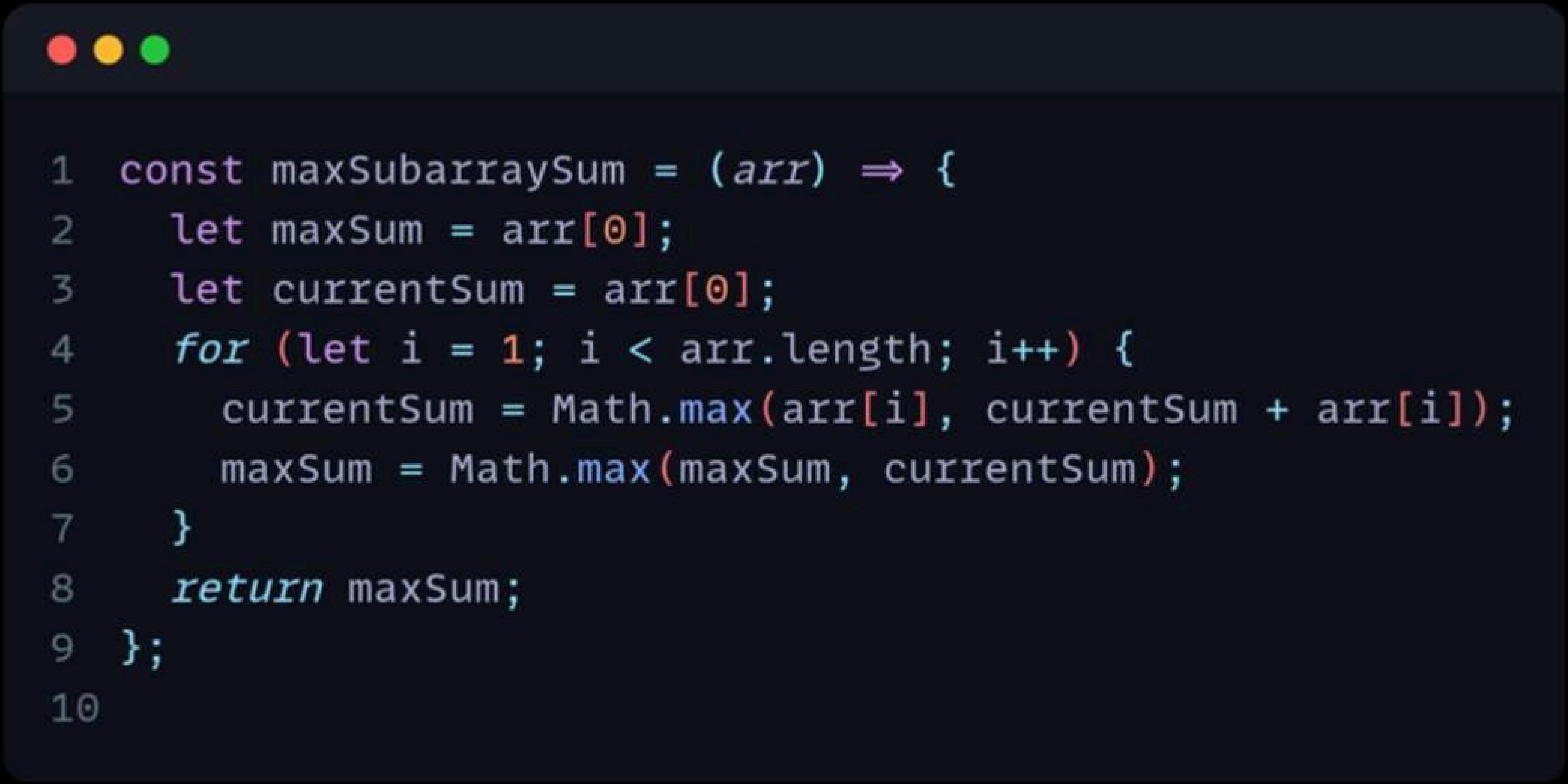
Common Array Coding Problems

[redacted]

