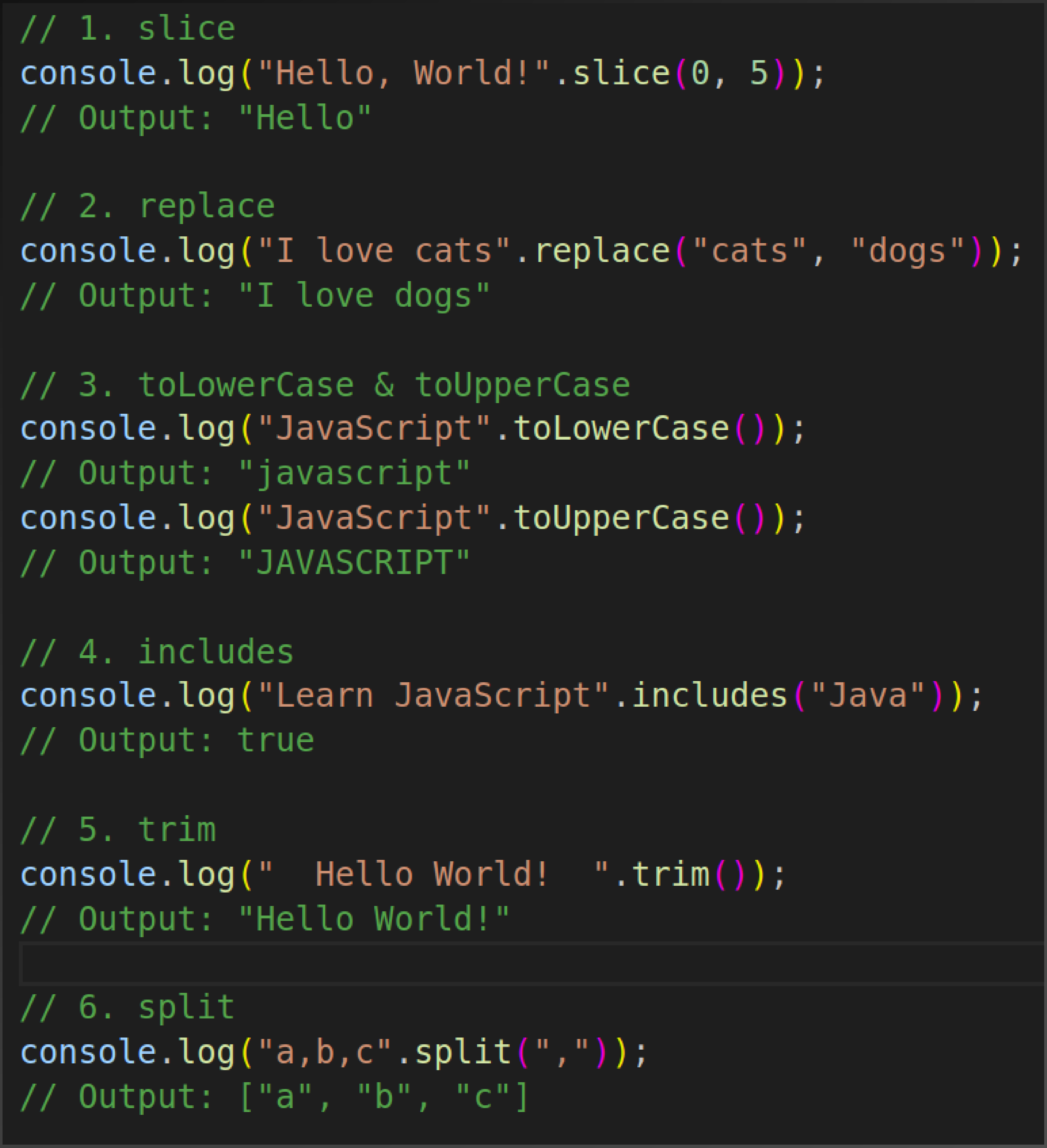
JavaScript

Must-Know JavaScript Methods

String Methods: Handle Text Easily

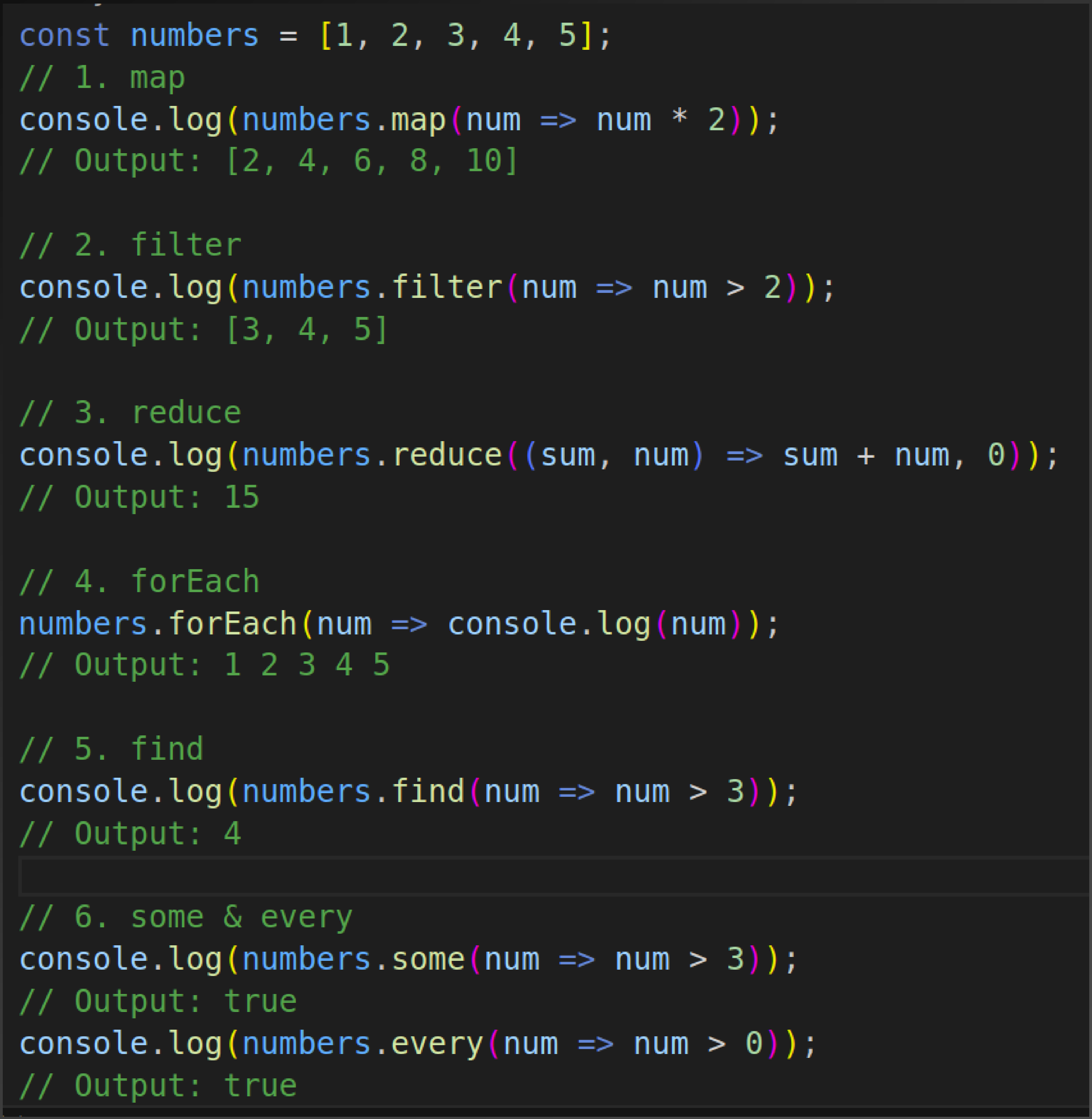
String methods let you manipulate and format text efficiently. From extracting, replacing, or splitting **strings**, these are essential for handling user **input** or **generating dynamic content**.



```javascript
// 1. slice
console.log("Hello, World!".slice(0, 5));
// Output: "Hello"

// 2. replace
console.log("I love cats".replace("cats", "dogs"));
// Output: "I love dogs"

// 3. toLowerCase & toUpperCase
console.log("JavaScript".toLowerCase());
// Output: "javascript"
console.log("JavaScript".toUpperCase());
// Output: "JAVASCRIPT"

// 4. includes
console.log("Learn JavaScript".includes("Java"));
// Output: true

// 5. trim
console.log("  Hello World!  ".trim());
// Output: "Hello World!"

// 6. split
console.log("a,b,c".split(","));
// Output: ["a", "b", "c"]
```

