Assignment: Operators, Objects, and Arrays in JavaScript

Instructions:

- Work in a single file (e.g., `assignment.js`).
- Include clear comments and separate each part.
- Do not use functions (apart from what's built into JavaScript) since functions haven't been covered yet.
- Run and test your code with Node.js (e.g., run `node assignment.js`).

Part 1: Mastering Operators

1.1 Arithmetic, Assignment, and Comparison Operators

1. **Basic Calculations:**
    - Create variables `a = 15` and `b = 7`.
    - Compute the sum, difference, product, quotient (round to two decimals), and remainder.
    - Log each result with descriptive messages.
      Example Output:

```
Sum of 15 and 7 is 22.
Difference is 8.
Product is 105.
Quotient is 2.14.
Remainder is 1.
```

2.
3. **Compound Operators:**
    - Declare a variable `x = 10`.
    - Increase `x` by 5 using `+=` and log the result.
    - Multiply `x` by 2 using `*=` and log the result.
    - Decrease `x` by 3 using `-=` and log the final value.
4. **Comparison and Logical Operators:**
    - Declare `p = 10` and `q = 20`.

- Log the results of these comparisons:
  - `p < q`
  - `p === q`
  - `p !== q`
- Combine comparisons with logical operators (e.g., `p < q && q > 15`) and log the boolean outcome.

Part 2: Exploring and Manipulating Objects

2.1 Creating and Modifying Objects

1. **Student Profile Object:**
   Create an object `student` with the following properties:
   - `name` (string)
   - `age` (number)
   - `major` (string)
   - `grades` (an array of numbers, e.g., [88, 92, 76])
   - `isEnrolled` (boolean)
   - **Include two methods:**
     - `introduce`: Returns a string such as "Hi, my name is [name] and I study [major]."
     - `averageGrade`: Calculates and returns the average grade from the `grades` array (use basic arithmetic and a loop).
2. **Access and Update:**
   - Use dot notation to log the student's name.
   - Use bracket notation to log the major.
   - Update the student's age and add a new property `year` (e.g., "Sophomore").
   - Log the entire updated object.

2.2 Object Methods and Advanced Features

1. **Iteration over Object Properties:**
   - Write a `for...in` loop to iterate over all keys of `student` (using `hasOwnProperty`) and log the key and its value.
2. **Destructuring and Merging:**
   - Use object destructuring to extract `name` and `major` into separate variables.
   - Create a separate object `contact` with properties:
     - `email`
     - `phone`

- Merge `contact` into `student` using the spread operator and log the merged object.

Part 3: Working Deeply with Arrays

3.1 Basic Array Operations

1. **Color Array:**
   - Declare an array `colors` with at least five color strings.
   - Log the first and last elements.
   - Replace the second element with a different color.
   - Add a color to the end using `push()` and one to the beginning using `unshift()`.
   - Remove the first and last elements using `shift()` and `pop()`, then log the removed elements and the final array.
2. **Iteration:**
   - Use a `for` loop to log every color with its index.

3.2 Array Methods: splice(), slice(), and forEach()

1. **Using splice():**
   - From your `colors` array, remove two elements starting at index 1.
   - Then, insert two new color names at the same position.
   - Log the modified array and explain what changed.
2. **Using slice():**
   - Create a new array `subsetColors` by extracting elements from index 1 to 3 (not including index 3).
   - Log `subsetColors` and confirm the original array remains unchanged.
3. **Using forEach():**
   - Iterate over the array using `forEach()` and log each element along with its index in a formatted string.

Part 4: Advanced Array Methods – Focusing on map()

4.1 Understanding map()

1. **Definition and Syntax:**

    - Write a comment explaining that `map()` returns a new array by applying a function to every element, and that it does not modify the original array.
2. **Example – Doubling Numbers:**
    - Given `const numbers = [1, 2, 3, 4, 5]`, use `map()` to create a new array `doubled` where each element is doubled.
    - Log the original and new array.
3. **Example – Extracting Data from Objects:**
    - Use the `student` object from Part 2. Suppose the student object now has an array property `grades`.
    - Use `map()` to create a new array of letter grades based on numeric scores (e.g., score >= 90 => "A", score >= 80 => "B", otherwise "C").
    - Log the new array.
4. **Example – Data Transformation with Real-world Scenario:**
    - Create an array `temperaturesC = [0, 15, 25, 30]`.
    - Use `map()` to convert each Celsius temperature to Fahrenheit (using the formula F = C × 9/5 + 32).
    - Log the original Celsius and new Fahrenheit arrays.

4.2 Chaining map() with Other Methods

1. **Combine map() and filter():**
    - Use an array of numbers, e.g., `[5, 10, 15, 20, 25]`.
    - First, use `map()` to multiply each number by 2, then use `filter()` to keep only numbers greater than 30.
    - Log the final result and explain how chaining methods creates a pipeline for data transformation.
2. **Combine map() with reduce():**
    - Given an array of prices `[19.99, 9.99, 4.99, 29.99]`, use `map()` to add a sales tax of 10% to each price, then use `reduce()` to calculate the total cost of all items.
    - Log the final total.

Part 5: Integrating Objects and Arrays

5.1 Managing a Product Inventory

1. **Inventory Array:**
    - Create an array `products` where each element is an object representing a product with the properties:
        - `name` (string)

        - `price` (number)
        - `quantity` (number)

Example:

```
const products = [
  { name: 'Laptop', price: 1200, quantity: 5 },
  { name: 'Smartphone', price: 800, quantity: 10 },
  { name: 'Tablet', price: 450, quantity: 7 }
];
```

        -

2. **Calculating Inventory Value:**
    - Use a loop to compute the total inventory value for each product (price × quantity) and log each value.
    - Then, use `reduce()` on the array (or a loop) to sum these values and log the overall inventory value.
3. **Updating Inventory:**
    - Assume a sale occurred: reduce the quantity of “Smartphone” by 2.
    - Log the updated `products` array.
4. **Using map() for Price Adjustments:**
    - Use `map()` to create a new array `discountedProducts` where each product’s price is reduced by 10%.
    - Log `discountedProducts` and explain how the original array remains unchanged.