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));
```

11. Convert Object to Array of Keys and Values

```
const obj = { name: "Alice", age: 25 };

const entries = Object.entries(obj);
console.log(entries); // [ ['name', 'Alice'], ['age', 25] ]
```

12. Use Map to Store Object Key-Value Pairs

```
const map = new Map();
map.set("name", "Alice");
map.set("age", 30);

console.log(map.get("name")); // Alice
```

13. Find Missing Keys Between Two Objects

```
const obj1 = { a: 1, b: 2, c: 3 };
const obj2 = { a: 1, c: 3 };

const missingKeys = Object.keys(obj1).filter(key => !(key in obj2));
console.log(missingKeys); // ['b']
```

14. Check Deep Equality of Two Objects

```
function deepEqual(a, b) {
  if (a === b) return true;
  if (typeof a !== "object" || typeof b !== "object" || a === null || b === null) return
false;

  const keysA = Object.keys(a);
  const keysB = Object.keys(b);

  if (keysA.length !== keysB.length) return false;

  return keysA.every(key => deepEqual(a[key], b[key]));
}

console.log(deepEqual({ a: 1 }, { a: 1 })); // true
```

15. Sort Object by Values

```
const scores = { John: 50, Alice: 70, Bob: 60 };

const sorted = Object.entries(scores).sort((a, b) => b[1] - a[1]);
console.log(sorted); // [['Alice', 70], ['Bob', 60], ['John', 50]]
```

16. Convert Array of Key-Value Pairs to Object

```
const entries = [["name", "John"], ["age", 30]];

const obj = Object.fromEntries(entries);
console.log(obj); // { name: 'John', age: 30 }
```

17. Get Nested Value Using a Path

```
const obj = { a: { b: { c: 42 } } };

function getValue(obj, path) {
  return path.split(".").reduce((acc, key) => acc?.[key], obj);
}

console.log(getValue(obj, "a.b.c")); // 42
```

18. Invert Object Keys and Values

```
const obj = { a: 1, b: 2, c: 3 };

const inverted = Object.entries(obj).reduce((acc, [key, val]) => {
  acc[val] = key;
  return acc;
}, {});

console.log(inverted); // { '1': 'a', '2': 'b', '3': 'c' }
```

19. Clone Object Without Reference

```
const original = { a: 1, b: { c: 2 } };
const clone = JSON.parse(JSON.stringify(original));

console.log(clone);
```

20. Sum Nested Object Values

```
const data = {
  Jan: { income: 1000, expense: 500 },
  Feb: { income: 1200, expense: 700 }
};

const total = Object.values(data).reduce((acc, curr) => {
  acc.income += curr.income;
  acc.expense += curr.expense;
  return acc;
}, { income: 0, expense: 0 });

console.log(total); // { income: 2200, expense: 1200 }
```

21. Count Keys in a Nested Object

```
const obj = { a: 1, b: { c: 2, d: { e: 3 } } };
```

```
function countKeys(o) {
  return Object.keys(o).reduce((acc, key) => {
    if (typeof o[key] === "object" && o[key] !== null) {
      return acc + 1 + countKeys(o[key]);
    }
    return acc + 1;
  }, 0);
}

console.log(countKeys(obj)); // 5
```

22. Check if All Values Are Numbers

```
const obj = { a: 1, b: 2, c: 3 };

const allNumbers = Object.values(obj).every(val => typeof val === "number");
console.log(allNumbers); // true
```

23. Transform Object Values

```
const obj = { a: 1, b: 2, c: 3 };

const doubled = Object.fromEntries(
  Object.entries(obj).map(([key, val]) => [key, val * 2])
);

console.log(doubled); // { a: 2, b: 4, c: 6 }
```

24. Remove Keys with Null or Undefined

```
const obj = { a: 1, b: null, c: 3, d: undefined };

const cleaned = Object.fromEntries(
  Object.entries(obj).filter(([_, val]) => val != null)
);

console.log(cleaned); // { a: 1, c: 3 }
```

25. Nest Flat Object by Dot Keys

```
const flat = { "a.b.c": 1, "a.b.d": 2 };

function nest(flat) {
  const result = {};
  for (const [key, value] of Object.entries(flat)) {
    const keys = key.split(".");
    keys.reduce((acc, k, i) => {
      if (i === keys.length - 1) {
        acc[k] = value;
      } else {
        acc[k] = acc[k] || {};
      }
      return acc[k];
    }, result);
  }
  return result;
}

console.log(nest(flat));
```

26. Filter Object Keys by Condition

```
const obj = { a: 1, b: 2, c: 3, d: 4 };

const filtered = Object.fromEntries(
  Object.entries(obj).filter(([key, value]) => value % 2 === 0)
);

console.log(filtered); // { b: 2, d: 4 }
```

27. Merge Objects with Priority to Later Object

