JavaScript Objects and Arrays

Understanding how to structure, access, and manipulate complex data using Objects and Arrays in JavaScript.

R.A: Assad Ullah Khan

assadullahkhan

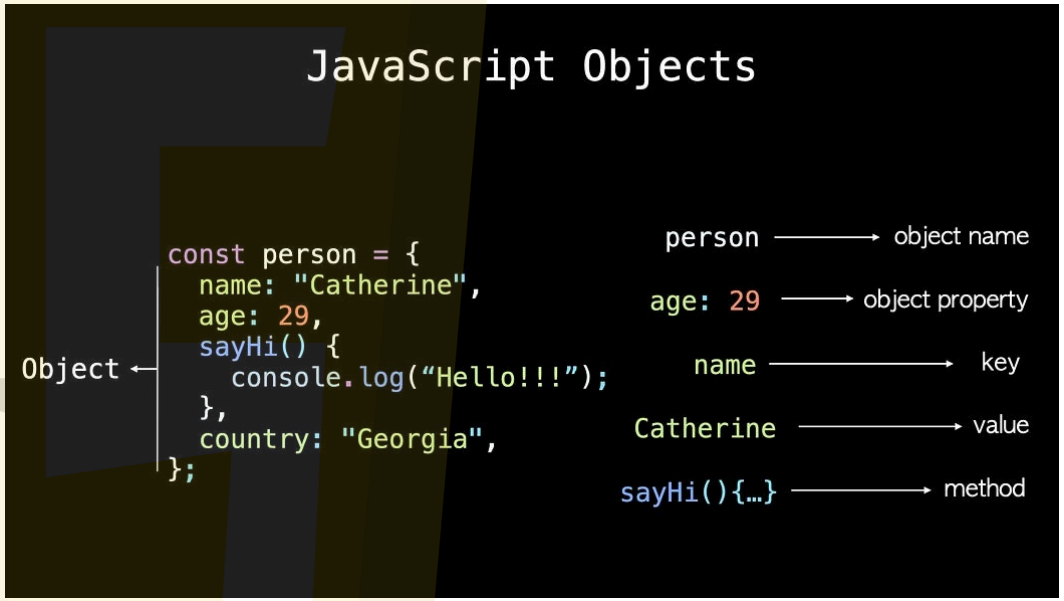
AssadUllahKhan

Objects Section

What are Objects in JavaScript?

Concept

- In JavaScript, an object is a collection of key–value pairs.
- Each key (called a property) represents some data or behavior of that object.
- Objects allow you to store multiple related values in a single variable.



Key Points

- Keys are strings (or symbols), and values can be any data type (string, number, array, another object, or even a function).
- Think of an object as a real-world entity, like a student, car, or product, each having properties.

Creating and Accessing Object Properties

Concept

- Objects in JavaScript are created using curly braces {}, containing key, value pairs.
- Each key is the property name, and each value can be any data type.

You can access object properties using:

- **Dot notation** → objectName.propertyName
- **Bracket notation** → objectName["propertyName"]

Creating and Accessing Object Properties

```
let student = {
  name: "Assad",
  age: 22,
  department: “Computer Science"
  course: "Web Development"
  isGraduate: false
};

// Accessing Properties
console.log(student.name);          // Output: Asad
console.log(student[“age”]);        // Output: 22
```

Property	Value
name	Assad
Age	22
department	Computer Science

Adding, Modifying, and Deleting Properties

Concept

JavaScript objects are dynamic, meaning you can add new properties, change existing ones, or delete them anytime — even after the object is created.

1. Adding a Property

You can add a property using dot or bracket notation.

```
const car = {
  brand: "Toyota",
  model: "Corolla"
};

// Add new property
car.year = 2022;
car["color"] = "White";

console.log(car);
// Output: { brand: 'Toyota', model: 'Corolla', year: 2022, color: 'White' }
```

2. Modifying a Property

If a property already exists, assigning a new value will update it.

```
car.color = "Black";
console.log(car.color);      // Output: Black
```

3. Deleting a Property

Use the delete keyword to remove a property from the object.

```
delete car.year;
console.log(car);
// Output: { brand: 'Toyota', model: 'Corolla', color: 'Black' }
```

Object Methods (Functions Inside Objects)

Concept

When a function is defined inside an object, it becomes an object method.Methods define the behavior of the object — what it can do.
Objects often combine data (properties) and behavior (methods) to represent real-world entities.

Explanation

- introduce is a method (a function inside an object).
- The keyword this refers to the current object (student).
- You can call the method using dot notation, e.g., student.introduce().

```
const student = {
  name: "Asad",
  age: 22,
  department: "Computer Science",

  // Defining a method
  introduce: function() {
    console.log(`Hi, I'm ${this.name}, studying ${this.department}.`);
  }
};

// Calling the method
student.introduce();
// Output: Hi, I'm Asad, studying Computer Science.
```

Using the this Keyword in Objects

Concept

The this keyword in JavaScript refers to the current object that owns the method being executed.It allows methods to access and modify the object's properties dynamically.

