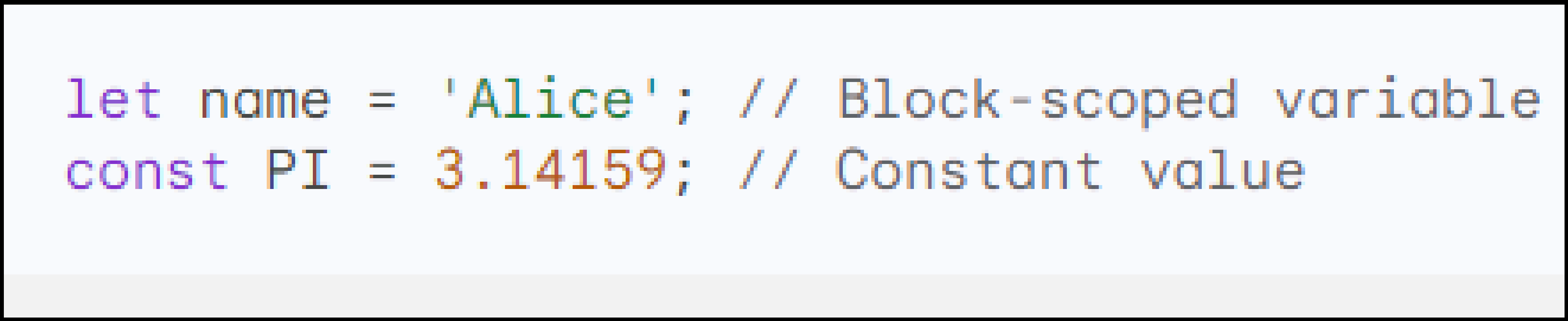
ES6 Cheat Sheet

Let & Const

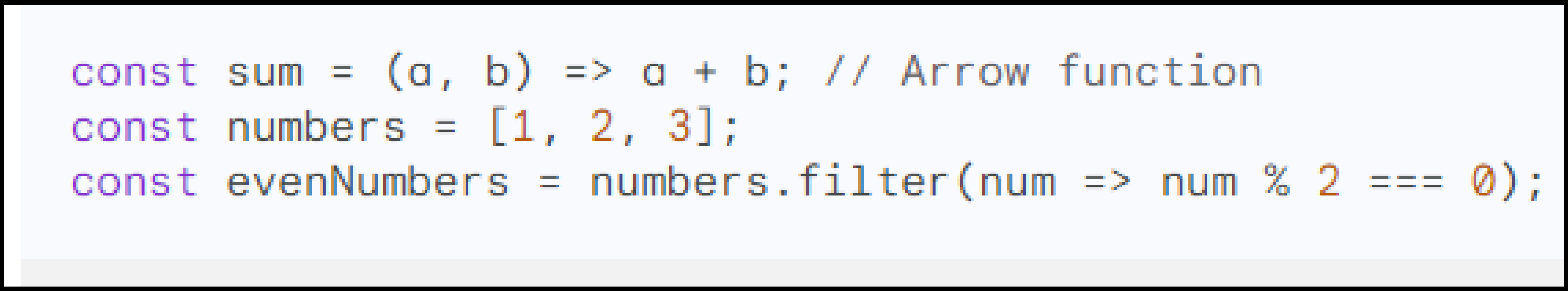
Say goodbye to var! ES6 introduces let and const for variable declaration, offering better block-level scoping and preventing accidental reassignments.



```
let name = 'Alice'; // Block-scoped variable
const PI = 3.14159; // Constant value
```