FIND MAXIMUM SUBARRAY SUM

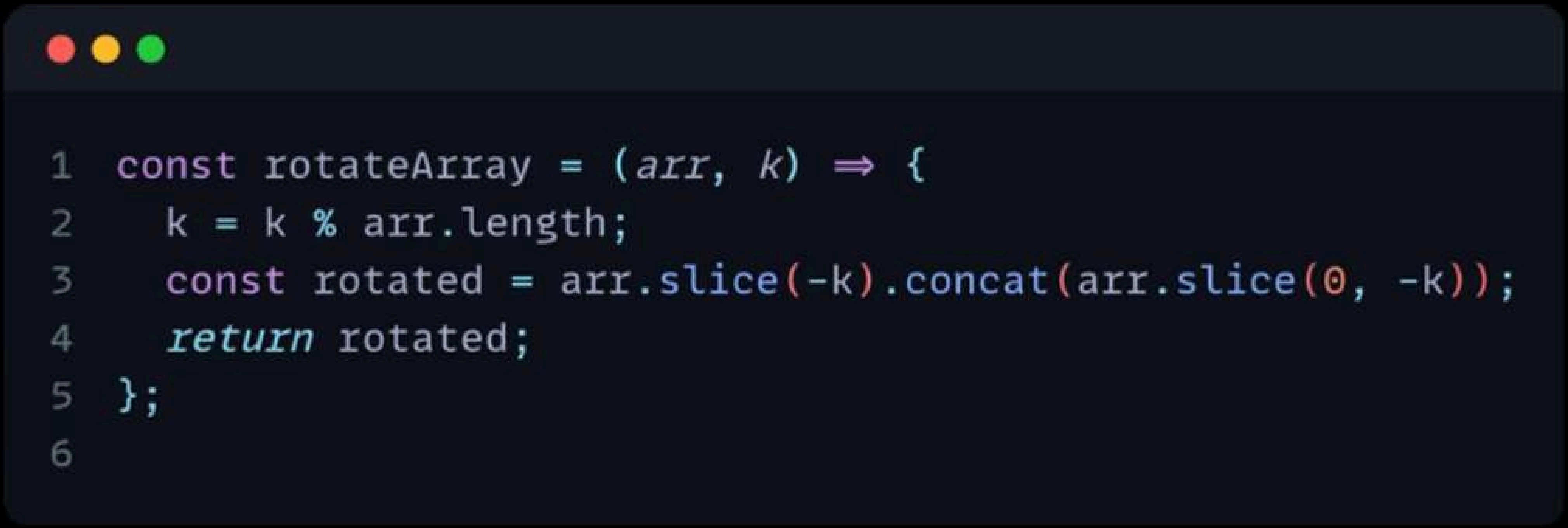


```javascript
const maxSubarraySum = (arr) => {
  let maxSum = arr[0];
  let currentSum = arr[0];
  for (let i = 1; i < arr.length; i++) {
    currentSum = Math.max(arr[i], currentSum + arr[i]);
    maxSum = Math.max(maxSum, currentSum);
  }
  return maxSum;
};
```

NEXT

[redacted]

