ARRAYS IN JAVA SCRIPT

Sai Vardhan T

ARRAYS

- JavaScript arrays are a fundamental data structure used to store multiple values in a single variable.
- They are a type of object used for storing multiple values in a single variable.
- Each value in an array is called an element, and each element can be accessed by its index.
- Arrays in JavaScript can hold elements of any data type, including strings, numbers, objects, and even other arrays.

CREATING ARRAYS

- You can create an array in JavaScript using either literal notation or the Array constructor.
- Literal Notation:

```
let fruits = ['Apple', 'Banana', 'Orange'];
```

- Using "**Array**" Constructor:

```
let fruits = new Array('Apple', 'Banana', 'Orange');
```

ACCESSING ELEMENTS

- We can access elements of an array using their index
- JavaScript arrays are Zero-indexed, meaning the index of the first element is 0.

```
let firstFruit = fruits[0]; // Accessing the first element
let secondFruit = fruits[1]; // Accessing the second element
```

MODIFYING ELEMENTS

- We can modify elements of an array by assigning a new value to the specific index.

```
fruits[0] = 'Mango'; // Modifying the first element
```

ARRAY METHODS

- JavaScript provides a variety of built-in methods for working with arrays
  - Push()
  - Pop()
  - Shift()
  - Unshift()
  - Slice()
  - Splice()
  - Concat()
  - Join()
  - Indexof()
  - forEach()
  - Map()
  - Filter()
  - Reduce()

PUSH()

- Adds one or more elements to the end of an array and returns the new length of the array.

```
let fruits = ['Apple', 'Banana', 'Orange'];
fruits.push('Mango');
console.log(fruits); // Output: ['Apple', 'Banana', 'Orange', 'Mango']
```

POP()

- Removes the last element from an array and returns that element.

```
let fruits = ['Apple', 'Banana', 'Orange'];
let lastFruit = fruits.pop();
console.log(lastFruit); // Output: 'Orange'
console.log(fruits); // Output: ['Apple', 'Banana']
```

SHIFT()

- Removes the first element from an array and returns that element, shifting all subsequent elements to a lower index.

```
let fruits = ['Apple', 'Banana', 'Orange'];
let firstFruit = fruits.shift();
console.log(firstFruit); // Output: 'Apple'
console.log(fruits); // Output: ['Banana', 'Orange']
```

UNSHIFT()

- Adds one or more elements to the beginning of an array and returns the new length of the array.

```
let fruits = ['Apple', 'Banana', 'Orange'];
fruits.unshift('Mango');
console.log(fruits); // Output: ['Mango', 'Apple', 'Banana', 'Orange']
```

SLICE()

- Returns a shallow copy of a portion of an array into a new array object selected from start to end(end not included)

```
let fruits = ['Apple', 'Banana', 'Orange', 'Mango', 'Grapes'];
let selectedFruits = fruits.slice(1, 3);
console.log(selectedFruits); // Output: ['Banana', 'Orange']
```

SPLICE()

- Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

```
let fruits = ['Apple', 'Banana', 'Orange', 'Mango', 'Grapes'];
fruits.splice(2, 1, 'Peach', 'Kiwi');
console.log(fruits); // Output: ['Apple', 'Banana', 'Peach', 'Kiwi', 'Mango', 'Grapes
```

CONCAT()

- Returns a new array comprised of the array joined with other arrays(s) and/or value(s)

```
let fruits1 = ['Apple', 'Banana'];
let fruits2 = ['Orange', 'Mango'];
let allFruits = fruits1.concat(fruits2);
console.log(allFruits); // Output: ['Apple', 'Banana', 'Orange', 'Mango']
```

JOIN()

- Joins all elements of an array into a string.

```
let fruits = ['Apple', 'Banana', 'Orange'];
let fruitString = fruits.join(', ');
console.log(fruitString); // Output: 'Apple, Banana, Orange'
```

indexOf()

- Returns the first index at which a given element can be found in the array or -1 if it is not present

```
let fruits = ['Apple', 'Banana', 'Orange'];
let index = fruits.indexOf('Banana');
console.log(index); // Output: 1
```

forEach()

- Executes a provided function once for each array element

```
let fruits = ['Apple', 'Banana', 'Orange'];
fruits.forEach(function(fruit) {
    console.log(fruit);
});
// Output:
// 'Apple'
// 'Banana'
// 'Orange'
```

Map()

- Creates a new array populated with the results of calling a provided function on every element in the calling array.

```
let numbers = [1, 2, 3];
let doubled = numbers.map(function(num) {
    return num * 2;
});
console.log(doubled); // Output: [2, 4, 6]
```

Filter()

- Creates a new array with all elements that pass the test implemented by the provided function.

```
let numbers = [1, 2, 3, 4, 5];
let evenNumbers = numbers.filter(function(num) {
    return num % 2 === 0;
});
console.log(evenNumbers); // Output: [2, 4]
```

Reduce()

- Applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value.

```
let numbers = [1, 2, 3, 4, 5];
let sum = numbers.reduce(function(accumulator, currentValue) {
    return accumulator + currentValue;
}, 0);
console.log(sum); // Output: 15
```