Day 3 — Functions, Scope & Closures

1. Function vs Arrow Functions

• Traditional functions have hoisting, their own "this", and an "arguments" object.

• Arrow functions do not hoist and use lexical "this" inherited from where they were defined.

• Arrow functions are ideal for callbacks and small utilities.

2. Lexical Scope

Lexical scope defines which variables are accessible based on where code is written. Inner functions can access outer variables, but not the other way around.

3. Closures

A closure is a function that remembers variables from its outer environment even after the outer function finishes. This allows creation of private state and advanced functional patterns.

4. Closure Based Helpers

counter(start): Creates a private counter with increment, decrement, reset, and get methods.
once(fn): Runs a function only once and reuses the stored result on later calls.
memorize(fn): Caches expensive function results and returns cached values for repeated inputs.
Summary
Functions define behavior. Scope defines access. Closures combine both, allowing functions to remember their environment and enabling powerful patterns like counters, memorization, and single run functions.