ROTATE ARRAY



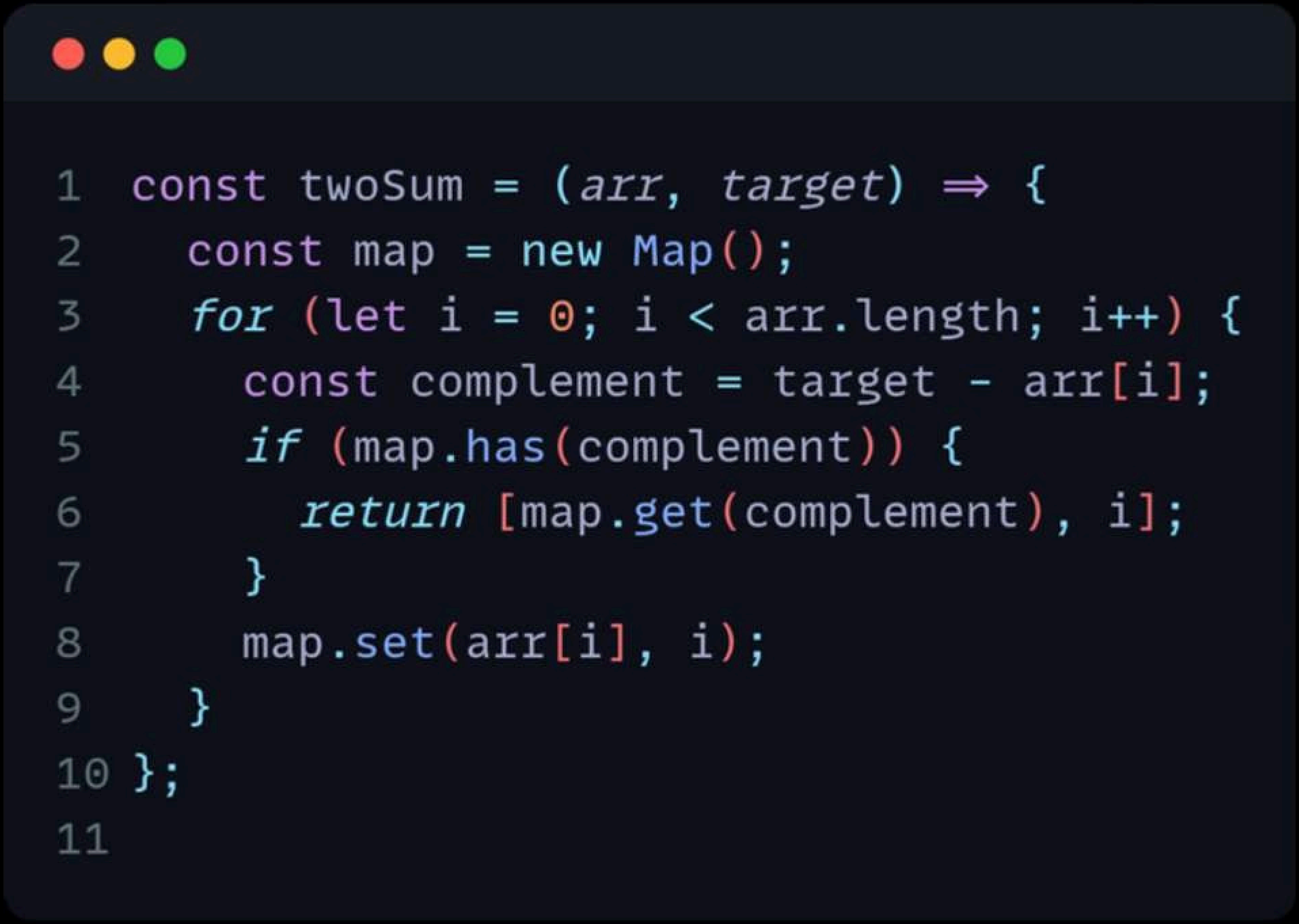
```javascript
const rotateArray = (arr, k) => {
  k = k % arr.length;
  const rotated = arr.slice(-k).concat(arr.slice(0, -k));
  return rotated;
};

```

NEXT

[redacted]

