MASTERING ARRAYS in JS

Jaspreet Kaur

SWIPE

CREATING ARRAYS

Arrays can be created using the array literal notation or the Array() constructor function.

Example:

```
const fruits = ['apple', 'banana', 'orange'];

const numbers = new Array(1, 2, 3, 4, 5);
```

Jaspreet Kaur

SWIPE

ACCESSING ARRAY ELEMENTS

Array elements can be accessed using index numbers, starting from 0.

Example:

```
const fruits = ['apple', 'banana', 'orange'];

console.log(fruits[0]);
// Output: "apple"
console.log(fruits[1]);
// Output: "banana"
```

Jaspreet Kaur

SWIPE

ARRAY METHODS

Arrays come with many built-in methods for performing common operations, such as adding or removing elements, sorting, and searching.

Example:

```
const fruits = ['apple', 'banana', 'orange'];

fruits.push('kiwi');
// adds "kiwi" to the end of the array
console.log(fruits);
// Output: ["apple", "banana", "orange", "kiwi"]
```

Jaspreet Kaur

SWIPE

ITERATING OVER ARRAYS

Arrays can be iterated over using loops or built-in methods like forEach(), map(), and reduce()

Example:

```
const numbers = [1, 2, 3, 4, 5];
for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}
// Output: 1, 2, 3, 4, 5

const doubledNumbers = numbers.map(number
=> number * 2);
console.log(doubledNumbers);
// Output: [2, 4, 6, 8, 10]
```

Jaspreet Kaur

SWIPE

MULTI-DIMENSIONAL ARRAYS

Arrays can contain other arrays, creating multi-dimensional arrays.

Example:

```
const numbers = [1, 2, 3, 4, 5];
for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}
// Output: 1, 2, 3, 4, 5

const doubledNumbers = numbers.map(number
=> number * 2);
console.log(doubledNumbers);
// Output: [2, 4, 6, 8, 10]
```

Jaspreet Kaur

SWIPE

Thank You

Jaspreet Kaur

FOLLOW FOR MORE