database/sql and context

Ivan Danyliuk, @idanyliuk
Golang BCN, March 2017

Quick Recap

- <u>Context</u> is a type for deadlines, cancelation signals etc.

- It's in standard library since Go 1.7 - <u>package "context"</u>

Quick Recap

type CancelFunc

type Context

func **WithCancel**(parent Context) (ctx Context, cancel CancelFunc)

func **WithDeadline**(parent Context, deadline time.Time) (Context, CancelFunc)

func **WithTimeout**(parent Context, timeout time.Duration) (Context, CancelFunc)
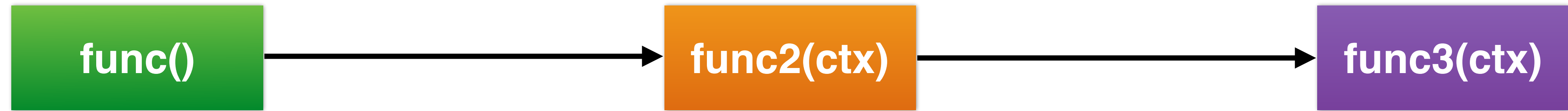
Quick Recap

```go
type Context interface {
    Deadline() (deadline time.Time, ok bool)
    Done() <-chan struct{}
    Err() error
    Value(key interface{}) interface{}
}
```
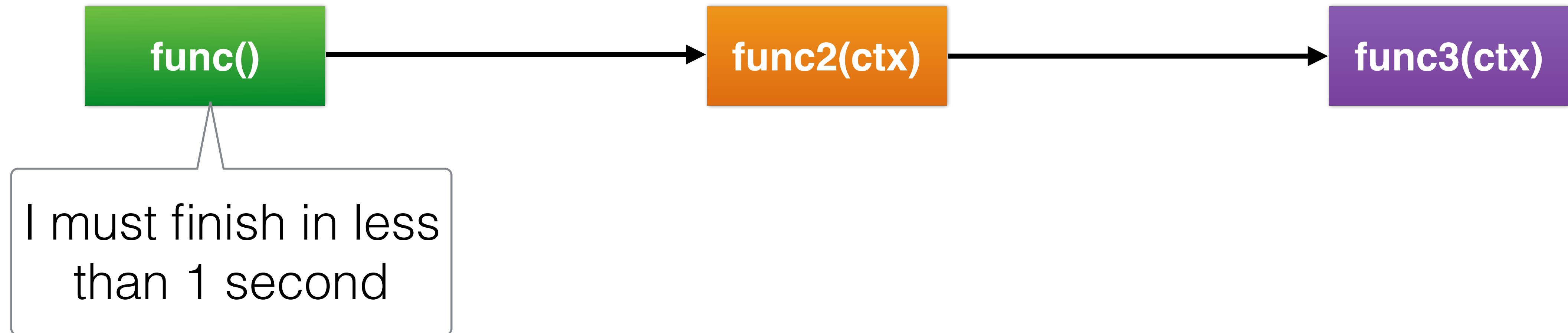
Quick Recap

```go
type Context interface {
    Deadline() (deadline time.Time, ok bool)
    Done() <-chan struct{}
    Err() error
    Value(key interface{}) interface{}
}
```
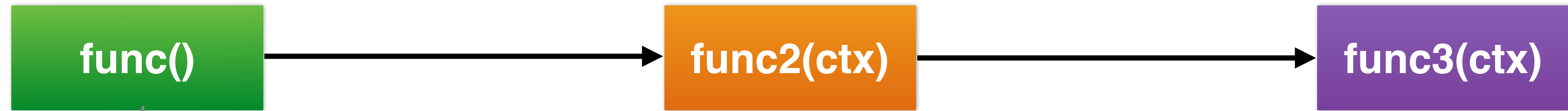
Quick Recap

```
func()  →  func2(ctx)  →  func3(ctx)
```

Quick Recap

func() → func2(ctx) → func3(ctx)

func()

