Objects — Properties, Methods & Nesting

AiCE Software Engineering Foundations — JavaScript & Node.js

Arrays are covered. Now let's move to Objects — the JavaScript version of Python dictionaries, but with one extra power dictionaries don't have: they can hold behavior, not just data.

1. Creating Objects & Accessing Properties

An object in JavaScript is written with curly brackets, holding key-value pairs — the same idea as a Python dictionary. Create a file named objects.js:

```
const student = {
    firstName: 'Adesina',
    age: 25,
    country: 'Nigeria',
    courses: ['Python', 'JavaScript'],
};

console.log(student.firstName);
console.log(student['country']);

const prop = 'age';
console.log(student[prop]);
console.log(student.email);

student.city = 'Lagos';
student.age = 26;
delete student.city;
console.log('age' in student);
```

Save and run:

Line	Result / Behavior
student.firstName	'Adesina' — dot notation, the standard way to access a property.
student['country']	'Nigeria' — bracket notation, the same result using a different syntax.
student[prop]	'26' — bracket notation is necessary when the property name is stored inside a variable, like prop here.
student.email	undefined, not an error — unlike Python, where a missing dictionary key raises a KeyError.
student.city = 'Lagos'; then delete student.city;	Adds a new property, then removes it with the delete keyword.
'age' in student	true — checks whether a property exists on the object.

2. Methods and this

Here is where JavaScript objects go further than Python dictionaries — an object can hold functions as properties. These are called methods. Inside a method, the keyword this refers to the object the method belongs to.

Create a file named object_methods.js:

```
const account = {
    owner: 'Goodnews Uche',
    balance: 250000,

    deposit(amount) {
        this.balance += amount;
        console.log(`Deposited ₦${amount}. New balance:
₦${this.balance}`);
    },
    checkBalance() {
```

```
        console.log(`${this.owner}: ₦${this.balance}`);
    },
};

account.checkBalance();
account.deposit(50000);
account.checkBalance();
```

Save and run — account.checkBalance() prints the owner's name and balance. account.deposit(50000) adds 50000 to the balance and prints a confirmation. Calling checkBalance() again shows the balance has genuinely changed.

> **What this Actually Means**
> *this.balance simply means "the balance property that belongs to whichever object called this method." Every object manages its own data through its own methods this way.*

3. Looping Through Objects & Nesting

Let's write more code in the same file:

```
const developer = { name: 'Kofi Otieno', country: 'Kenya',
level: 'Expert' };

for (const key in developer) {
    console.log(`${key}: ${developer[key]}`);
}

console.log(Object.keys(developer));
console.log(Object.values(developer));

const talent = {
    name: 'Nkrumah Kwame',
```

```
    contact: { email: 'nkrumah@aice.africa', address: { city:
'Accra' } },
};
console.log(talent.contact.address.city);
```

Line / Concept	What It Does
for (const key in developer)	Loops through each key in the object, printing every property name and its value.
Object.keys(developer)	Gives all the property names as an array.
Object.values(developer)	Gives all the values as an array.
talent.contact.address.city	Objects can contain other objects — chain the dots to reach a deeply nested value.

4. Practice: An African Tech Talent Registry

Let's build a small registry combining arrays and objects together. Create a file named registry.js:

```
// AiCE African Tech Talent Registry
const registry = {
    talents: [],
    addTalent(name, country, skills) {
        const talent = { id: this.talents.length + 1, name,
country, skills };
        this.talents.push(talent);
        console.log(`${name} registered with ID
${talent.id}.`);
    },
    findByCountry(country) {
```

```
        return this.talents.filter(t => t.country ===
country);
    },
};

registry.addTalent('Adesina', 'Nigeria', ['Python',
'JavaScript']);
registry.addTalent('Kwame', 'Ghana', ['JavaScript', 'React']);
registry.addTalent('Chiamaka', 'Nigeria', ['Python',
'MySQL']);

const nigerians = registry.findByCountry('Nigeria');
nigerians.forEach(t => console.log(`${t.name}:
${t.skills.join(', ')}`));
```

Save and run — registry is an object holding an array of talents, plus its own methods for adding a talent and finding talents by country. filter() works like find(), but instead of giving back one match, it gives every item matching the condition, as a new array. Push everything built in this lesson to GitHub.

Key Takeaways

Objects use curly brackets and key-value pairs, accessed with dot notation or bracket notation — bracket notation being required whenever the property name is stored in a variable. Accessing a missing property returns undefined rather than an error, unlike Python's KeyError. Objects can hold methods, and inside a method, this refers to the object that called it — the mechanism behind every object managing its own data. for...in, Object.keys(), and Object.values() let you loop through an object's contents, and objects can nest inside each other, reached by chaining dots. With arrays and objects both covered, the next lesson moves into control flow — conditionals, comparison and logical operators, and every type of loop JavaScript offers.