Exercise: Fetching Data with Promises

Objective:

Write TypeScript code to fetch data from multiple sources concurrently and demonstrate the use of various Promise methods.

Scenario:

You are building a web application that fetches data from different APIs. Each API request is represented as a Promise. Your goal is to fetch data from these sources and use the provided Promise methods to handle the results.

Instructions:

1. Create a TypeScript file, e.g., fetchData.ts
2. Define three functions, each representing an API request and returning a Promise that resolves with some data. You can simulate these functions as follows:

Typescript code:

```
typescript:
// Function to fetch data from Source A
function fetchFromSourceA(): Promise<string> {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('Data from Source A');
    }, 2000);
  });
}

// Function to fetch data from Source B
function fetchFromSourceB(): Promise<string> {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('Data from Source B');
    }, 1500);
  });
}

// Function to fetch data from Source C
function fetchFromSourceC(): Promise<string> {
  return new Promise((resolve) => {
```

```
    setTimeout(() => {
      resolve('Data from Source C');
    }, 1000);
  });
}
```

3. Use the provided functions to fetch data from three different sources *concurrently*.
4. Implement the following tasks using Promise methods: a. Use **Promise.all()** to wait for all three requests to complete and log the results when all Promises are fulfilled. b. Use **Promise.allSettled()** to wait for all three requests to complete and log the status of each Promise, including both fulfilled and rejected Promises. c. Use **Promise.any()** to resolve with the result of the first successful request and log that result. d. Use **Promise.race()** to resolve with the result of the fastest request and log that result.
5. Run the TypeScript code and observe the output to verify that the different Promise methods work as expected.

Note:

You can adjust the simulated delays and data in the **fetchFromSourceX** functions as needed. You can directly call those **fetchFromSourceX** methods or use array like objects whatever you are comfortable to utilize the promises.