```
const a = { x: 1, y: 2 };
const b = { y: 3, z: 4 };

const merged = { ...a, ...b };
console.log(merged); // { x: 1, y: 3, z: 4 }
```

28. Count Frequency of Characters in a String Using Object

```
const str = "hello world";

const freq = {};
for (const char of str) {
  if (char !== " ") freq[char] = (freq[char] || 0) + 1;
}

console.log(freq);
```

29. Group Objects by Category

```
const items = [
  { name: "Apple", category: "Fruit" },
  { name: "Carrot", category: "Vegetable" },
  { name: "Banana", category: "Fruit" }
];

const grouped = items.reduce((acc, item) => {
  const cat = item.category;
  if (!acc[cat]) acc[cat] = [];
  acc[cat].push(item);
  return acc;
}, {});

console.log(grouped);
```

30. Sort Array of Objects by Date

```
const posts = [
  { title: "Post A", date: "2023-06-01" },
  { title: "Post B", date: "2022-12-15" }
];

posts.sort((a, b) => new Date(b.date) - new Date(a.date));
console.log(posts);
```

31. Convert Object to Query String

```
const params = { search: "apple", page: 2 };

const query = Object.entries(params).map(([k, v]) => \`\${k}=\${v}\`).join("&");
console.log(query); // search=apple&page=2
```

32. Convert Query String to Object

```
const query = "search=apple&page=2";

const obj = Object.fromEntries(new URLSearchParams(query));
console.log(obj);
```

33. Freeze Object to Prevent Modification

```
const config = { debug: true };
Object.freeze(config);

config.debug = false;
console.log(config.debug); // true
```

34. Check if Object is Empty

```
const obj = {};
const isEmpty = Object.keys(obj).length === 0;
console.log(isEmpty); // true
```

35. Pick Specific Keys from Object

```
const user = { id: 1, name: "John", age: 25 };

const picked = (({ name, age }) => ({ name, age }))(user);
console.log(picked); // { name: "John", age: 25 }
```

36. Exclude Specific Keys from Object

```
const user = { id: 1, name: "John", age: 25 };

const { id, ...rest } = user;
console.log(rest); // { name: "John", age: 25 }
```

37. Count Number of Properties

```
const obj = { a: 1, b: 2, c: 3 };

console.log(Object.keys(obj).length);
```

38. Check If Two Objects Have Same Keys

```
const a = { x: 1, y: 2 };
const b = { y: 3, x: 4 };

const sameKeys = JSON.stringify(Object.keys(a).sort()) ===
JSON.stringify(Object.keys(b).sort());
console.log(sameKeys);
```

39. Convert Object to JSON and Back

```
const obj = { name: "Alice", age: 30 };

const jsonStr = JSON.stringify(obj);
const newObj = JSON.parse(jsonStr);

console.log(newObj);
```

40. Create Object from Two Arrays

```
const keys = ["a", "b", "c"];
const values = [1, 2, 3];

const obj = Object.fromEntries(keys.map((k, i) => [k, values[i]]));
```

```
console.log(obj);
```

41. Use Reduce to Convert Array of Objects to Map

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

const userMap = users.reduce((map, user) => {
  map[user.id] = user.name;
  return map;
}, {});

console.log(userMap);
```

42. Map Over Object Values

```
const obj = { a: 1, b: 2, c: 3 };

const mapped = Object.fromEntries(
  Object.entries(obj).map(([k, v]) => [k, v * 10])
);

console.log(mapped);
```

43. Destructure Nested Object

```
const obj = { user: { name: "John", age: 30 } };

const { user: { name } } = obj;
console.log(name); // John
```

44. Safe Access with Optional Chaining

```
const obj = { user: { name: "Jane" } };

console.log(obj?.user?.name); // Jane
```

45. Use Set to Remove Duplicate Objects by Key

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

```
const seen = new Set();
const unique = users.filter(u => {
  if (seen.has(u.id)) return false;
  seen.add(u.id);
  return true;
});

console.log(unique);
```

46. Use Object.entries to Log Key-Value Pairs

```
const obj = { x: 10, y: 20 };

for (const [key, value] of Object.entries(obj)) {
  console.log(key, value);
}
```

47. Create Object from Array of Objects Using Specific Key

```
const arr = [
  { id: "a", val: 1 },
  { id: "b", val: 2 }
];

const obj = Object.fromEntries(arr.map(item => [item.id, item]));
console.log(obj);
```

48. Check If Object Has a Certain Key

```
const obj = { a: 1, b: 2 };

console.log("a" in obj); // true
```

49. Deep Clone with StructuredClone

```
const original = { a: 1, b: { c: 2 } };

const clone = structuredClone(original);
console.log(clone);
```

50. Detect Changes Between Two Objects

```
const oldData = { name: "John", age: 25 };
const newData = { name: "John", age: 26 };

const changes = Object.keys(newData).filter(key => oldData[key] !== newData[key]);
console.log(changes); // ['age']
```