Arrays in JS:

In JavaScript, an array is a type of object that can store a collection of values. These values can be of any type, including numbers, strings, and other objects. Arrays are created using the [] notation, and values are added to an array using the push() method or by specifying the index of the value you want to add. Arrays can also be accessed and modified using the square bracket notation, and various built-in methods like pop(), shift(), unshift(), slice(), splice(), sort(), reverse(), concat(), join(), map(), filter(), reduce(), and forEach() are available for manipulating the elements of an array.

Here is an example of creating and manipulating an array in JavaScript:

```javascript
// Creating an array
let myArray = [1, 2, 3, 4, 5];

// Adding a value to the end of the array
myArray.push(6); // myArray is now [1, 2, 3, 4, 5, 6]

// Removing the last value from the array
myArray.pop(); // myArray is now [1, 2, 3, 4, 5]

// Adding a value to the beginning of the array
myArray.unshift(0); // myArray is now [0, 1, 2, 3, 4, 5]

// Removing the first value from the array
myArray.shift(); // myArray is now [1, 2, 3, 4, 5]

// Sorting the array
myArray.sort(); // myArray is now [1, 2, 3, 4, 5]

// Reversing the array
myArray.reverse(); // myArray is now [5, 4, 3, 2, 1]
```

You can also use the forEach() method to iterate over the elements of an array:

```javascript
// Iterating over the elements of an array
myArray.forEach(function(element) {
  console.log(element);
});
```

This will output each element of the array to the console.

You can also use the map(), filter() and reduce() methods for functional programming.

```javascript
// Using the map() method
let newArray = myArray.map(function(element) {
  return element * 2;
});

// Using the filter() method
let filteredArray = myArray.filter(function(element) {
  return element > 3;
});

// Using the reduce() method
let total = myArray.reduce(function(accumulator, currentValue) {
  return accumulator + currentValue;
});
```

Note: Map, filter and reduce methods will return a new array, the original array remains unchanged.

Arrays - sort() Method:

The sort() method in JavaScript is used to sort the elements of an array in ascending order. By default, the sort() method sorts the elements of an array as strings.

Here's an example of using the sort() method on an array of numbers:

```javascript
let myArray = [5, 2, 9, 1, 7];
myArray.sort();
console.log(myArray); // Output: [1, 2, 5, 7, 9]
```

The sort() method can also accept a comparison function as an argument, which is used to determine the order of the elements. The comparison function should return a negative, zero, or positive value, depending on the arguments, like:

- a negative value if a should be sorted lower than b
- a positive value if a should be sorted higher than b,
- 0 if a and b are equal.

Here's an example of using a comparison function to sort an array of objects:

```javascript
let myArray = [{name: 'John', age: 25}, {name: 'Mary', age: 30},
{name: 'Bob', age: 20}];
myArray.sort(function(a, b) {
  if (a.age < b.age) {
    return -1;
  }
  if (a.age > b.age) {
    return 1;
  }
  return 0;
});
console.log(myArray);
/* Output:
[{name: 'Bob', age: 20},
 {name: 'John', age: 25},
 {name: 'Mary', age: 30}]
*/
```

In this example, the comparison function compares the age property of each object and sorts the array based on that.

Note: sort() method modifies the original array, it does not create a new array.

Arrays - Push and Pop Methods:

push() and pop() are built-in methods in JavaScript that are used to add and remove elements from the end of an array, respectively.

The push() method is used to add one or more elements to the end of an array. It takes one or more arguments, and it returns the new length of the array. Here's an example:

```javascript
let myArray = [1, 2, 3];
let newLength = myArray.push(4, 5);
console.log(myArray); // Output: [1, 2, 3, 4, 5]
console.log(newLength); // Output: 5
```

The pop() method is used to remove the last element from an array and return that element. Here's an example:

```javascript
let myArray = [1, 2, 3];
let lastElement = myArray.pop();
console.log(myArray); // Output: [1, 2]
console.log(lastElement); // Output: 3
```

Both push() and pop() methods modify the original array, they do not create a new array.

In short, push() method is used to add an element to the end of an array, and pop() method is used to remove the last element from an array.

Arrays - Shift, Slice and Split Methods:

shift(), slice(), and splice() are built-in methods in JavaScript that are used to manipulate the elements of an array.

The shift() method is used to remove the first element of an array and return that element. Here's an example:

```javascript
let myArray = [1, 2, 3];
let firstElement = myArray.shift();
console.log(myArray); // Output: [2, 3]
console.log(firstElement); // Output: 1
```

The slice() method is used to extract a section of an array and return it as a new array. The slice() method takes two arguments: the starting index and the ending index (not inclusive). Here's an example:

```javascript
let myArray = [1, 2, 3, 4, 5];
let newArray = myArray.slice(1, 3);
console.log(myArray); // Output: [1, 2, 3, 4, 5]
console.log(newArray); // Output: [2, 3]
```

The splice() method is used to add or remove elements from an array. The splice() method takes three arguments: the index to start at, the number of elements to remove, and the elements to add. Here's an example:

```javascript
let myArray = [1, 2, 3, 4, 5];
let removed = myArray.splice(1, 2, 6, 7);
console.log(myArray); // Output: [1, 6, 7, 4, 5]
console.log(removed); // Output: [2, 3]
```

In short, shift() method is used to remove the first element of an array and return it, slice() method is used to extract a section of an array and return it as a new array, and splice() method is used to add or remove elements from an array.

Note: shift(), slice(), and splice() methods modify the original array, they do not create a new array.