Estimated Time: 6 hours
Tools: Use CodeSandbox for implementation and testing.

Assignment Instructions

- Create a new project in CodeSandbox.
- Complete all sections below, ensuring to write clean, modular, and commented code.
- Use TypeScript for TypeScript-specific tasks and JavaScript for other tasks.
- After completing each section, test your code thoroughly.
- Submit your sandbox link with all the solutions organized into separate files or folders.

Section 1: JavaScript Core Concepts

1. Event Loop and Async Programming

Create a JavaScript file that demonstrates the following:

1. Write a function that fetches random jokes from the https://official-joke-api.appspot.com/random_ten API using fetch.
   a. Display the jokes in the console.
   b. Log "Start fetching" before the request and "Fetching complete" after the jokes are displayed.
2. Use setTimeout to add a delay of 2 seconds before displaying a message: "All jokes displayed."
3. Use Promise.all to fetch jokes from the API twice in parallel and display all jokes together.

---

2. DOM Manipulation and Event Handling

1. Create a simple webpage with:
   a. An input field for the user to enter their name.
   b. A button labeled "Greet Me".
   c. A <div> to display a personalized greeting.
2. Write JavaScript to:
   a. Display "Hello [Name]!" inside the <div> when the button is clicked.
   b. Add an event listener that logs "Button clicked!" in the console whenever the button is clicked.

---

3. Array and Object Methods

1. Create an array of objects representing books, with each object containing title, author, and price.
2. Write functions to:
    a. Filter books priced above 500.
    b. Sort books alphabetically by title.
    c. Map the array to a new array containing only titles.
3. Log the results for each operation.

Section 2: ES6 Concepts

4. Classes and Inheritance

1. Create a Person class with the following:
    a. Properties: name, age.
    b. A method introduce that logs “Hi, I’m [name], and I’m [age] years old.”
2. Extend the Person class to create an Employee class with additional properties: position and salary.
    a. Add a method work that logs “I am working as a [position].”
3. Instantiate objects of both classes and call their respective methods.

---

5. Destructuring and Spread Operators

1. Create an object user with properties: name, email, age, and address.
2. Write functions to:
    a. Extract name and email using object destructuring and log them.
    b. Merge user with another object containing additional properties like phone and isAdmin.
    c. Use the spread operator to create a new array combining two arrays of user data.

Section 3: TypeScript Basics

6. TypeScript Type Definitions

1. Define a TypeScript interface Product with properties:
    - id (number)
    - name (string)
    - price (number)
    - category (optional, string).
2. Create a function printProductDetails that accepts an array of Product and logs details for each product.
3. Use the Product interface to create an array of products and pass it to the function.

---

7. Generics in TypeScript

1. Create a generic function reverseArray that:
    - Accepts an array of any type.
    - Returns the reversed array.
2. Use the function with arrays of numbers, strings, and custom objects.

---

8. Enum and Type Aliases

1. Create an enum UserRole with values: Admin, Editor, Viewer.
2. Define a type alias User with properties:
    - id (number)
    - name (string)
    - role (UserRole)
3. 3. Write a function checkAccess that:
    - Accepts a User.
    - Logs whether the user has admin access based on their role.

Section 4: Advanced Concepts

9. Promises and Async/Await

1. Create a function fetchUserData that:
    - Fetches user data from https://jsonplaceholder.typicode.com/users.
    - Uses async/await for fetching and logs the response.
2. Add error handling to log an error message if the fetch fails.

---

10. Modules and Import/Export

1. Split your code into modules:
   - A math.js file exporting functions for addition, subtraction, multiplication, and division
   - A main.js file that imports and uses these functions.
2. Test these modules in CodeSandbox.

Submission Guidelines

- Organize your files and solutions clearly.
- Ensure that your CodeSandbox project includes all solutions.
- Test your code and fix any bugs before submission.
- Share the sandbox link for review.