

```
1 // ✅ JavaScript Arrays - Mastering the Fundamentals
2
3 // An array is a data structure that allows you to store multiple values in a single variable.
4
5 // It can hold various data types, including numbers, strings, objects and even other arrays.
6
7 // 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.
8
9 // 📦 We will cover the following topics:
10
11 /*
12
13 1. Creating Arrays, Accessing Elements and Modifying Elements
14
15 2. Array Traversal, Iterations
16
17 3. Updating and Deleting Elements
18
19 4. Filter and Search
20
21 5. Sort and Compare
22
23 6. Important Array Methods
24
25 ...and a lot more things.
26
27 */
28
29 console.clear();
30
31 // 💎 Example: Creating an Array
32
33 // ✅ Using Array Literal
34
35 let arr = [1, 2, 3, 4, 5];
36
37 console.log(typeof arr); // object
```

```
38
39 console.log(arr);
40
41 // ✅ Using Array Constructor
42
43 let fruitsArr = new Array("apple", "banana", "orange", "grapes", "pineapple");
44
45 console.log(fruitsArr);
46
47 // 💠 Accessing Elements in an Array
48
49 console.log(fruitsArr[0]); // apple
50
51 console.log(fruitsArr[1]); // banana
52
53 console.log(fruitsArr[2]); // orange
54
55 // 💠 Modifying Elements in an Array
56
57 fruitsArr[0] = "mango";
58
59 console.log(fruitsArr);
60
61 // 💠 Array Traversal
62
63 // ✅ 1. for...of loop → Iterates over values
64
65 console.log("Using for...of loop");
66
67 for (const fruitItem of fruitsArr) {
68     console.log(fruitItem);
69 }
70
71 // Output: mango, banana, orange, grapes, pineapple
72
73
74 // ✅ 2. for loop → Index-based iteration
75
```

O

```
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 // ✅ 3. for...in loop → Iterates over keys (indexes)
84
85 console.log("Using for...in loop");
86
87 for (const key in fruitsArr) {
88
89     console.log(key);
90 }
91
92 // ✅ 4. forEach() → Executes callback for each element. It doesn't return anything.
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);
109 });
110
111 // ✅ 5. map() → Creates a new array with the results of the callback
112
113 const newArr = fruitsArr.map((fruitItem, index) => {
```

O

```
114
115     return fruitItem + " " + index;
116 });
117
118 console.log(newArr);
119
120 // 💡 Array Update Methods
121
122 /*
123
124 ✓ 1. push(): Adds one or more elements to the end of an array.
125
126     ► Modifies the original array. Returns the new length.
127
128 ✓ 2. pop(): Removes the last element from an array.
129
130     ► Returns the removed element.
131
132 ✓ 3. unshift(): Adds one or more elements to the beginning of an array.
133
134     ► Modifies the original array. Returns the new length.
135
136 ✓ 4. shift(): Removes the first element from an array.
137
138     ► Returns the removed element.
139
140 ✓ 5. splice(): Removes elements from an array and optionally inserts new ones.
141
142     ► Returns the removed elements.
143
144 ✓ 6. split(): Splits a string into an array of substrings based on a separator.
145
146     ► Returns the new array of substrings.
147
148 ✓ 7. slice(): Returns a shallow copy of a portion of an array.
149
150     ► Returns the new array.
151
```

```
152  ✓ 8. concat(): Concatenates two or more arrays.
153
154  ➤ Returns the new concatenated array.
155
156  ✓ 9. join(): Joins all elements of an array into a string.
157
158  ➤ Returns the joined string.