Key Points

Inside an object's method, this points to that object.
The value of this changes depending on where it's used.
In global scope, this refers to the window object (in browsers).

```
const student = {
  name: "Asad",
  age: 22,
  department: "Computer Science",
  // Defining a method
  introduce: function() {
    console.log(`Hi, I'm ${this.name}, studying ${this.department}.`);
  }
};
// Calling the method
student.introduce();
// Output: Hi, I'm Asad, studying Computer Science.
```

Using the this Keyword in Objects

Why this is Important

Without this, you'd need to hardcode property names, which breaks flexibility:

```
// ❌ Not ideal
console.log(`Hello, my name is person.name`);
```

Using this ensures that the code remains reusable for different objects.

Looping Through Objects (for...in)

Concept

When you want to iterate (loop) through all the properties of an object, you can use the for...in loop.It allows you to access every key (property name) in an object one by one.

Syntax

```
for (let key in objectName) {
  // code block
}
```

Here:

- key → represents each property name (string).
- objectName[key] → gives you the property value.

Looping Through Objects (for...in)

Example Code

```
const student = {
  name: "Asad",
  age: 22,
  department: "Computer Science"
};

// Loop through object
for (let key in student) {
  console.log(`${key}: ${student[key]}`);
}

/* Output:
name: Asad
age: 22
department: Computer Science
*/
```

Important Tip

Use for...in only for objects, not arrays.

Object Destructuring

Concept

Object Destructuring is a modern JavaScript feature (ES6) that allows you to extract values from objects and assign them to variables easily and cleanly.
Instead of accessing each property manually, destructuring simplifies the process into a single line of code.

Without Destructuring

```
const student = {
  name: "Asad",
  age: 22,
  department: "Computer Science"
};

const name = student.name;
const age = student.age;
const department = student.department;

console.log(name, age, department);
```

Object Destructuring

With Destructuring

```
const student = {
  name: "Asad",
  age: 22,
  department: "Computer Science"
};

// Destructuring
const { name, age, department } = student;

console.log(name);          // Asad
console.log(age);           // 22
console.log(department);          // Computer Science
```

Object Destructuring

Rename or Set Default Values

You can rename variables or set default values during destructuring:

```
const { name: studentName, hobby = "Coding" } = student;
console.log(studentName);          // Asad
console.log(hobby);        // Coding (default)
```

Arrays Section

Introduction to Arrays

Concept

- An array in JavaScript is a special type of object used to store multiple values in a single variable.
- Arrays are ordered, and each element has an index (starting from 0).

You can store numbers, strings, objects, or even other arrays inside an array.

Creating an Array

```
// Using square brackets
const fruits = ["Apple", "Banana", "Mango"];

// Using Array constructor
const numbers = new Array(10, 20, 30);
```

Accessing Array Elements

```
console.log(fruits[0]);      // Output: Apple
console.log(fruits[2]);      // Output: Mango
```

Introduction to Arrays

Array Characteristics

- Arrays are zero-indexed → first element is at index 0.
- You can modify values directly by using the index.
- Arrays can hold mixed data types.

```
const mixed = ["Asad", 22, true];
console.log(mixed[1]);      // Output: 22
```

Changing Array Values

```
fruits[1] = "Orange";
console.log(fruits);          // ["Apple", "Orange", "Mango"]
```

Accessing and Modifying Array Elements

Concept

- Arrays are lists that store multiple values in a single variable.
- Each value (called an element) has a numeric index, starting from 0.
- You can create, access, and modify array elements easily using these indexes.

Modifying

```
languages[1] = "Java";
console.log(languages);    // ["JavaScript", "Java", "C++"]
```

Adding Elements Beyond Length

If you add an element at a non-existing index, the array automatically expands:

```
languages[3] = "Go";
console.log(languages);    // ["JavaScript", "Java", "C++", "Go"]
```

Accessing and Modifying Array Elements

Array Length Property

You can find how many elements are in an array:

```
console.log(languages.length); // Output: 4
```

Common Array Methods

Concept

- JavaScript provides several built-in methods to easily add or remove elements from arrays.
- These methods modify the contents and length of the array dynamically.