TWO SUM



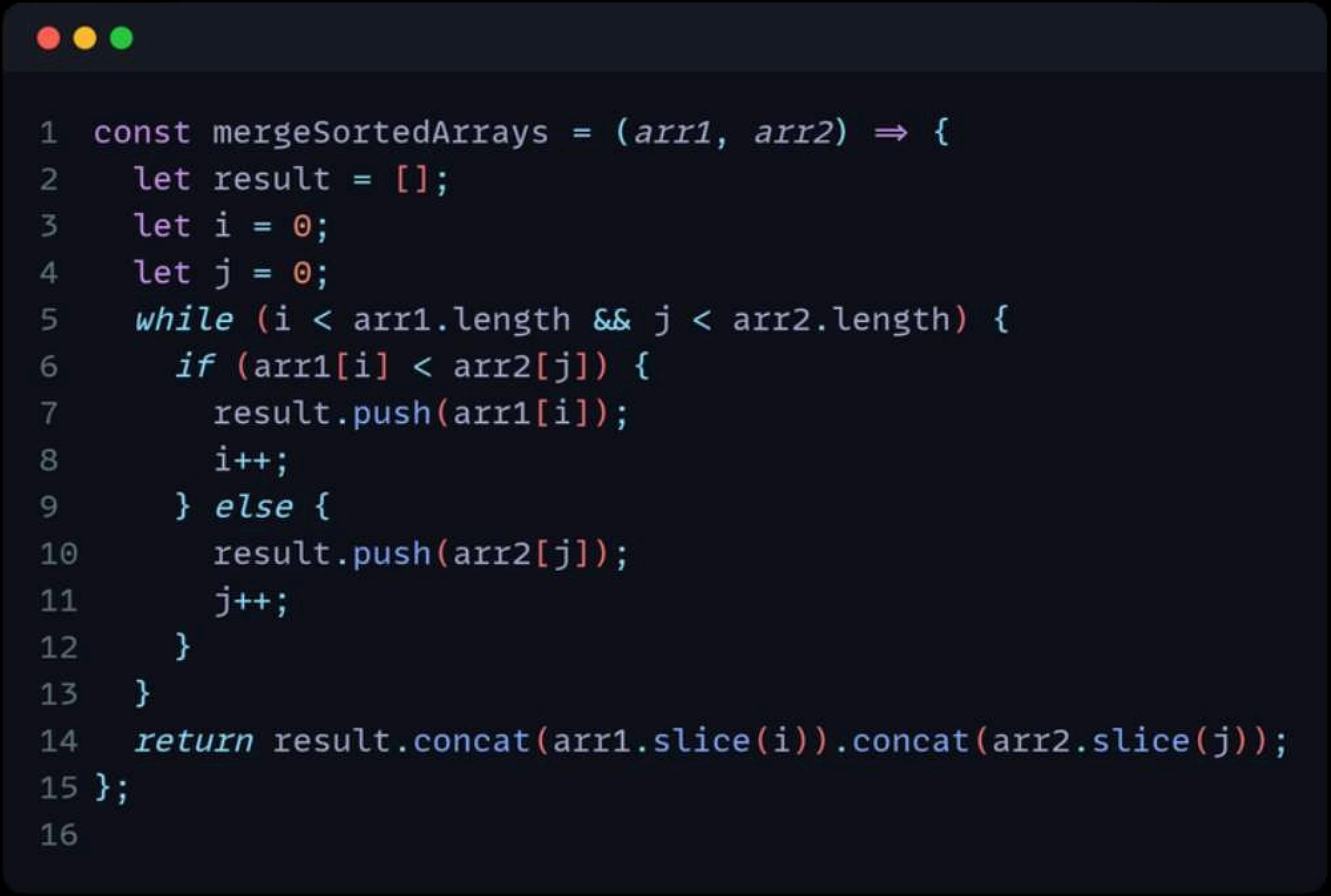
```
const twoSum = (arr, target) => {
  const map = new Map();
  for (let i = 0; i < arr.length; i++) {
    const complement = target - arr[i];
    if (map.has(complement)) {
      return [map.get(complement), i];
    }
    map.set(arr[i], i);
  }
};

```

NEXT

[redacted]