159
160  */
161
162  const numbersArray = [1, 2, 3, 4, 5];
163
164  console.log(numbersArray);
165
166  numbersArray.push(6);      // [1, 2, 3, 4, 5, 6]
167
168  console.log(numbersArray);
169
170  numbersArray.pop();        // [1, 2, 3, 4, 5]
171
172  console.log(numbersArray);
173
174  numbersArray.unshift(0);   // [0, 1, 2, 3, 4, 5]
175
176  console.log(numbersArray);
177
178  numbersArray.shift();      // [1, 2, 3, 4, 5]
179
180  console.log(numbersArray);
181
182  numbersArray.splice(3, 1, 7); // [1, 2, 3, 7, 5]
183
184  console.log(numbersArray);
185
186  const stringArray = "Hello World";
187
188  console.log(stringArray.split("")); // ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
189
```

```
190 console.log(stringArray.slice(0, 5)); // Hello
191
192 const arr1 = [1, 2, 3];
193
194 const arr2 = [4, 5, 6];
195
196 const arr3 = arr1.concat(arr2);
197
198 console.log(arr3); // [1, 2, 3, 4, 5, 6]
199
200 console.log(arr3.join("")); // 123456
201
202 // ♦ Searching in an Array
203
204 /*
205
206 ✓ 1. indexOf(): Returns the index of the first occurrence of a specified value.
207
208   ► Returns -1 if the value is not found.
209
210 ✓ 2. lastIndexOf(): Returns the index of the last occurrence of a specified value.
211
212   ► Returns -1 if the value is not found.
213
214 ✓ 3. includes(): Returns true if an array contains a specified value.
215
216   ► Returns false if the value is not found.
217
218 */
219
220 const searchArray = [1, 2, 3, 6, 4, 5, 6, 7, 8, 9];
221
222 console.log(searchArray);
223
224 console.log("The index of 4 is", searchArray.indexOf(4)); // 4
225
226 console.log("The last index of 6 is", searchArray.lastIndexOf(6)); // 6
227
```

```
228 console.log("Array contains 2 ?", searchArray.includes(2)); // true
229
230 // ♦ Filter Methods
231
232 /*
233
234 ✓ 1. find(): Returns the value of the first element that satisfies the condition.
235
236 ✓ 2. findIndex(): Returns the index of the first element that satisfies the condition.
237
238 ✓ 3. filter(): Returns a new array with all elements that pass the condition.
239
240 ✓ 4. sort(): Sorts the array in place and returns it. Accepts a comparator function.
241
242 ✓ 5. every(): Returns true if all elements in the array pass the condition.
243
244 ✓ 6. some(): Returns true if at least one element in the array passes the condition.
245
246 ✓ 7. reduce(): Applies a function against an accumulator and each value of the array (from left to right) to reduce it to a
single value.
247
248 */
249
250 console.log("Filtering an Array");
251
252 const filterArray = [1, 2, 3, 4, 5, 6, 7, 8, 6, 9];
253
254 console.log(filterArray);
255
256 // find → First element > 5
257
258 const findElement = filterArray.find((currentElement) => {
259
260     return currentElement > 5;
261 });
262
263 console.log(findElement); // 6
264
```

```
265 // findIndex → Index of first element > 5
266
267 const findIndexElement = filterArray.findIndex((currentElement) => {
268
269     return currentElement > 5;
270 });
271
272 console.log(findIndexElement); // 5
273
274 // filter → All elements except value 6
275
276 let value = 6;
277
278 const newArray = filterArray.filter((currentElement) => {
279
280     return currentElement !== value;
281 });
282
283 console.log(newArray);
284
285 // every → All elements > 0
286
287 const everyElement = filterArray.every((currentElement) => {
288
289     return currentElement > 0;
290 });
291
292 console.log(everyElement); // true
293
294 // some → At least one element > 4
295
296 const someElement = filterArray.some((currentElement) => {
297
298     return currentElement > 4;
299 });
300
301 console.log(someElement); // true
302
```



```
303 // reduce → Sum of all elements - Initial value of accumulator is 0
304
305 const reduceElement = filterArray.reduce((accumulator, currentElement) => {
306
307     return accumulator + currentElement;
308
309 }, 0);
310
311 // ✅ Sort Method → Sorts the elements in ascending order or descending order using comparator logic
312
313 console.log("Sorting an Array in Ascending Order:");
314
315 const jumbledArray = [2, 5, 1, 4, 3, 9, 6, 8, 7];
316
317 jumbledArray.sort((a, b) => {
318
319     return a - b;
320 });
321
322 console.log(jumbledArray);
323
324 console.log("Sorting an Array in Descending Order:");
325
326 jumbledArray.sort((a, b) => {
327
328     return b - a;
329 });
330
331 console.log(jumbledArray);
```