Arrow Functions

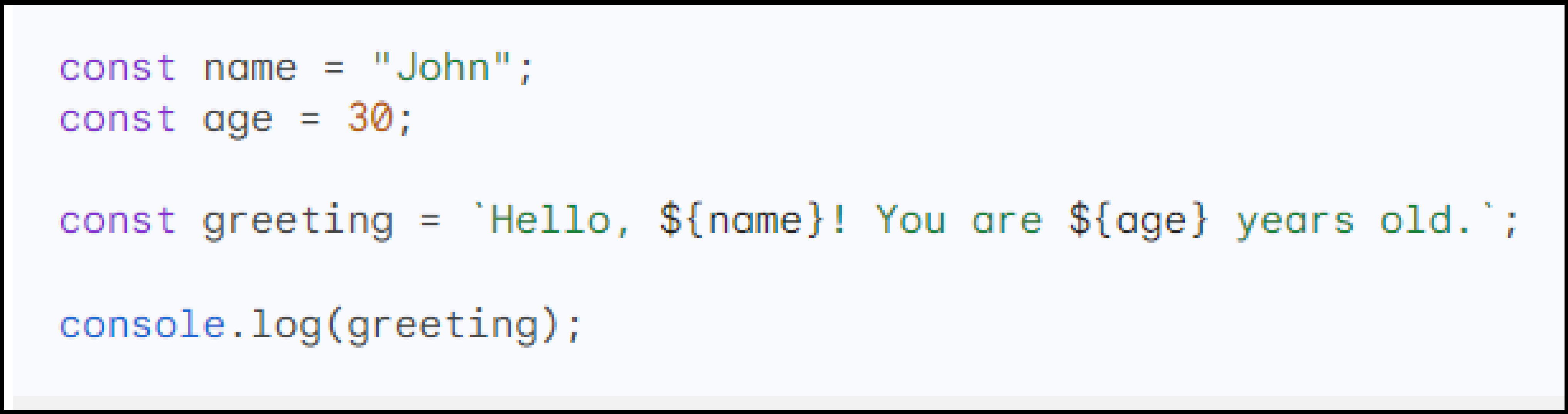
These concise and elegant functions are perfect for anonymous functions and callbacks.



```javascript
const sum = (a, b) => a + b; // Arrow function
const numbers = [1, 2, 3];
const evenNumbers = numbers.filter(num => num % 2 === 0);
```

Template Literals

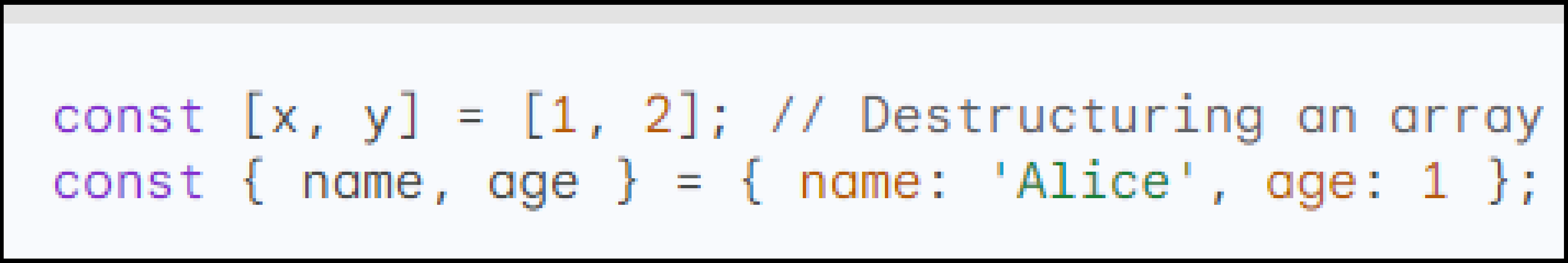
Say goodbye to string concatenation! Template literals provide a clean and flexible way to work with strings.



```javascript
const name = "John";
const age = 30;

const greeting = `Hello, ${name}! You are ${age} years old.`;

console.log(greeting);
```

