Different use cases of Array.prototype.reduce() method

- The Array.reduce method in JavaScript is a powerful higher-order function that can be used to perform a wide range of operations on an array.

- The reduce method takes a callback function as its first argument, which is executed on each element of the array. The callback function takes two arguments: an accumulator (which stores the result of the previous callback invocation) and the current value of the array element being processed.

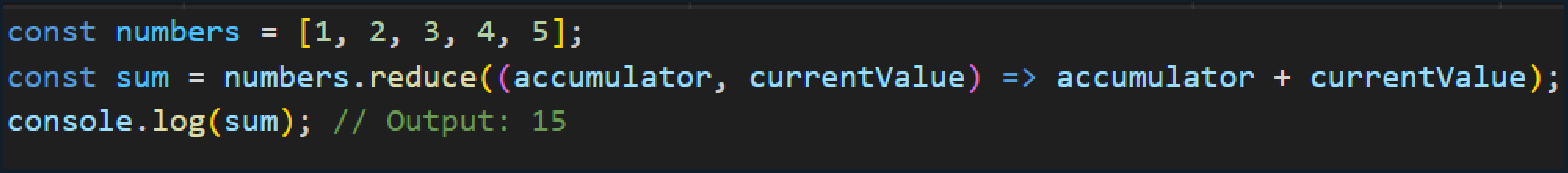
- The callback function can also take two optional arguments: the current index of the element being processed and the entire array being processed.

- The reduce method also takes an optional second argument, which is the initial value of the accumulator. If this argument is not provided, the first element of the array is used as the initial value of the accumulator and the iteration starts from the second element.

- The reduce method returns the final value of the accumulator after the last iteration. This final value can be of any data type, including arrays and objects.

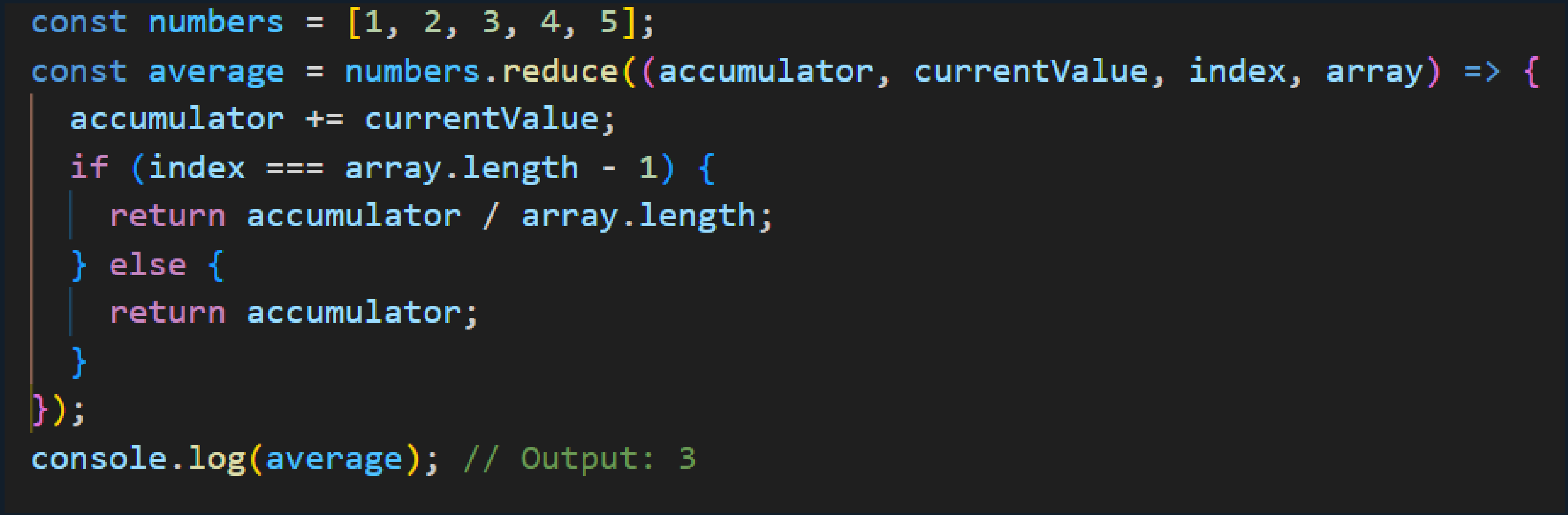
- When using reduce, it is important to always return the accumulator from the callback function, as this is what will be used as the accumulator value in the next iteration.

