Arrays — Creating, Modifying, Slicing & Sorting

AiCE Software Engineering Foundations — JavaScript & Node.js

You're comfortable with strings and numbers. Now let's move into JavaScript's two core data structures, starting with Arrays — the JavaScript version of Python lists. Objects come next; arrays first.

1. Creating & Modifying Arrays

An array in JavaScript is written with square brackets, and it uses zero-based indexing, exactly like a Python list. Create a file named arrays.js:

```
const students = ['Adesina', 'Goodnews', 'Chiamaka', 'Kwame'];

console.log(students[0]);
console.log(students[students.length - 1]);
console.log(students[10]);

students[1] = 'Israel';
console.log(students);

console.log(typeof students);
console.log(Array.isArray(students));
```

Save and run:

Line	Result / Behavior
students[0]	'Adesina' — the first item.
students[students.length - 1]	'Kwame' — the last item.
students[10]	undefined — JavaScript doesn't throw an error for a missing index, it just says nothing is there.
students[1] = 'Israel';	Works — arrays in JavaScript are mutable, unlike strings.

Line	Result / Behavior
typeof students	"object" — arrays are actually a special kind of object in JavaScript.
Array.isArray(students)	true — the correct way to check if something is specifically an array.

2. Adding, Removing & Finding Items

Create a file named array_methods.js:

```
const cities = ['Lagos', 'Accra', 'Nairobi'];

cities.push('Dakar');
console.log(cities);

cities.pop();
console.log(cities);

cities.unshift('Cairo');
console.log(cities);

cities.shift();
console.log(cities);

const scores = [45, 72, 38, 90];
console.log(scores.indexOf(90));
console.log(scores.includes(38));
console.log(scores.find(s => s >= 70));
console.log(scores.findIndex(s => s >= 70));
```

Method	What It Does
push(item)	Adds an item to the end of the array.
pop()	Removes the last item.
unshift(item)	Adds an item to the beginning.
shift()	Removes the first item.
indexOf(value)	Gives the position of a value, or -1 if not found.
includes(value)	Gives true or false depending on whether the value exists.
find(condition)	Gives back the first item matching a condition — here, the first score ≥ 70, which is 72.
findIndex(condition)	Same as find(), but gives the position of the match instead of the item itself — here, 1.

3. Slicing, Combining & Sorting

Let's write more code in the same file:

```
const west = ['Nigeria', 'Ghana'];
const east = ['Kenya', 'Tanzania'];

console.log(west.slice(0, 1));
const combined = [...west, ...east];
console.log(combined);

const scores = [85, 42, 90, 68];
console.log([...scores].sort());
console.log([...scores].sort((a, b) => a - b));
console.log([...scores].sort((a, b) => b - a));
```

slice() copies out a portion of an array without changing the original — west.slice(0, 1) gives just ['Nigeria'].

The three dots — ...west and ...east — are called the spread operator. It's the modern way of combining or copying arrays: [...west, ...east] pulls every item out of both arrays into one new array.

> **sort() Without a Comparison Function Sorts as Text**
> *[...scores].sort() with nothing inside it gives the wrong order for numbers, because by default sort() treats everything as text. To sort numbers correctly, pass sort() a comparison function — (a, b) => a - b for ascending order, or (a, b) => b - a for descending order.*

Notice scores was wrapped in [...scores] before sorting. This is because sort() changes the original array — using the spread operator makes a copy first, so the original scores array stays untouched.

4. Iterating With forEach

forEach() loops through every item in an array and runs code on each one — the same job Python's for item in list does. Create a file named foreach.js:

```
const students = [
    { name: 'Adesina', country: 'Nigeria', score: 85 },
    { name: 'Kwame', country: 'Ghana', score: 90 },
];

students.forEach((student, index) => {
    const status = student.score >= 70 ? 'PASS' : 'FAIL';
    console.log(`${index + 1}. ${student.name}
(${student.country}) - ${status}`);
});
```

Save and run — forEach() goes through the students array one item at a time, and for each student, prints their position, name, country, and whether they passed or failed.

Key Takeaways

Arrays use square brackets and zero-based indexing, and unlike strings, they're mutable — items can be reassigned directly. push()/pop() and unshift()/shift() add and remove

from either end, while indexOf(), includes(), find(), and findIndex() search an array. The spread operator (...) copies or combines arrays, sort() needs a comparison function to sort numbers correctly, and forEach() loops through every item. Notice that the students array in this lesson held objects — { name, country, score } — which is exactly where the next lesson picks up: Objects, JavaScript's version of Python dictionaries, with one extra power dictionaries don't have.