = scores.indexOf(100);       // -1
s = scores.lastIndexOf(92);    // 5
s = scores.lastIndexOf(92, 4); // 1
s = scores.lastIndexOf(50);    // -1
```

Sorting an array

The array method ***sort()*** sorts an array in ascending (increasing) order. sort()'s default behavior is to sort each element as a string using the string's Unicode values. Sorting by Unicode values may yield unsatisfactory results for arrays that store numbers. Ex: 10 is sorted before 2 because "10" is < "2" when comparing the Unicode values of "1" to "2".

The sort() method can sort elements in other ways by passing a comparison function to sort(). The comparison function returns a number that helps sort() determine the sorting order of the array's elements:

- Returns a value < 0 if the first argument should appear before the second argument.
- Returns a value > 0 if the first argument should appear after the second argument.

- Returns 0 if the order of the first and second arguments does not matter.

Sorting an array of numbers.

```
let numbers = [200, 30, 1000, 4];

// Sort based on Unicode values: [1000, 200, 30, 4]
numbers.sort();

// Sort numbers in ascending order: [4, 30, 200, 1000]
numbers.sort(function(a, b) {
  return a - b;
});
```

Example : Operations on Arrays

```
let book=["math","physics","bio","computer science"];
console.log("length: ",book.length);
//to add element
book.push("Hindi")
console.log("After adding in end: ",book);
//to add in the beginning
book.unshift("english");
console.log("After adding in beginning: ",book);
//deleting at end
book.pop()
console.log("After deleting from end: ",book)
//deleting from beginning
book.shift()
console.log("After deleting from beginning: ",book)
//deleting from index.splice [1,2] 1 ->start index ,  2 - number of elements
book.splice(1,2);
console.log("after splicing: ",book)
//To empty array :
book=[];
console.log("After emptying the array " , book)
//Another way to empty
book.length=0;
console.log("After emptying the array " , book)
```

Output :

```
length:  4
After adding in end:  ▶ (5) ['math', 'physics', 'bio', 'computer science', 'Hindi']
After adding in beginning:
▶ (6) ['english', 'math', 'physics', 'bio', 'computer science', 'Hindi']
After deleting from end:  ▶ (5) ['english', 'math', 'physics', 'bio', 'computer science']
After deleting from beginning:  ▶ (4) ['math', 'physics', 'bio', 'computer science']
after splicing:  ▶ (2) ['math', 'computer science']
After emptying the array  ▶ []
After emptying the array  ▶ []
```

```
//To know the position of any element
position=book.indexOf("bio");
console.log("position: ",position);
let book1="Math"
//to find whether it is array
console.log("Is book an array : ",Array.isArray(book));
let text = "this is a random text ";
//To store it in different element, use split.
//Argument is a space in this case. if you want to split by comma,use comma
let wordArray=text.split(' ')
console.log("After splitting ",wordArray)
//splitting at s
let wr=text.split('s')
console.log("After splitting ",wr)
//To display as text. use join
let textjoin=book.join(' ');
console.log("After joining: ",textjoin)
//can use other seperator too.
let textjoinOther=book.join('_');
console.log("After joining: ",textjoinOther)
```

Output :

```
▶ (4) ['math', 'physics', 'bio', 'computer science']
length:  4
position:  2
Is book an array :  true
After splitting  ▶ (6) ['this', 'is', 'a', 'random', 'text', '']
After splitting  ▶ (3) ['thi', ' i', ' a random text ']
After joining:  math physics bio computer science
After joining:  math_physics_bio_computer science
```