Gmail Bikash Adhikari <virtual.bikash@gmail.com>

# Task 2.3: Memoization FunctionWrite a function mem...

1 message

**Bikash** <bikash.caim@gmail.com> Sat, Mar 8, 2025 at 6:54 PM
To: virtual.bikash@gmail.com

JavaScript

### Task 3: Memoization Function

Write a function `memoize(fn)` that returns a memoized version of fn. The memoized function should cache the results of function calls, and return the cached result if the same inputs are provided again.

```
function memoize(fn) {
  const cache = {};

  return function(...args) {
    const key = JSON.stringify(args); // Create a unique key for the arguments

    if (cache[key] !== undefined) {
      return cache[key]; // Return cached result if available
    } else {
      const result = fn(...args); // Call the original function
      cache[key] = result; // Store the result in the cache
      return result;
    }
  };
}

// Example usage:
function expensiveCalculation(a, b) {
  console.log("Calculating..."); // To see when the function is actually called
  return a + b;
}

const memoizedCalculation = memoize(expensiveCalculation);

console.log(memoizedCalculation(2, 3)); // Output: Calculating... 5
console.log(memoizedCalculation(2, 3)); // Output: 5 (cached)
console.log(memoizedCalculation(4, 5)); // Output: Calculating... 9
console.log(memoizedCalculation(4, 5)); // Output: 9 (cached)

function complexObjectCalculation(obj) {
    console.log("Calculating with object");
    return obj.a + obj.b;
}

const memoizedComplexObject = memoize(complexObjectCalculation);

console.log(memoizedComplexObject({a:1, b:2})); //calculating with object, 3
```

```
console.log(memoizedComplexObject({a:1, b:2})); //3 (cached)
console.log(memoizedComplexObject({a:2, b:2})); //calculating with object, 4
```

**Explanation:**

1. `memoize(fn)` **function:**
   - Creates an empty object `cache` to store the results of function calls.
   - Returns a new anonymous function (the memoized function).
2. **Returned (memoized) function:**
   - Creates a unique `key` for the function arguments using `JSON.stringify(args)`. This allows the function to handle various data types as arguments, including objects and arrays.
   - Checks if the `key` exists in the `cache`.
     - If it exists, it means the function has been called with the same arguments before, so it returns the cached result from `cache[key]`.
     - If it doesn't exist, it calls the original function `fn` with the provided arguments.
     - It stores the result of `fn` in the `cache` using the `key`.
     - Then it returns the result.
3. **Caching:**
   - The `cache` object acts as a lookup table, storing the results of function calls based on their arguments.
   - By checking the `cache` before calling the original function, the memoized function avoids redundant calculations, improving performance for expensive or frequently called functions.
   - The use of JSON.stringify allows for the caching of object and array arguments, which would not be possible with simple equality checks.