Question 1: Reverse an Array

Problem: Write a function that takes an array and returns a new array with the elements in reverse order.

Input: [1, 2, 3, 4, 5]

Output: [5, 4, 3, 2, 1]

Use Case: This function can be used in a web application where user reviews need to be displayed in reverse chronological order.

Code:

```
//Reverse an Array
//Write a function that takes an array and returns a new array with the elements in reverse order.

function reverseArray(arr) {
return arr.slice().reverse();
    }
    const inputArray = [1, 2, 3, 4, 5];
    const reversedArray = reverseArray(inputArray);
    console.log(reversedArray); // Output: [5, 4, 3, 2, 1]
```

Question 2: Flatten an Array

Problem: Write a function that takes a nested array and flattens it to a single-level array.

Input: [1, [2, 3], [4, [5]]]

Output: [1, 2, 3, 4, 5]

Use Case: Useful for aggregating user-selected items from multiple categories into a single list for checkout.

Code:

```
//Flatten an Array
//Write a function that takes a nested array and flattens it to a single-level array.


function flattenArray(arr) {
    return arr.flat(Infinity);
}
const inputArray = [1, [2, 3], [4, [5]]];
const flattenedArray = flattenArray(inputArray);
console.log(flattenedArray); //Output: [1, 2, 3, 4, 5]
```

Question 3: Check for Duplicates

Problem: Write a function that checks if an array contains duplicates.

Input: [1, 2, 3, 4, 5, 1]

Output: true

Input: [1, 2, 3, 4, 5]

Output: false

Use Case: Can be used to validate user inputs in forms, such as ensuring usernames are unique during registration.

Code:

```
//Check for Duplicate
//Write a function that checks if an array contains duplicates.

function hasDuplicates(arr) {
    const uniqueElements = new Set(arr);
    return uniqueElements.size !== arr.length;
}
console.log(hasDuplicates([1, 2, 3, 4, 5, 1])); // Output: true
console.log(hasDuplicates([1, 2, 3, 4, 5]));    // Output: false
```

Question 4: Merge Two Objects

Problem: Write a function that merges two objects into one.

Input: { a: 1, b: 2 }, { b: 2, c: 4 }

Output: { a: 1, b: 2, c: 4 }

Use Case: This can be used in a web application to combine user profile settings from different sources.

Code:

```
//Merge Two Objects
//Write a function that merges two objects into one.

function mergeObjects(obj1, obj2) {
    return { ...obj1, ...obj2 };
}
const object1 = { a: 1, b: 2 };
const object2 = { b: 2, c: 4 };
const mergedObject = mergeObjects(object1, object2);
console.log(mergedObject); //Output: { a: 1, b: 2, c: 4 }
```

Question 5: Find the Maximum Number in an Array

Problem: Write a function that finds the maximum number in an array.

Input: [1, 3, 2, 8, 5]

Output: 8

Use Case: This function can help in analytics dashboards to find the highest sales figure or user activity.

Code:

```
//Find the Maximum Number in an Array
//Write a function that finds the maximum number in an array.

function findMaxNumber(arr) {
    return Math.max(...arr);
}
const numbers = [1, 3, 2, 8, 5];
const maxNumber = findMaxNumber(numbers);
console.log(maxNumber); //Output: 8
```

Question 6: Group Array of Objects by Property

Problem: Write a function that groups an array of objects by a specific property.

Input: [{ id: 1, category: 'fruit' }, { id: 2, category: 'vegetable' }, { id: 3, category: 'fruit' }]

Output: {

fruit: [{ id: 1, category: 'fruit' }, { id: 3, category: 'fruit' }],

vegetable: [{ id: 2, category: 'vegetable' }]

}

Use Case: Useful for organizing products by category in an e-commerce application.

Code:

```
//Group Array of Objects by Property
//Write a function that groups an array of objects by a specific property.

function groupBy(arr, property) {
  return arr.reduce((acc, obj) => {
      const key = obj[property];
      if (!acc[key]) {
          acc[key] = [];
      }
      acc[key].push(obj);
      return acc;
  }, {});
}
const items = [
  { id: 1, category: 'fruit' },
  { id: 2, category: 'vegetable' },
  { id: 3, category: 'fruit' }
];
const groupedItems = groupBy(items, 'category');
console.log(groupedItems);

  // Output:
  // {
  //   fruit: [
  //     { id: 1, category: 'fruit' },
  //     { id: 3, category: 'fruit' }
  //   ],
  //   vegetable: [
  //     { id: 2, category: 'vegetable' }
  //   ]
  // }
```

Question 7: Find the Intersection of Two Arrays

Problem: Write a function that returns the intersection of two arrays.

Input: [1, 2, 3], [2, 3, 4]

Output: [2, 3]

Use Case: This can be used in social media applications to find mutual friends between users.

Code:

```
//Find the Intersection of Two Arrays
//Write a function that returns the intersection of two arrays.


function getIntersection(arr1, arr2) {
    return arr1.filter(element => arr2.includes(element));
}
const array1 = [1, 2, 3];
const array2 = [2, 3, 4];
const intersection = getIntersection(array1, array2);
console.log(intersection); //Output: [2, 3]
```

Question 8: Calculate the Sum of Array Elements

Problem: Write a function that calculates the sum of all numbers in an array.

Input: [1, 2, 3, 4, 5]

Output: 15

Use Case: Useful in financial applications to calculate the total expenses or revenue.

Code:

```
//Calculate the Sum of Array Elements
//Write a function that calculates the sum of all numbers in an array.


function sumArray(arr) {
    return arr.reduce((acc, num) => acc + num, 0);
}
const numbers = [1, 2, 3, 4, 5];
const sum = sumArray(numbers);
console.log(sum); //Output: 15
```

Question 9: Remove Falsy Values from an Array

Problem: Write a function that removes all falsy values from an array.

Input: [0, 1, false, 2, '', 3]

Output: [1, 2, 3]

Use Case: This function can be used to clean up user inputs or configuration arrays.

Code:

```
//Remove Falsy Values from an Array

//Write a function that removes all falsy values from an array

function removeFalsyValues(arr) {
    return arr.filter(Boolean);
}
const inputArray = [0, 1, false, 2, '', 3];
const cleanedArray = removeFalsyValues(inputArray);
console.log(cleanedArray); //Output: [1, 2, 3]
```

Question 10: Calculate Average of an Array

Problem: Write a function that calculates the average of all numbers in an array.

Input: [1, 2, 3, 4, 5]

Output: 3

Use Case: This function is useful in educational applications where you need to compute the average score of students from an array of their grades.

Code:

```
//Calculate Average of an Array
//Write a function that calculates the average of all numbers in an array.


function calculateAverage(arr) {
    const sum = arr.reduce((acc, num) => acc + num, 0);
    return sum / arr.length;
}
const numbers = [1, 2, 3, 4, 5];
const average = calculateAverage(numbers);
console.log(average); // Output: 3
```