Use Case 1: Summing up all the values in an array



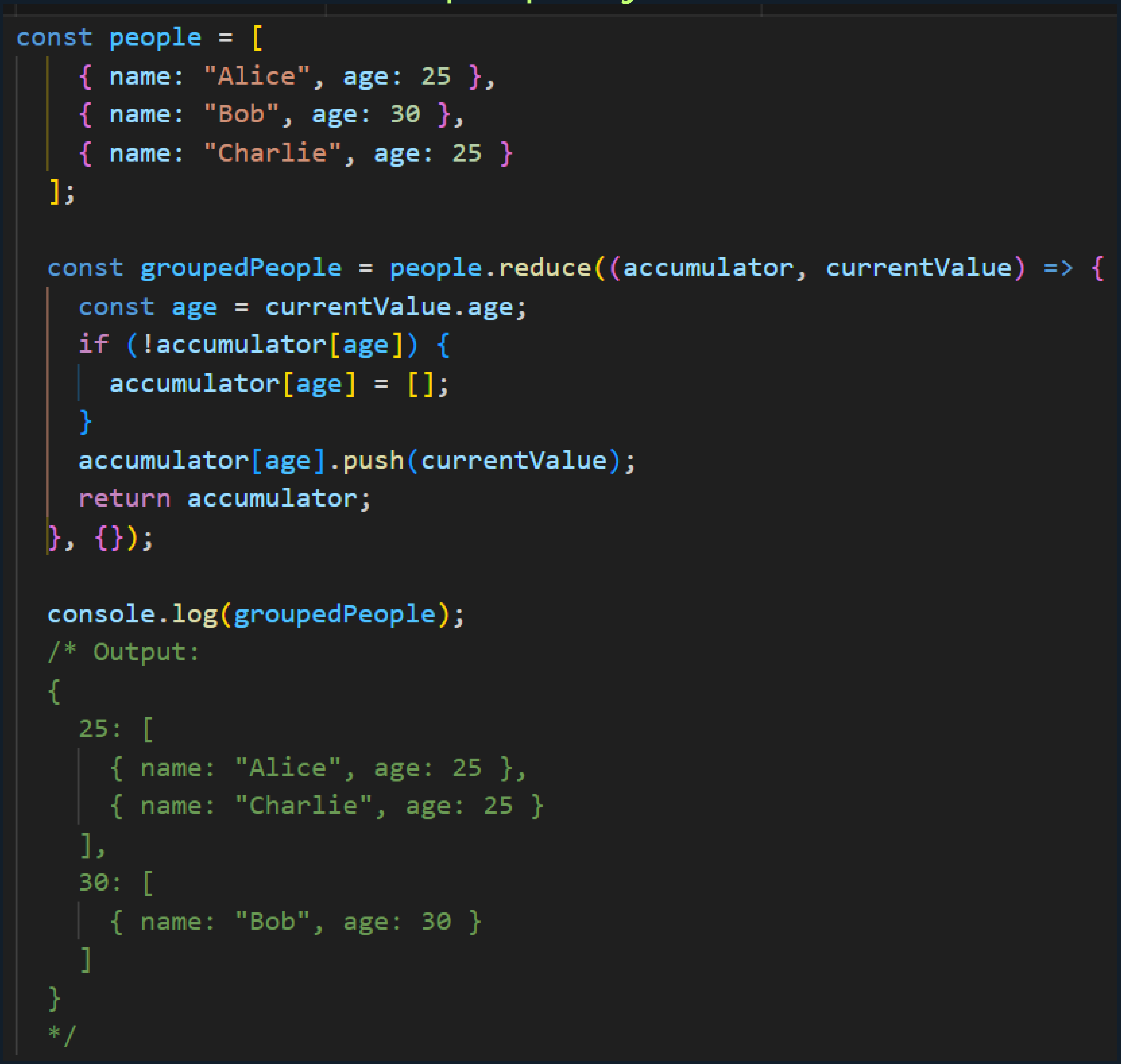
```javascript
const numbers = [1, 2, 3, 4, 5];
const sum = numbers.reduce((accumulator, currentValue) => accumulator + currentValue);
console.log(sum); // Output: 15
```

Use Case 2: Calculating the average of all the values in an array



```javascript
const numbers = [1, 2, 3, 4, 5];
const average = numbers.reduce((accumulator, currentValue, index, array) => {
  accumulator += currentValue;
  if (index === array.length - 1) {
    return accumulator / array.length;
  } else {
    return accumulator;
  }
});
console.log(average); // Output: 3
```

