1. Count Occurrences of Each Item

```
const fruits = ["apple", "banana", "apple", "orange", "banana", "apple"];

function countFruits(arr) {
  return arr.reduce((count, item) => {
    count[item] = (count[item] || 0) + 1;
    return count;
  }, {});
}

console.log(countFruits(fruits));
```

2. Convert an Array to an Object Using a Key

```
const users = [
  { id: 1, name: "Alice" },
  { id: 2, name: "Bob" },
  { id: 3, name: "Charlie" }
];

function arrayToObject(arr, key) {
  return arr.reduce((obj, item) => {
    obj[item[key]] = item;
    return obj;
  }, {});
}

console.log(arrayToObject(users, 'id'));
```

3. Find the Highest Priced Item

```
const products = [
  { name: "Phone", price: 600 },
  { name: "Laptop", price: 1200 },
  { name: "Tablet", price: 800 }
];

function findMaxPrice(products) {
  return products.reduce((max, item) => {
    return item.price > max.price ? item : max;
  });
}

console.log(findMaxPrice(products));
```

4. Remove Duplicates Based on Object Property

```
const items = [
  { id: 1, name: "Pen" },
  { id: 2, name: "Pencil" },
  { id: 1, name: "Pen" }
];

function removeDuplicates(arr) {
  const seen = new Set();
  return arr.filter(item => {
    if (seen.has(item.id)) return false;
    seen.add(item.id);
    return true;
  });
}

console.log(removeDuplicates(items));
```

5. Merge Two Objects Deeply

```
const obj1 = { a: 1, b: { x: 10 } };
const obj2 = { b: { y: 20 }, c: 3 };

function deepMerge(obj1, obj2) {
  const result = { ...obj1 };
  for (let key in obj2) {
    if (obj2.hasOwnProperty(key)) {
       if (typeof obj2[key] === 'object' && obj2[key] !== null && typeof result[key] === 
'object') {
        result[key] = deepMerge(result[key], obj2[key]);
      } else {
        result[key] = obj2[key];
      }
    }
  }
  return result;
}

console.log(deepMerge(obj1, obj2));
```

6. Filter Objects by Value Range

```
const products = [
  { name: "TV", price: 450 },
  { name: "Pen", price: 10 },
```

```
  { name: "Phone", price: 700 },
  { name: "Keyboard", price: 200 }
];

function filterByPrice(products, min, max) {
  return products.filter(p => p.price >= min && p.price <= max);
}

console.log(filterByPrice(products, 50, 500));
```

7. Sum Values by Key

```
const cart = [
  { item: "Shoes", price: 100 },
  { item: "Hat", price: 50 },
  { item: "Bag", price: 150 }
];

function totalCost(items) {
  return items.reduce((sum, item) => sum + item.price, 0);
}

console.log(totalCost(cart));
```

8. Create an Object Grouped by First Letter

```
const people = ["Alice", "Bob", "Charlie", "Anita", "David"];

function groupByFirstLetter(names) {
  return names.reduce((grouped, name) => {
    const letter = name[0];
    if (!grouped[letter]) {
      grouped[letter] = [];
    }
    grouped[letter].push(name);
    return grouped;
  }, {});
}

console.log(groupByFirstLetter(people));
```

9. Flatten a Nested Object

```
const data = {
  user: {
    name: "John",
```

```
    address: {
      city: "Mumbai",
      pin: 400001
    }
  }
};

function flattenObject(obj, parent = "", result = {}) {
  for (let key in obj) {
    const prop = parent ? \`\${parent}.\${key}\` : key;
    if (typeof obj[key] === "object" && obj[key] !== null) {
      flattenObject(obj[key], prop, result);
    } else {
      result[prop] = obj[key];
    }
  }
  return result;
}

console.log(flattenObject(data));
```

10. Find Duplicate Values in Object Array Based on a Key

```
const users = [
  { id: 1, email: "a@example.com" },
  { id: 2, email: "b@example.com" },
  { id: 3, email: "a@example.com" },
  { id: 4, email: "c@example.com" }
];

function findDuplicates(users) {
  const emailCount = {};
  users.forEach(u => {
    emailCount[u.email] = (emailCount[u.email] || 0) + 1;
  });
  return users.filter(u => emailCount[u.email] > 1);
}

console.log(findDuplicates(users));
```