Destructuring

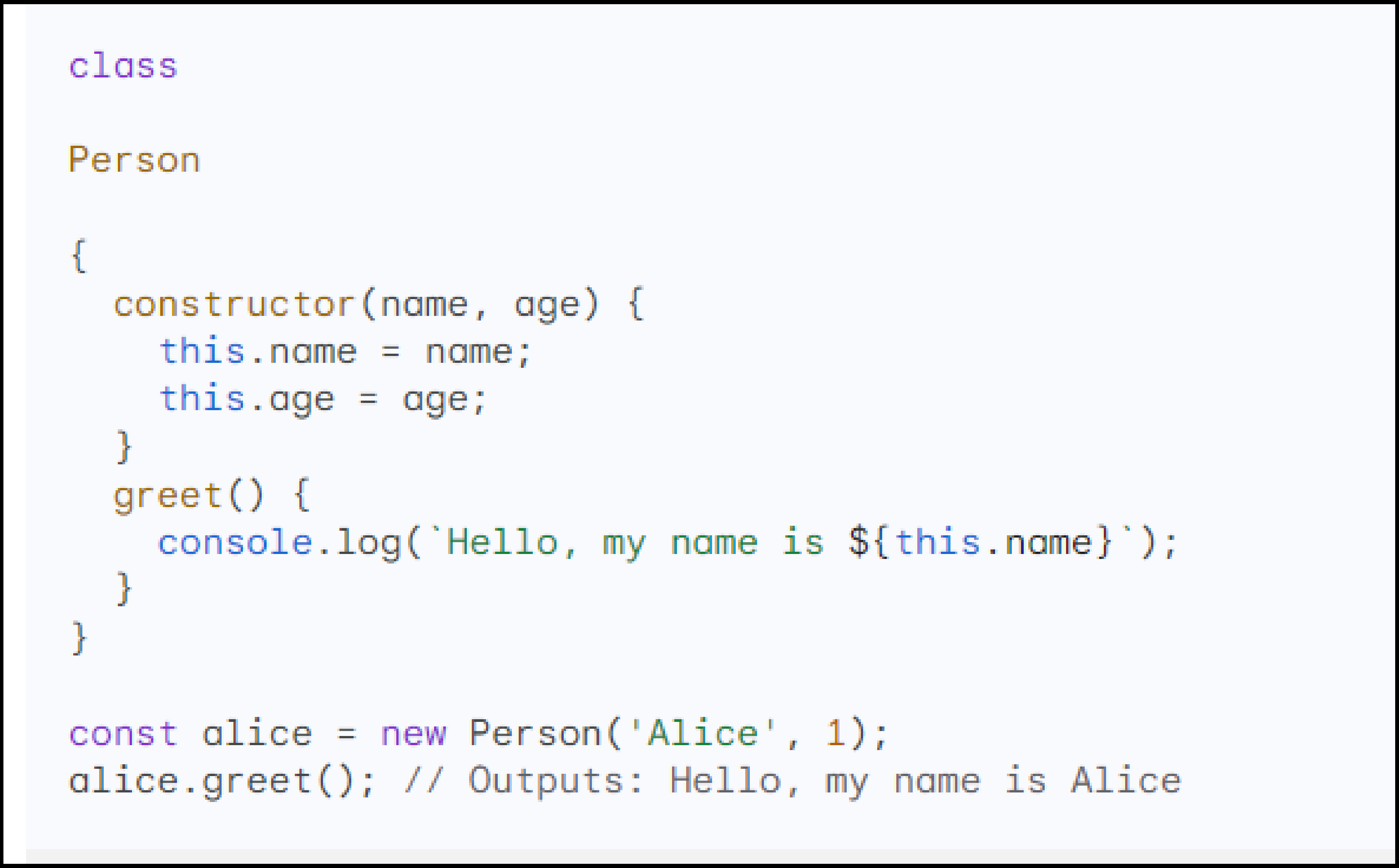
Extract specific values from data structures like arrays and objects with ease.



```
const [x, y] = [1, 2]; // Destructuring an array
const { name, age } = { name: 'Alice', age: 1 };
```

Classes

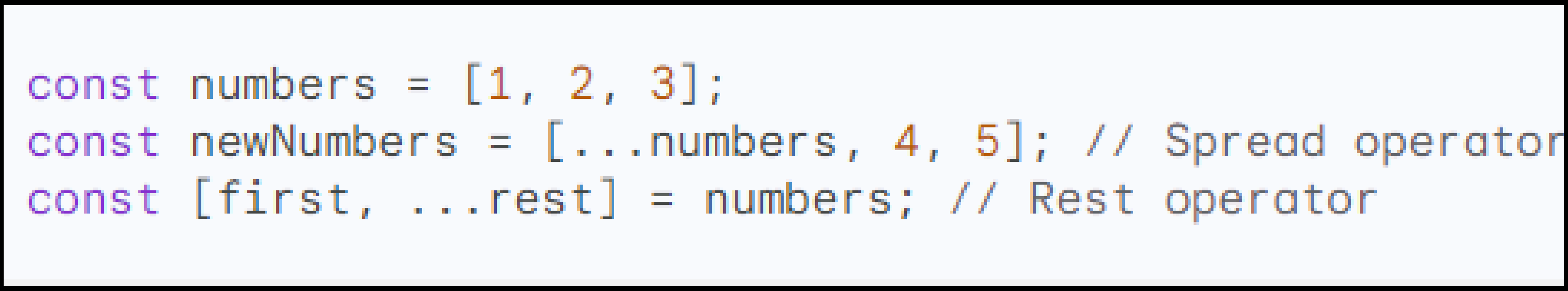
Object-oriented programming gets a boost with ES6 classes.



```
class

Person

{
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  greet() {
    console.log(`Hello, my name is ${this.name}`);
  }
}

const alice = new Person('Alice', 1);
alice.greet(); // Outputs: Hello, my name is Alice
```