Use Case 3: Grouping objects in an array by a specific property

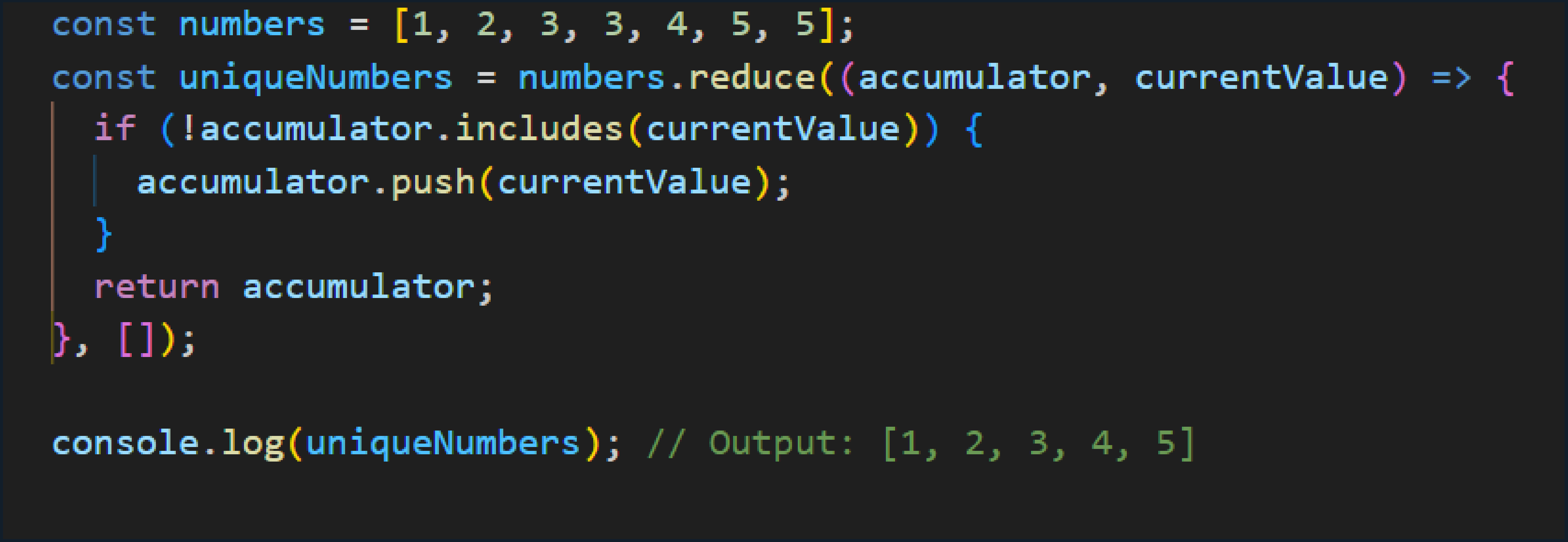


```javascript
const people = [
    { name: "Alice", age: 25 },
    { name: "Bob", age: 30 },
    { name: "Charlie", age: 25 }
  ];

  const groupedPeople = people.reduce((accumulator, currentValue) => {
    const age = currentValue.age;
    if (!accumulator[age]) {
      accumulator[age] = [];
    }
    accumulator[age].push(currentValue);
    return accumulator;
  }, {});

  console.log(groupedPeople);
  /* Output:
  {
    25: [
      { name: "Alice", age: 25 },
      { name: "Charlie", age: 25 }
    ],
    30: [
      { name: "Bob", age: 30 }
    ]
  }
  */
```