Array Methods: Simplify Data Processing

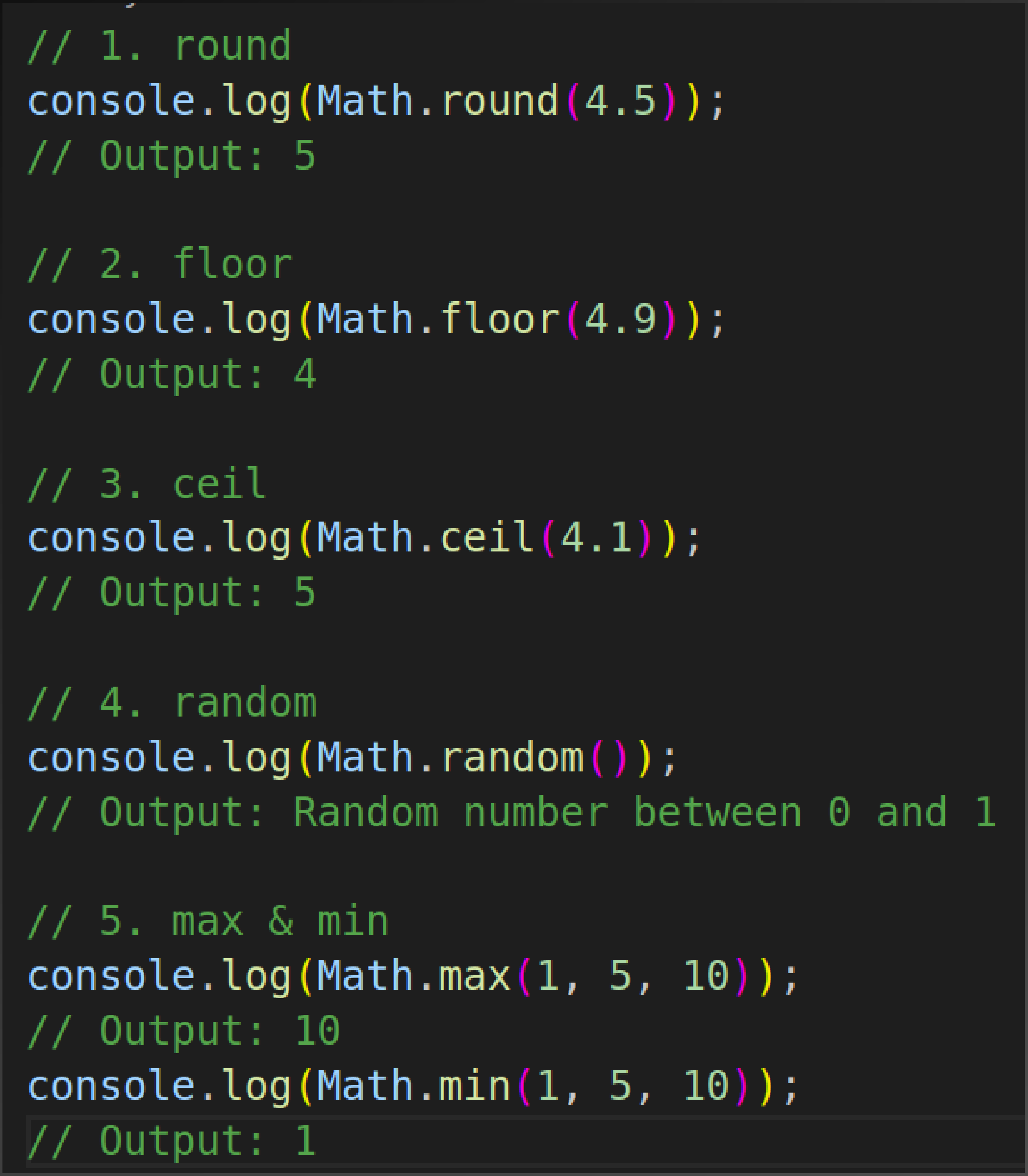
Array methods make data handling a breeze. **Filter**, **map**, **reduce**, and **transform arrays** effortlessly to work with lists, **APIs**, or **dynamic data**.



```javascript
const numbers = [1, 2, 3, 4, 5];
// 1. map
console.log(numbers.map(num => num * 2));
// Output: [2, 4, 6, 8, 10]

// 2. filter
console.log(numbers.filter(num => num > 2));
// Output: [3, 4, 5]

// 3. reduce
console.log(numbers.reduce((sum, num) => sum + num, 0));
// Output: 15

// 4. forEach
numbers.forEach(num => console.log(num));
// Output: 1 2 3 4 5

// 5. find
console.log(numbers.find(num => num > 3));
// Output: 4

// 6. some & every
console.log(numbers.some(num => num > 3));
// Output: true
console.log(numbers.every(num => num > 0));
// Output: true
```

