```
// 🔄 Clear Console

console.clear();

// 👥 Array of Objects: Users Data

const users = [

    {
        id: 1,
        name: "Ajay Suneja",
        isActive: true,
        age: 31,
        designation: "Front End Developer",
    },

    {
        id: 2,
        name: "Manohar Batra",
        isActive: true,
        age: 36,
        designation: "Solution Architect",
    },

    {
        id: 3,
        name: "Dimple Kumari",
        isActive: true,
        age: 26,
        designation: "Software Engineer",
    },

    {
        id: 4,
        name: "Anshika Gupta",
        isActive: false,
        age: 25,
```

```
        designation: "Back End Developer",
    },
];

/*

================================================

Use Case 1: Check if user with such names exists

================================================

*/

// Method 1 : Using for loop - It is a basic way of iterating over an array of an object.

const isNameExistsUsingForLoop = (name, users) => {

    let userExists = false;

    for (let i = 0; i < users.length; i++) {

        if (users[i].name === name) {

            userExists = true;

            break;
        }
    }

    return userExists;
};

console.log("✅ User with name 'Manohar Batra' exists:", isNameExistsUsingForLoop("Manohar Batra", users));

console.log("❌ User with name 'Akshay Saini' exists:", isNameExistsUsingForLoop("Akshay Saini", users));

```

```
// Method 2: Using find() - It returns the first element that matches the condition. Here we have explicitely used Boolean because find method gives us the whole object.

const isNameExistsUsingFind = (name, users) => {

    const user = users.find((user) => user.name === name);

    return Boolean(user);
};

console.log("✅ User with name 'Dimple Kumari' exists:", isNameExistsUsingFind("Dimple Kumari", users));

console.log("❌ User with name 'Akshay Saini' exists:", isNameExistsUsingFind("Akshay Saini", users));

// Method 3 - Using findIndex() - It returns the index of the first element that matches the condition. Here we have to include a condition because findIndex method gives us the index.

const isNameExistsUsingFindIndex = (name, users) => {

    const index = users.findIndex((user) => user.name === name);

    return index >= 0;
}

console.log("✅ User with name 'Ajay Suneja' exists:", isNameExistsUsingFindIndex("Ajay Suneja", users));

console.log("❌ User with name 'Akshay Saini' exists:", isNameExistsUsingFindIndex("Akshay Saini", users));

// Method 4 - Using some() - It returns true if at least one element matches the condition (doesn't return the whole object).

const isNameExistsUsingSome = (name, users) => {

    const userExists = users.some((user) => user.name === name);

    return userExists;
}

console.log("✅ User with name 'Anshika Gupta' exists:", isNameExistsUsingSome("Anshika Gupta", users));
```

```
111
112 console.log("❌ User with name 'Akshay Saini' exists:", isNameExistsUsingSome("Akshay Saini", users));
113
114 /*
115
116 ==========================================
117
118 Use Case 2: Adding Elements to an Array
119
120 ==========================================
121
122 */
123
124 const arr = [1, 2, 3, 4];
125
126 // Method 1: Using push() - It adds an element to the end of the array. It mutates the original array.
127
128 arr.push(5);
129
130 console.log("✅ Array after adding 5 (using push):", arr);
131
132 // Method 2: Using Spread Operator - It allows us to take all the elements from the original array and add them individually
    to a new array.
133
134 const newArr = [...arr, 6];
135
136 console.log("✅ Array after adding 6 (using spread operator):", newArr);
137
138 console.log("Original Array:", arr);
139
140 /*
141
142 ======================================================
143
144 Use Case 3: Removing Duplicate Elements from an Array
145
146 ======================================================
147
```

```
148 */
149
150 const duplicateElementsArr = [1, 2, 3, 4, 1, 2, 3, 4];
151
152 // Method 1: Using includes() - It returns true if the element is present in the array else false.
153
154 const uniqueElementsArr = () => {
155
156     const uniqueElements = [];
157
158     duplicateElementsArr.forEach((element) => {
159
160         if (!uniqueElements.includes(element)) {
161
162             uniqueElements.push(element);
163         }
164     })
165
166     return uniqueElements;
167 }
168
169 console.log("✅ Array after removing duplicates (using includes()):", uniqueElementsArr());
170
171 // Method 2 - Using Set() - It is a built in data structure that stores only unique values of any type. Set is object so every
    valus will be on index 0. We want it to be in the form of an array so we use the spread operator.
172
173 const uniqueElementsArrUsingSet = () => {
174
175     return [...new Set(duplicateElementsArr)];
176 }
177
178 console.log("✅ Array after removing duplicates (using Set()):", uniqueElementsArrUsingSet());
179
180 // Method 3 - Using Reduce() - It returns a single value after processing all the elements of the array.
181
182 const uniqueElementsArrUsingReduce = () => {
183
184     return duplicateElementsArr.reduce((acc, element) => {
```

```
185
186         return acc.includes(element) ? acc : [...acc, element];
187
188     }, []);
189 }
190
191 console.log("✅ Array after removing duplicates (using Reduce()):", uniqueElementsArrUsingReduce());
192
193 /*
194
195 ====================================
196
197 Use Case 4: Concatenating Arrays
198
199 ====================================
200
201 */
202
203 const arr1 = [1, 2, 3];
204
205 const arr2 = [4, 5, 6];
206
207 // Method 1: Using Spread Operator - It allows us to take all the elements from the original array and add them individually
    to a new array.
208
209 const concatenatedArray = [...arr1, ...arr2];
210
211 console.log("✅ Concatenated Array:", concatenatedArray);
212
213 // Method 2: Using concat() - It concatenates two or more arrays and returns a new array.
214
215 const concatenatedArrayUsingConcat = arr1.concat(arr2);
216
217 console.log("✅ Concatenated Array (using concat()):", concatenatedArrayUsingConcat);
218
219 console.log("Original Arrays:", arr1, arr2);
```