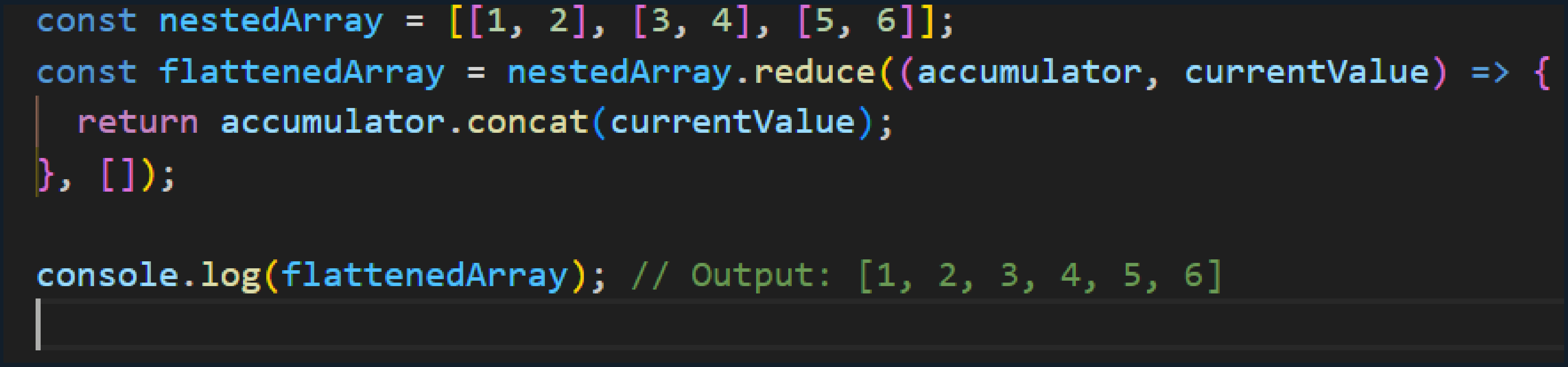
Use Case 4: Removing duplicate values from an array



```
const numbers = [1, 2, 3, 3, 4, 5, 5];
const uniqueNumbers = numbers.reduce((accumulator, currentValue) => {
  if (!accumulator.includes(currentValue)) {
    accumulator.push(currentValue);
  }
  return accumulator;
}, []);

console.log(uniqueNumbers); // Output: [1, 2, 3, 4, 5]
```

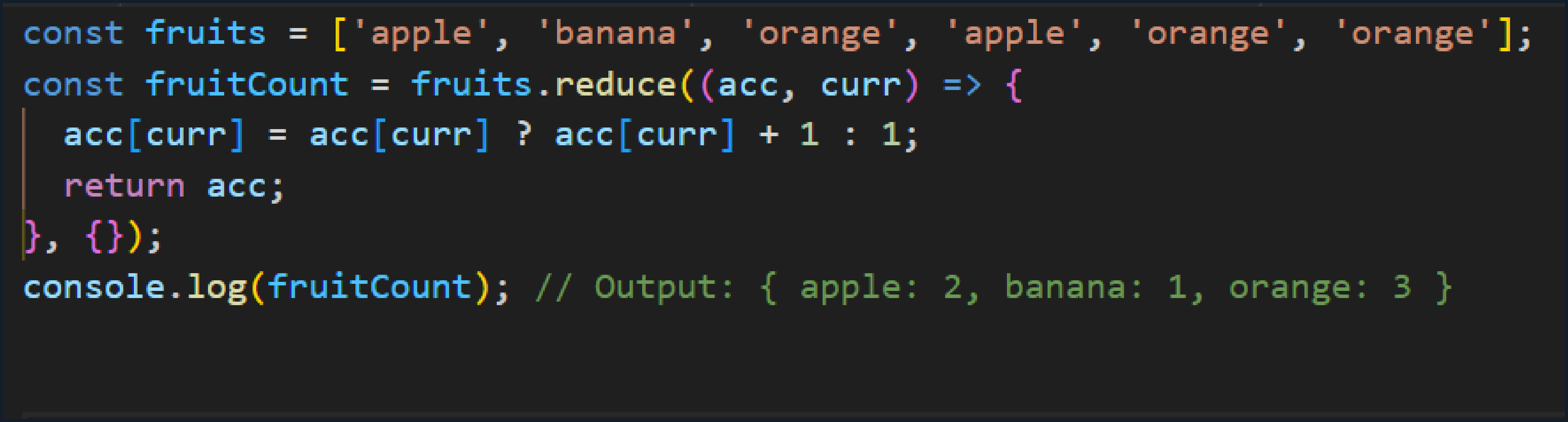
Use Case 5: Flattening an array of nested arrays



```
const nestedArray = [[1, 2], [3, 4], [5, 6]];
const flattenedArray = nestedArray.reduce((accumulator, currentValue) => {
  return accumulator.concat(currentValue);
}, []);

console.log(flattenedArray); // Output: [1, 2, 3, 4, 5, 6]
```

Use Case 6: Counting the occurrences of each element in an array



```
const fruits = ['apple', 'banana', 'orange', 'apple', 'orange', 'orange'];
const fruitCount = fruits.reduce((acc, curr) => {
  acc[curr] = acc[curr] ? acc[curr] + 1 : 1;
  return acc;
}, {});
console.log(fruitCount); // Output: { apple: 2, banana: 1, orange: 3 }
```

AI generated image

Thanks for checking these slides. Consider following me on LinkedIn with below link
Also like and share this