MERGE SORTED ARRAYS



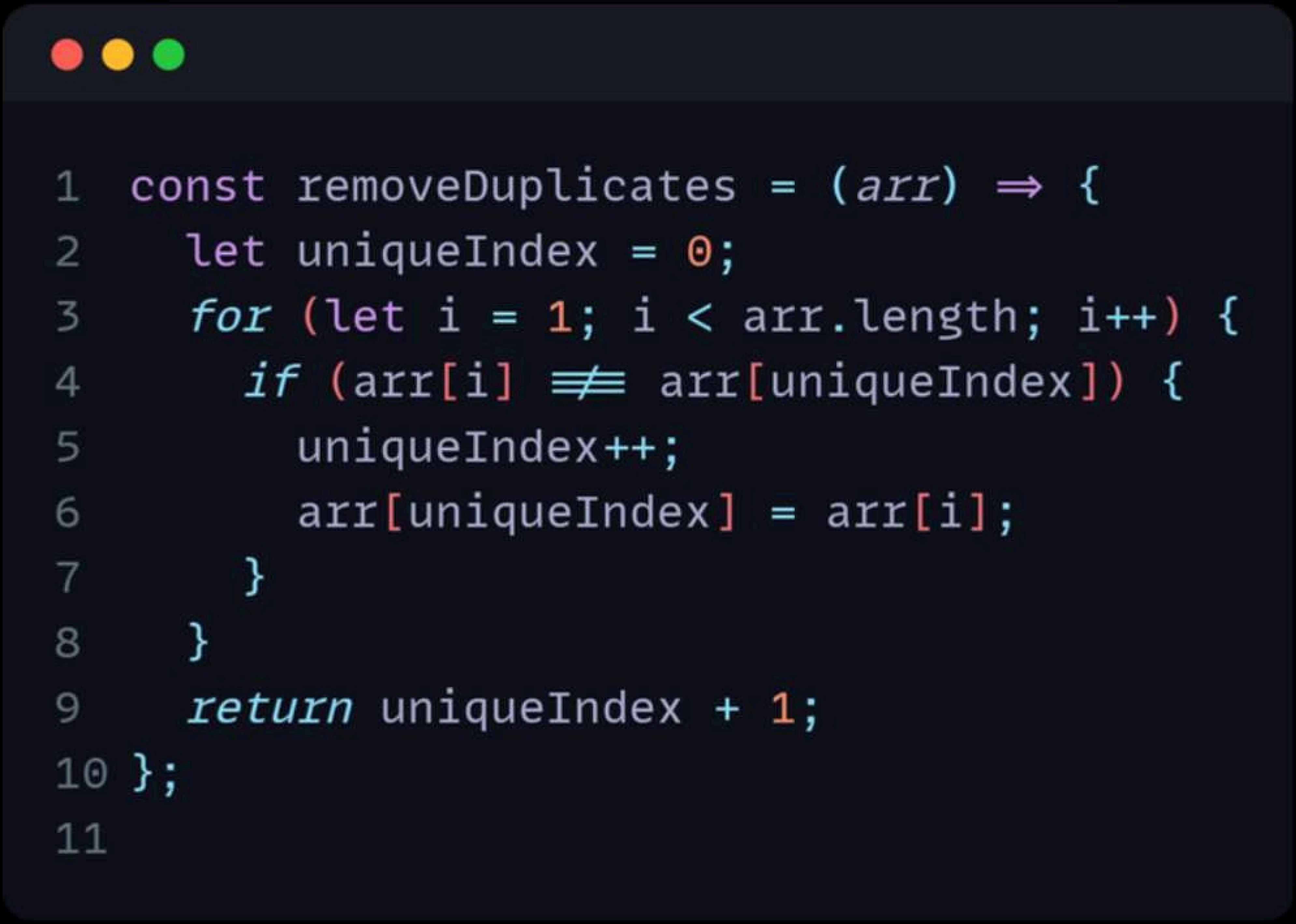
```javascript
const mergeSortedArrays = (arr1, arr2) => {
  let result = [];
  let i = 0;
  let j = 0;
  while (i < arr1.length && j < arr2.length) {
    if (arr1[i] < arr2[j]) {
      result.push(arr1[i]);
      i++;
    } else {
      result.push(arr2[j]);
      j++;
    }
  }
  return result.concat(arr1.slice(i)).concat(arr2.slice(j));
};

```

NEXT

[redacted]

