Go!

David G. Andersen, Randal E. Bryant
Code first

• Section 1: Intro to Go: Hello world, strings, slices

• Section 2: The Network

• Section 3: A “goroutine”

  • Digression: What is a goroutine and how does it relate to threads and processes?

  • Neat! How do we share between goroutines? Before answering this, let’s get a few more helper ideas — structs and interfaces
1.go notes:

- No semicolons (compiler adds automatically for line-terminators)

- Mandatory scoping of names - “import” and “package” statements. Also `fmt.Println` instead of having global namespace polluted. Like C++ namespaces.
2.go notes

• Limited *type inference* for declarations:

  • foo := valuetype
    shorthand for:  var foo type = valuetype

• You will come to love this and miss it in other languages. C++ “auto” similar.

• Has a built-in string type (is UTF-friendly)
3.go notes

• Strings are *immutable*
  • You can modify them by copying or by copying into []bytes.

• *Slices* are a length-limited window into an array
  • It’s a compile or run-time error to exceed the length (bounds-checked)
Structs and Interfaces
10.go notes

- A Channel is a thread-safe queue that the language and runtime manages for you.
- It does the right thing with blocking threads that read on it, etc.
- Hides a lot of pain of inter-thread communication.
  - Internally, it uses mutexes and semaphores just as one might expect.
11.go notes

- Multiple senders can write to the same channel
- This is really useful for notifications, multiplexing, etc.
- And it’s totally thread-safe. phew.
- But: only one can close, and can’t send after close!
12.go notes

- Select can be used to wait for messages on one of several channels.

- You can implement timeouts by using a timer channel.
13.go notes

• Aw, crud - reads from closed channels return nil.

• But we can get notification of channel closing using the two-argument form.

 • Neat! Go functions can return multiple values

• Note: Hangs after both channels closed because no more work arriving - should add some logic to close things up.