Math Methods: Perform Calculations

The **Math** object provides tools for rounding, finding **max/min**, and generating random **numbers**, ideal for **games**, **stats**, or any **math-related task**.



```
// 1. round
console.log(Math.round(4.5));
// Output: 5

// 2. floor
console.log(Math.floor(4.9));
// Output: 4

// 3. ceil
console.log(Math.ceil(4.1));
// Output: 5

// 4. random
console.log(Math.random());
// Output: Random number between 0 and 1

// 5. max & min
console.log(Math.max(1, 5, 10));
// Output: 10
console.log(Math.min(1, 5, 10));
// Output: 1
```

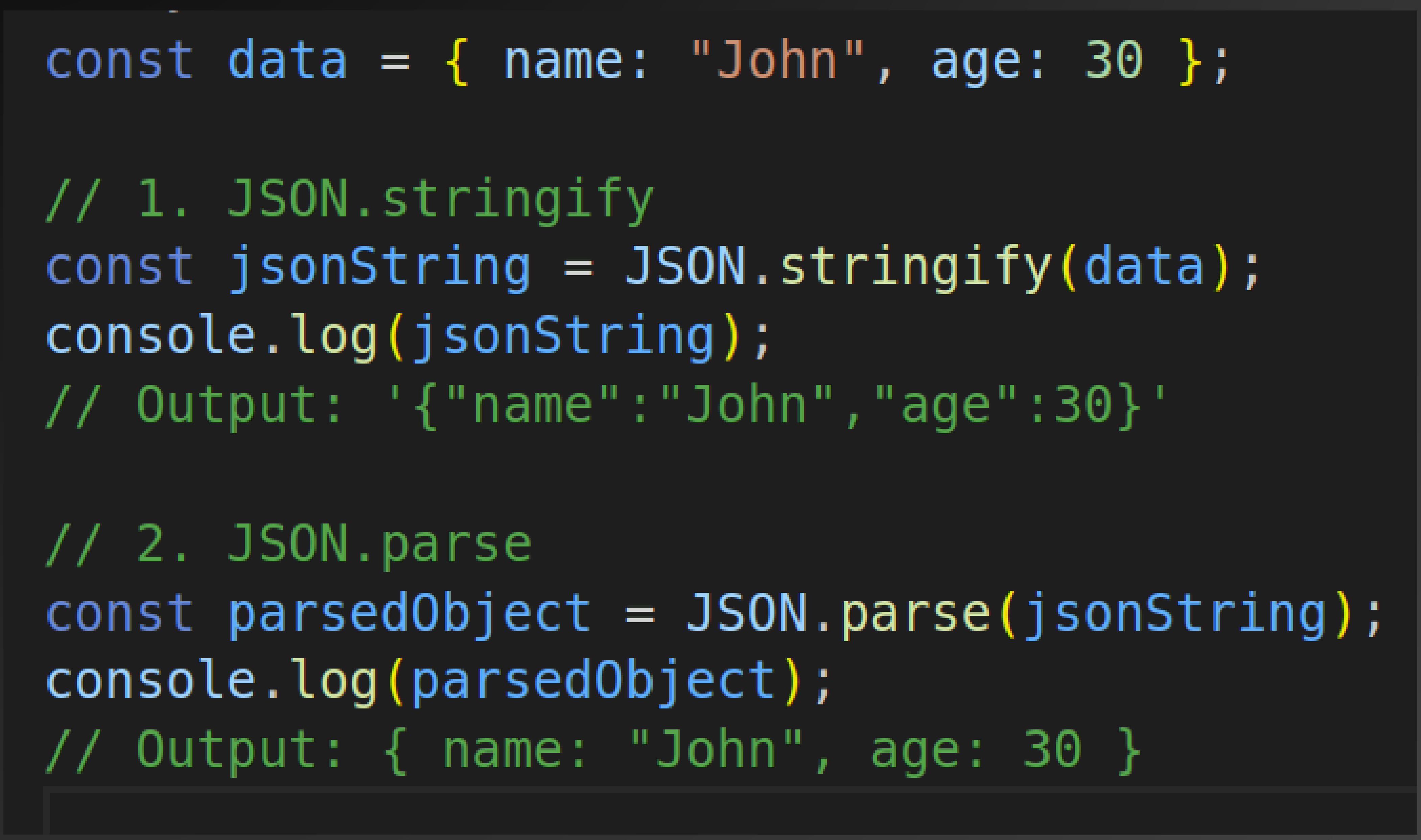
Date Methods: Work with Time

Date methods simplify working with **timestamps** and formatting **dates**, essential for **scheduling**, **tracking**, and **displaying time** in apps.

```javascript
const today = new Date();

// 1. new Date
console.log(today);
// Output: Current date and time

// 2. getFullYear
console.log(today.getFullYear());
// Output: Current year

// 3. getMonth
console.log(today.getMonth());
// Output: Current month (0-11)

// 4. toLocaleDateString
console.log(today.toLocaleDateString());
// Output: "MM/DD/YYYY" (format varies by locale)

// 5. getTime
console.log(today.getTime());
// Output: Timestamp in milliseconds
```