Spread & Rest Operators

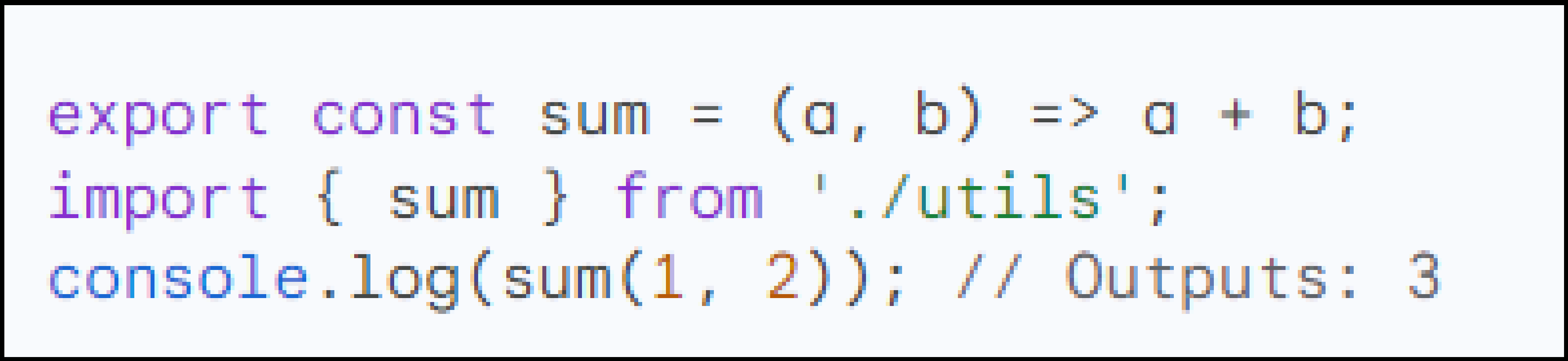
Spread operator (...) expands an iterable into individual elements, while rest operator (...) gathers remaining elements into an array.



```
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4, 5]; // Spread operator
const [first, ...rest] = numbers; // Rest operator
```

Modules

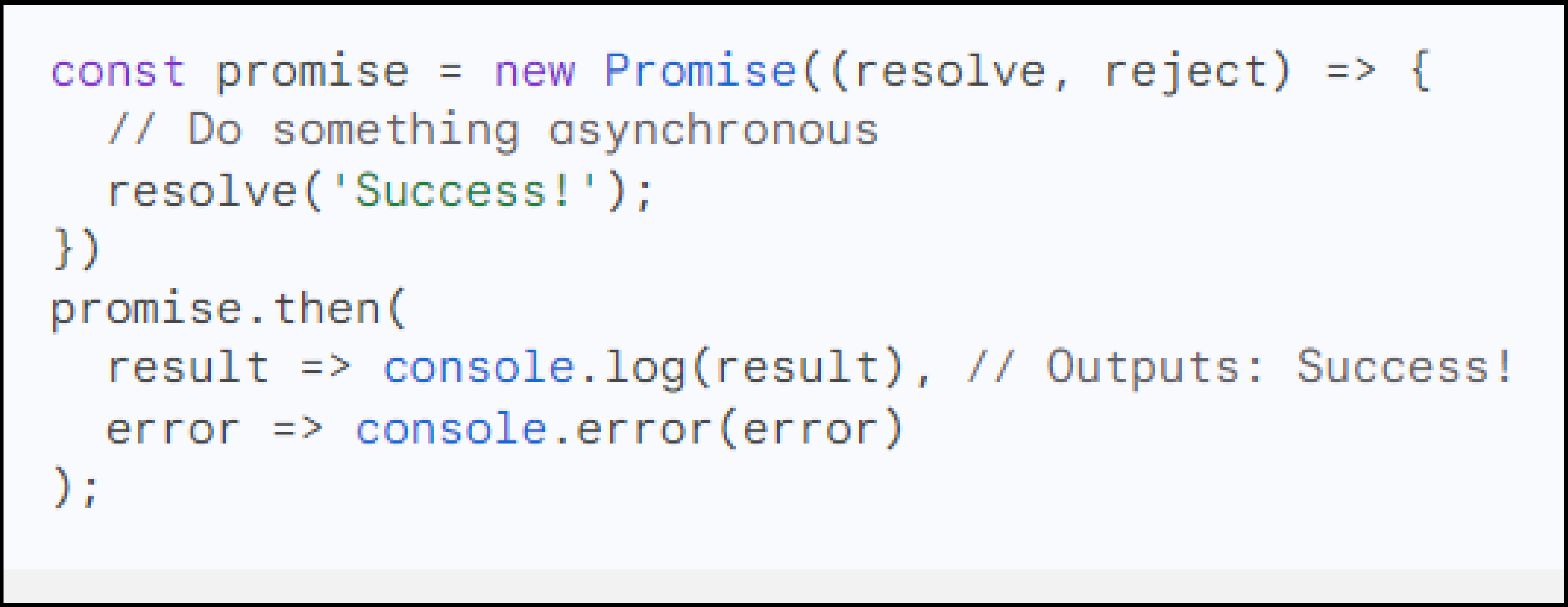
Organize your code into self-contained modules for better maintainability and modularity.



```
export const sum = (a, b) => a + b;
import { sum } from './utils';
console.log(sum(1, 2)); // Outputs: 3
```

