1 Go

1.1 Data Races / Race Conditions

Data races are the fairly typical type of error that arises whenever you have resources, be it state, variables or larger services, that are shared between several concurrent threads.

With concurrency the access/operation order is no longer predictably linear, so any outcome that depends on the order of operations is now not reliable in a race condition. Such situations can lead to errant behaviour, bugs and crashes.

Avoiding these in golang is principally an issue of writing correct code; avoiding sharing variables where possible, adding mutual exclusion using the `sync` package and such. In addition golang has a built in race condition detector, which can be run on build or test to check for any race conditions within the code base. It does this by monitoring events for any variables that are access by different goroutines in quick succession.

1.2 Defer

Defer is a builtin that when preceding an ordinary function call, will delay the evaluation of the statement until the function that contains the defer call has finished, be that by returning, falling out of the function or panicking.

Multiple defer calls in a single function are executed in a FILO fashion.

1.3 Proxy design pattern in Go

The proxy design pattern is providing an object that intercepts all calls to another object to control access to that object.

In golang you'd implement this with an `ObjectImpl` struct with set methods, a `Object` interface definition that describes the format of the methods, and a `Proxy` object that implements the `Object` interface, contains a reference of an `ObjectImpl` and any additional access control logic that was desired.