**Real World Practice Coding Question for memoization in JS:**

In a weather application, you need to implement a function that retrieves the temperature for a given city from an external API. However, retrieving the temperature can be a slow and resource-intensive operation, so you want to cache the results for each city to avoid making unnecessary API calls. Write a function **getTemperatureForCity** that takes a city name as a parameter and returns the temperature for that city. The function should use memoization to cache the results.

Assume that you have a hardcoded list of temperatures for various cities in place of API. You can use the following data:

const temperatureData = {

'New York': 20,

'London': 18,

'Paris': 22,

'Tokyo': 25,

'Sydney': 28,

};

**Example usage:**

const temperature1 = getTemperatureForCity('New York');

console.log(temperature1); // 20

const temperature2 = getTemperatureForCity('New York');

console.log(temperature2); // 20 (retrieved from cache)

const temperature3 = getTemperatureForCity('London');

console.log(temperature3); // 18

const temperature4 = getTemperatureForCity('London');

console.log(temperature4); // 18 (retrieved from cache)

**Hint:**

* You can use an object as a cache to store the results of previous API calls.
* The keys of the cache object can be the city names, and the values can be the corresponding temperature values.
* You can use a closure to create a private cache object that is not accessible from outside the function.

**JavaScript Best Practices:**

* Use descriptive function and variable names.
* Use meaningful comments to explain your code.
* Avoid using global variables whenever possible.
* Use the const keyword to declare variables that won't be reassigned.
* Follow the principle of least privilege by making variables and functions as private as possible.