Promises

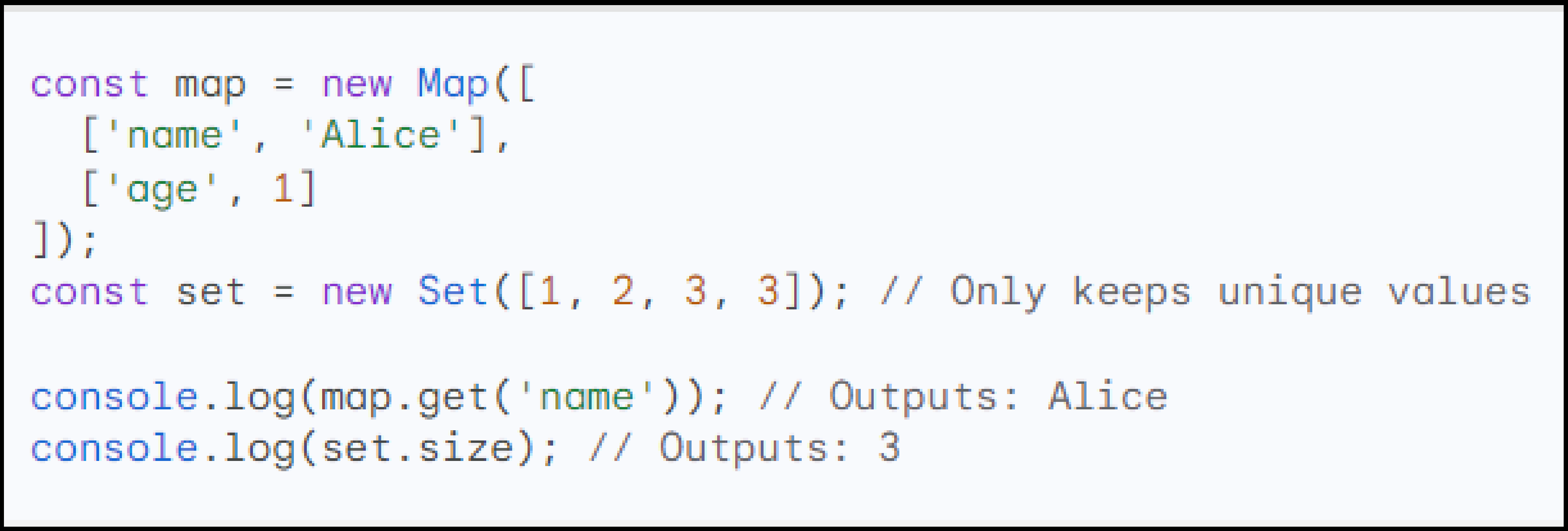
Handle asynchronous operations efficiently with promises.



```javascript
const promise = new Promise((resolve, reject) => {
  // Do something asynchronous
  resolve('Success!');
})
promise.then(
  result => console.log(result), // Outputs: Success!
  error => console.error(error)
);
```

Maps & Sets

Efficiently store unique values and key-value pairs.



```
const map = new Map([
  ['name', 'Alice'],
  ['age', 1]
]);
const set = new Set([1, 2, 3, 3]); // Only keeps unique values

console.log(map.get('name')); // Outputs: Alice
console.log(set.size); // Outputs: 3
```

Follow For More