REMOVE DUPLICATES



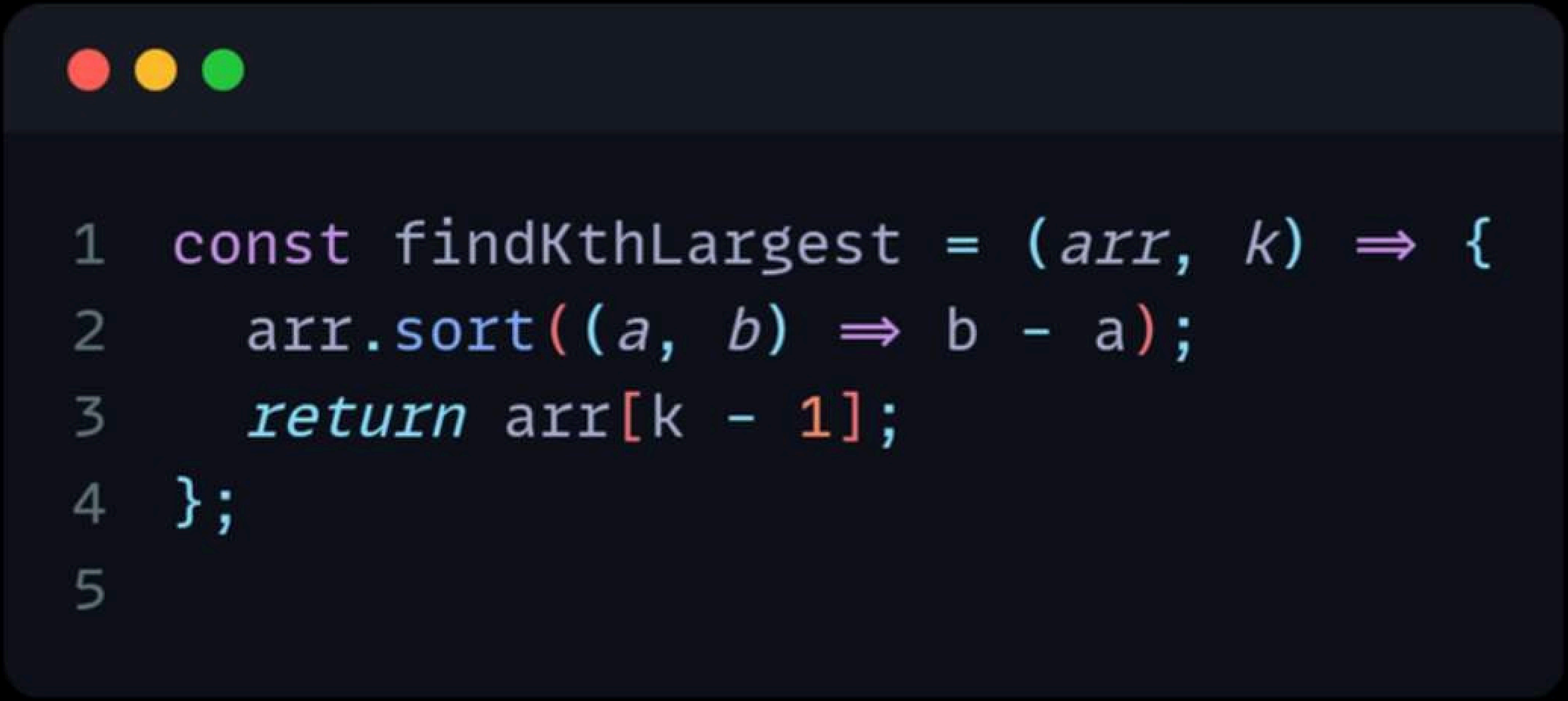
```javascript
const removeDuplicates = (arr) => {
  let uniqueIndex = 0;
  for (let i = 1; i < arr.length; i++) {
    if (arr[i] !== arr[uniqueIndex]) {
      uniqueIndex++;
      arr[uniqueIndex] = arr[i];
    }
  }
  return uniqueIndex + 1;
};

```

NEXT

[redacted]