1: push() – Add at the End

- Adds one or more elements to the end of the array

```
const fruits = ["Apple", "Banana"];
fruits.push("Mango");
console.log(fruits);          // ["Apple", "Banana", "Mango"]
```

2: pop() – Remove from the End

- Removes the last element from the array.

```
fruits.pop();
console.log(fruits);          // ["Apple", "Banana"]
```

Common Array Methods

3: shift() – Remove from the Start

- Removes the first element and shifts all others left.

```
fruits.shift();
console.log(fruits);          // ["Banana"]
```

4: unshift() – Add at the Start

- Adds one or more elements to the beginning of the array.

```
fruits.unshift();
console.log(fruits);          // [« Orange, Banana"]
```

Common Array Methods

Summary

Method	Action	Position
push()	Add element	End
pop()	Remove element	End
shift()	Remove element	Start
unshift()	Add element	Start

Advanced Array Methods

Concept

- JavaScript provides powerful array methods to handle data efficiently and perform transformations without traditional loops.

map()

- Creates a new array by applying a function to each element.

```
const numbers = [1, 2, 3];
const doubled = numbers.map(num => num * 2);
// [2, 4, 6]
```

filter()

- Returns elements that pass a specific condition.

```
const even = numbers.filter(num => num % 2 === 0);
// [2]
```

Advanced Array Methods

forEach()

- Executes a function for each element, but does not return a new array.

```
numbers.forEach(num => console.log(num));
```

reduce()

- Reduces an array to a single value (like sum or product).

```
const sum = numbers.reduce((acc, num) => acc + num, 0);
// 6
```

Key Benefit: These methods make code cleaner, shorter, and more readable than traditional loops.

Array of Objects

Concept

- In JavaScript, arrays can store objects — allowing you to organize complex, structured data efficiently.
  - Each object can represent an entity with multiple properties.
  - You can use array methods like map(), filter(), and forEach() to process these objects easily.

Example

```
const students = [
  { name: "Ali", age: 21, grade: "A" },
  { name: "Sara", age: 20, grade: "B" },
  { name: "Ahmed", age: 22, grade: "A" }
];

// Filter students with grade A
const topStudents = students.filter(s => s.grade === "A");

// Get only student names
const names = students.map(s => s.name);
```

Sorting, Merging, and Slicing Arrays

Concept

- Arrays in JavaScript can be sorted, merged, and sliced using built-in methods for better data handling and organization.

Sorting Arrays

- Used to arrange array elements alphabetically or numerically.

```
const numbers = [4, 1, 8, 3];
numbers.sort((a, b) => a - b);          // [1, 3, 4, 8]
```

Merging Arrays

- Combine multiple arrays into one.

```
const arr1 = [1, 2];
const arr2 = [3, 4];
const merged = arr1.concat(arr2);
// [1, 2, 3, 4]
```

OR

```
const merged = [...arr1, ...arr2];
```

Sorting, Merging, and Slicing Arrays

Slicing Arrays

- Extract a portion of an array without changing the original.

```
const fruits = ["apple", "banana", "cherry", "mango"];
const citrus = fruits.slice(1, 3);                    // ["banana", "cherry"]
```

Key Idea:

Use these methods to organize, combine, and extract specific parts of data efficiently.

JSON (Intro, Syntax, and Conversion Object ↔ JSON)

Concept

- JSON stands for JavaScript Object Notation — a lightweight data format used for storing and transferring data between a client and server.

What is JSON?

- A text-based format that represents structured data.
- Easy to read and write for humans and machines.
- Commonly used in APIs and data exchange.

JSON Syntax

```
{
  "name": "Assad",
  "age": 23,
  "skills": ["JavaScript", "Python", "AI"]
}
```

JSON (Intro, Syntax, and Conversion Object ↔ JSON)

Convert Object ↔ JSON

```
// Object to JSON
const obj = { name: "Ali", age: 25 };
const jsonString = JSON.stringify(obj);

// JSON to Object
const parsedObj = JSON.parse(jsonString);
```

Practice Challenge: Manage a Student Record System

Objective

- Build a small Student Record System using objects and arrays in JavaScript.You'll apply all concepts learned — arrays, objects, methods, loops, and functions.

Tasks to Perform:

1) Create an array named students.
2) Each student should be an object with properties:
   - name, age, grade, and department.
3) Add methods to:
   - Display all students.
   - Add a new student.
   - Find a student by name.
4) Use loops, object destructuring, and array methods (forEach, filter, push).

Practice Challenge: Manage a Student Record System

Example Code:

```
const students = [
  { name: "Asad", age: 22, grade: "A", department: "CS" },
  { name: "Ali", age: 21, grade: "B", department: "IT" },
  { name: "Sara", age: 23, grade: "A", department: "SE" }
];

// Add a new student
students.push({ name: "Bilal", age: 22, grade: "B+", department:
"CS" });

// Display all students
students.forEach(s => {
  console.log(`${s.name} (${s.department}) - Grade: ${s.grade}`);
});

// Find a student
const found = students.filter(s => s.name === "Asad");
console.log(found);
```

Any Question

Thank You