script.js

```
// Arrays in JavaScript

// An array is a data structure that allows you to store multiple values in a single variable. It can hold various data types, including numbers, strings, objects and even other arrays. Arrays are zero-indexed, meaning that the first element in an array is at index 0, the second element with index 1 and so on.

// We will cover the following topics:

/*

1. Creating Arrays, Accessing Elements and Modifying Elements

2. Array Traversal, Iterations

3. Updating and Deleting Elements

4. Filter and Search

5. Sort and Compare

5. Important Array Methods

and a lot more things....

*/

// Example: Creating an Array

// Using Array Literal

let arr = [1, 2, 3, 4, 5];

console.log(typeof arr); // object

console.log(arr);

```

```
// Using Array Constructor

let fruitsArr = new Array("apple", "banana", "orange", "grapes", "pineapple");

console.log(fruitsArr);

// Accessing Elements in an Array

console.log(fruitsArr[0]); // apple

console.log(fruitsArr[1]); // banana

console.log(fruitsArr[2]); // orange

// Modifying Elements in an Array

fruitsArr[0] = "mango";

console.log(fruitsArr);

// Array Traversal

// 1. for of loop is used to iterate over the values of an iterable object such as an array or strings. For example:

console.log("Using for of loop");

for (const fruitItem of fruitsArr) {

    console.log(fruitItem);
}

// fruitItem[0] = 'mango'

// fruitItem[1] = 'banana'

// fruitItem[2] = 'orange'

// fruitItem[3] = 'grapes'
```

```
73
74  // fruitItem[4] = 'pineapple'
75
76  console.log("Using for loop");
77
78  for (let item = 0; item < fruitsArr.length; item++) {
79
80      console.log(fruitsArr[item]);
81  }
82
83  // 2. for in loop is used to iterate over the keys of an object. It is used when you want to iterate over the keys of an object.
    It is a bit more efficient than the for of loop.
84
85  console.log("Using for in loop");
86
87  for (const key in fruitsArr) {
88
89      console.log(key);
90  }
91
92  // 3. for each loop calls a function for each element in an array. It is a bit more efficient than the for of loop. Here's a
    breakdown of each part:
93
94  /*
95
96  1. array: The array on which the forEach() method is called.
97
98  2. callback: A function that is called once for each element in the array.
99
100 3. currentValue: The current element being processed in the array.
101
102 4. index: The index of the current element being processed in the array.
103
104 */
105
106 fruitsArr.forEach((fruitItem, index, arr) => {
107
108     console.log(index, fruitItem, arr);
```

```
});

// 4. map() creates a new array with the results of calling a function for every array element. It allows method chaining.

const newArr = fruitsArr.map((fruitItem, index, arr) => {

    return fruitItem + " " + index;
});

console.log(newArr);

// Array Methods

/*

1. push(): Adds one or more elements to the end of an array. It modifies the original array and returns the new length of the array.

2. pop(): Removes the last element from an array. It modifies the original array and returns the removed element.

3. unshift(): Adds one or more elements to the beginning of an array.

4. shift(): Removes the first element from an array.

5. splice(): Removes elements from an array and, if necessary, inserts new elements in their place. It modifies the original array and returns the removed elements.

*/

const numbersArray = [1, 2, 3, 4, 5];

console.log(numbersArray);

numbersArray.push(6); // 1, 2, 3, 4, 5, 6

console.log(numbersArray);

numbersArray.pop(); // 1, 2, 3, 4, 5
```

```
145
146 console.log(numbersArray);
147
148 numbersArray.unshift(0); // 0, 1, 2, 3, 4, 5
149
150 console.log(numbersArray);
151
152 numbersArray.shift(); // 1, 2, 3, 4, 5
153
154 console.log(numbersArray);
155
156 numbersArray.splice(3, 1, 7); // 1 2 3 7 5
157
158 console.log(numbersArray);
159
160 // Searching in an Array
161
162 /*
163
164 1. indexOf(): Returns the index of the first occurrence of a specified value in an array. It returns -1 if the value is not
    found.
165
166 2. lastIndexOf(): Returns the index of the last occurrence of a specified value in an array. It returns -1 if the value is not
    found.
167
168 3. includes(): Returns true if an array contains a specified value. It returns false if the value is not found.
169
170 */
171
172 const searchArray = [1, 2, 3, 6, 4, 5, 6, 7, 8, 9];
173
174 console.log(searchArray);
175
176 console.log("The index of 4 is", searchArray.indexOf(5)); // 4
177
178 console.log("The last index of 6 is", searchArray.lastIndexOf(6, 5)); // 3
179
180 console.log("Array contains 2 ?", searchArray.includes(2)); // true
```

```
181
182 // Filtering an Array
183
184 console.log("Filtering an Array");
185
186 const filterArray = [1, 2, 3, 4, 5, 6, 7, 8, 6, 9];
187
188 console.log(filterArray);
189
190 /*
191
192 1. find(): Returns the value of the first element in the array that satisfies the provided testing function.
193
194 2. findIndex(): Returns the index of the first element in the array that satisfies the provided testing function.
195
196 3. filter(): Returns a new array with all elements that pass the test implemented by the provided function.
197
198 4. sort(): Sorts the elements of an array in place and returns the sorted array. It takes a comparator function as an argument.
199
200 */
201
202 const findElement = filterArray.find((currentElement) => {
203
204     return currentElement > 5;
205 });
206
207 console.log(findElement); // 6
208
209 const findIndexElement = filterArray.findIndex((currentElement) => {
210
211     return currentElement > 5;
212 });
213
214 console.log(findIndexElement); // 6
215
216 let value = 6;
217
218 const newArray = filterArray.filter((currentElement) => {
```

```
219
220     return currentElement !== value;
221 });
222
223 console.log(newArray);
224
225 // Sorting an Array: It takes 2 comparator functions as arguments and sorts the elements of an array in place. It returns the
    sorted array.
226
227 console.log("Sorting an Array");
228
229 const jumbledArray = [1, 2, 3, 4, 5, 6, 7, 8, 6, 9];
230
231 jumbledArray.sort((a, b) => {
232
233     return a - b;
234 });
235
236 console.log(jumbledArray);
237
238 // Reduce Method: It takes 2 arguments: accumulator and currentElement. Initial value of accumulator is 0 and after each
    iteration, the accumulator value is updated. The final value of accumulator is returned. It returns a new array.
239
240 console.log("Reduce Method");
241
242 const reduceArray = [1, 2, 3, 4, 5, 6, 7, 8, 6, 9];
243
244 const sum = reduceArray.reduce((accumulator, currentElement) => {
245
246     return accumulator + currentElement;
247
248 }, 0);
249
250 console.log(sum);
```