JSON Methods: Manage API Data

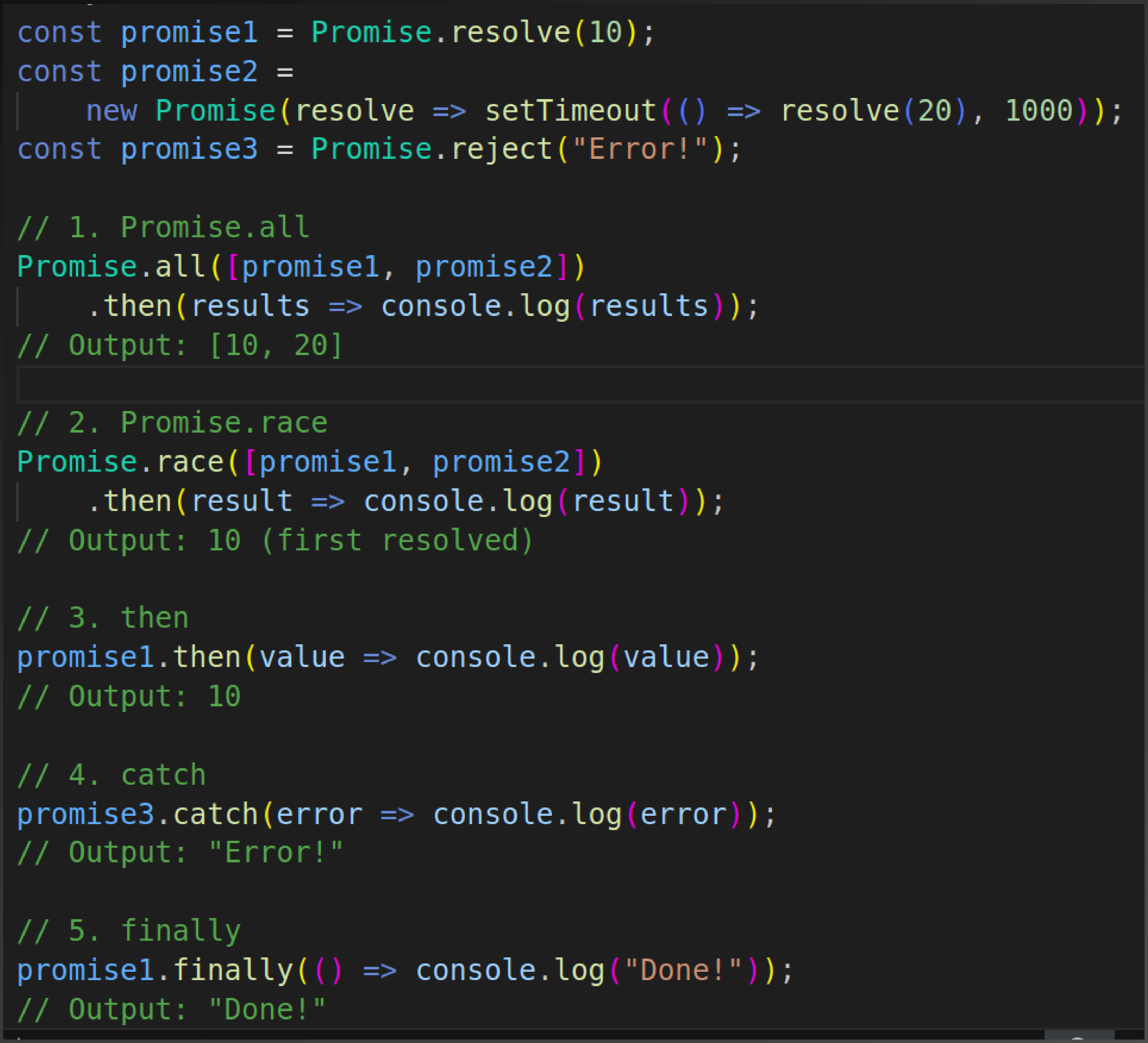
JSON methods help convert objects to **JSON** and parse **JSON** into **objects**, **making** them **crucial** for handling **API data** seamlessly.



```javascript
const data = { name: "John", age: 30 };

// 1. JSON.stringify
const jsonString = JSON.stringify(data);
console.log(jsonString);
// Output: '{"name":"John","age":30}'

// 2. JSON.parse
const parsedObject = JSON.parse(jsonString);
console.log(parsedObject);
// Output: { name: "John", age: 30 }
```

Promise Methods: Handle Async Tasks

Promises make handling **async** tasks easy. Use them for **API calls**, file **handling**, or **running** multiple **tasks** in **parallel** with **efficiency**.



```javascript
const promise1 = Promise.resolve(10);
const promise2 =
    new Promise(resolve => setTimeout(() => resolve(20), 1000));
const promise3 = Promise.reject("Error!");

// 1. Promise.all
Promise.all([promise1, promise2])
    .then(results => console.log(results));
// Output: [10, 20]

// 2. Promise.race
Promise.race([promise1, promise2])
    .then(result => console.log(result));
// Output: 10 (first resolved)

// 3. then
promise1.then(value => console.log(value));
// Output: 10

// 4. catch
promise3.catch(error => console.log(error));
// Output: "Error!"

// 5. finally
promise1.finally(() => console.log("Done!"));
// Output: "Done!"
```