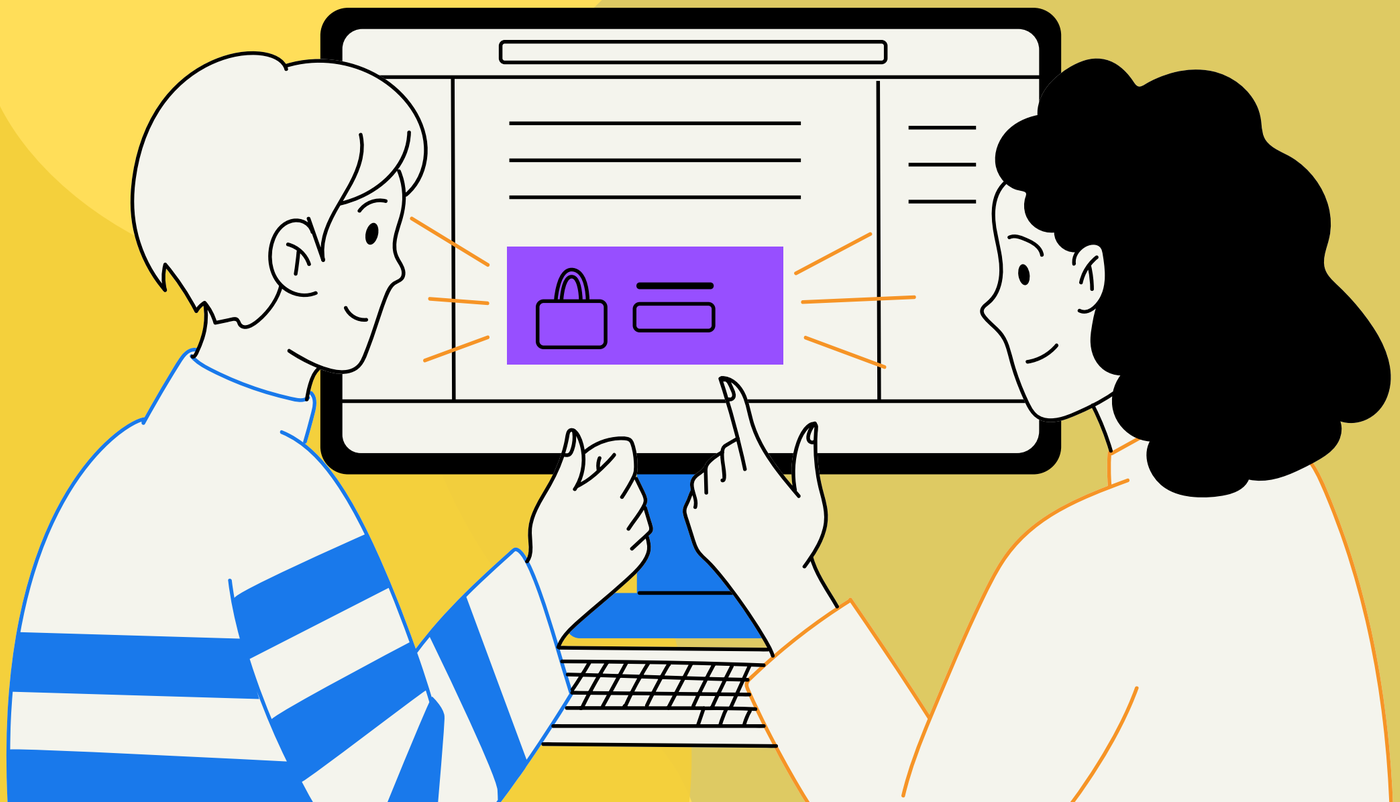
CONCURRENCY

GO



WHAT'S CONCURRENCY?

CONCURRENCY IS THE ABILITY TO HANDLE MULTIPLE TASKS IN OVERLAPPING TIME PERIODS, THOUGH NOT NECESSARILY SIMULTANEOUSLY.

IN GO, IT'S THE COMPOSITION OF INDEPENDENTLY EXECUTING PROCESSES.

GO'S CONCURRENCY MODEL VS. TRADITIONAL THREADING

Key Differences Summarized:

Feature	Go's Concurrency (Goroutines/Channels)	Traditional Threading
Lightweightness	Lightweight, managed by Go runtime	Heavyweight, OS-managed
Resource Usage	Low memory overhead	Higher memory overhead
Context Switching	Efficient, managed by Go runtime	Potentially more expensive, OS-level
Communication	Channels for safe and efficient data exchange	Shared memory, potentially leading to race conditions
Complexity	Simpler concurrency model, easier to reason about	Can be more complex, requires careful synchronization

GOROUTINE

A GOROUTINE IS A LIGHTWEIGHT THREAD OF EXECUTION MANAGED BY THE GO RUNTIME.

IT RUNS CONCURRENTLY WITH OTHER GOROUTINES IN THE SAME ADDRESS SPACE.

CHANNEL

A CHANNEL IS A TYPED CONDUIT THROUGH WHICH YOU CAN SEND AND RECEIVE VALUES BETWEEN GOROUTINES.

CHANNELS PROVIDE SYNCHRONIZATION AND COMMUNICATION.

BUFFERED VS UNBUFFERED CHANNELS

Aspect	Unbuffered Channel	Buffered Channel
Blocking Behavior	Sender and receiver block until both are ready to communicate.	Sender blocks if the buffer is full; receiver blocks if the buffer is empty.
Buffer Size	No buffer (size = 0).	Can hold multiple values (buffer size > 0).
Use Case	For strict synchronization between sender and receiver.	When you want to allow some buffering and reduce blocking.
Example	`ch := make(chan int)`	`ch := make(chan int, 3)`

KEY DIFFERENCE:

UNBUFFERED: DIRECT, SYNCHRONOUS COMMUNICATION.

BUFFERED: ALLOWS ASYNCHRONOUS COMMUNICATION WITH SOME BUFFERING.

SELECT STATEMENT

IN GO, THE SELECT STATEMENT ALLOWS YOU TO WAIT ON MULTIPLE CHANNEL OPERATIONS, SUCH AS SENDING OR RECEIVING VALUES.

THANK YOU