KTH LARGEST ELEMEN



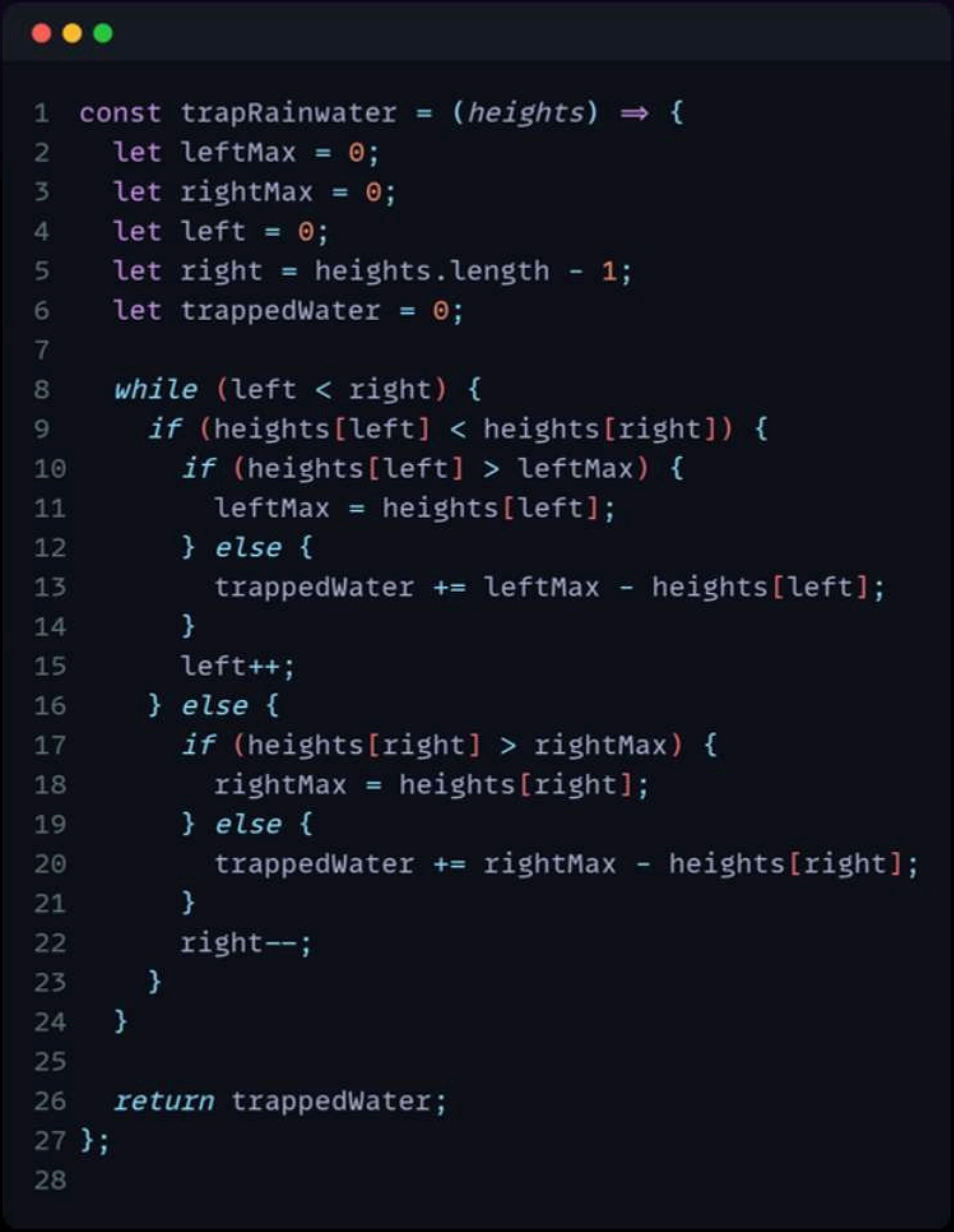
```javascript
const findKthLargest = (arr, k) => {
  arr.sort((a, b) => b - a);
  return arr[k - 1];
};

```

NEXT

[redacted]

TRAPPING RAINWATER



```javascript
const trapRainwater = (heights) ⇒ {
  let leftMax = 0;
  let rightMax = 0;
  let left = 0;
  let right = heights.length - 1;
  let trappedWater = 0;

  while (left < right) {
    if (heights[left] < heights[right]) {
      if (heights[left] > leftMax) {
        leftMax = heights[left];
      } else {
        trappedWater += leftMax - heights[left];
      }
      left++;
    } else {
      if (heights[right] > rightMax) {
        rightMax = heights[right];
      } else {
        trappedWater += rightMax - heights[right];
      }
      right--;
    }
  }

  return trappedWater;
};

```

NEXT

DID YOU FIND THIS POST HELPFUL?

Like, comment, save, and share to help us spread the word!