I must finish in less than 1 second
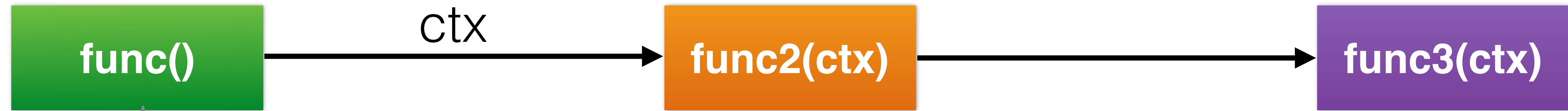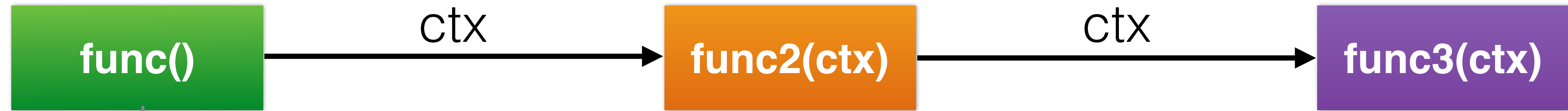
Quick Recap

```
func()  →  func2(ctx)  →  func3(ctx)
```

I must finish in less
than 1 second

```
ctx := ctx.Background()
ctx = context.WithTimeout(ctx, 1*time.Second)
```

Quick Recap

```
func() --ctx--> func2(ctx) -----> func3(ctx)
```

I must finish in less than 1 second

```go
ctx := ctx.Background()
ctx = context.WithTimeout(ctx, 1*time.Second)
```

Quick Recap

```
func()  --ctx-->  func2(ctx)  --ctx-->  func3(ctx)
```
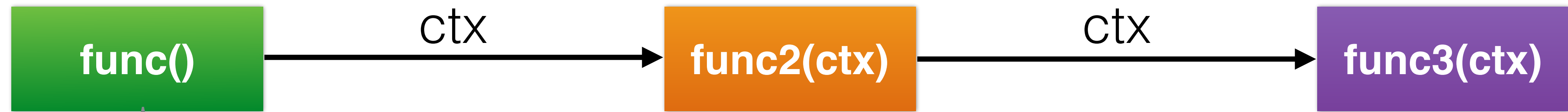
I must finish in less than 1 second

```go
ctx := ctx.Background()
ctx = context.WithTimeout(ctx, 1*time.Second)
```

Quick Recap


```
func()  --ctx-->  func2(ctx)  --ctx-->  func3(ctx)
```

I must finish in less than 1 second

```
ctx := ctx.Background()
ctx = context.WithTimeout(ctx, 1*time.Second)
...
1 second passed
...
ctx.Done() channel is closed
```
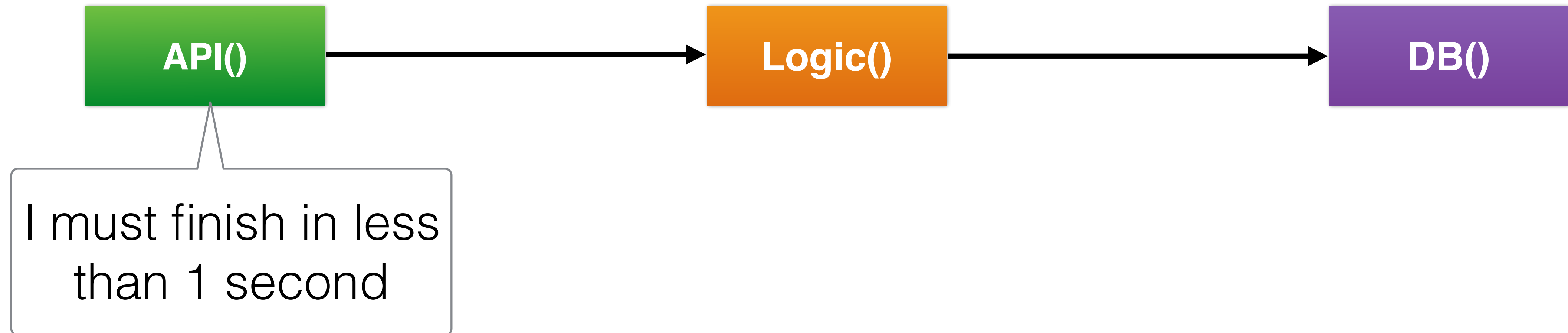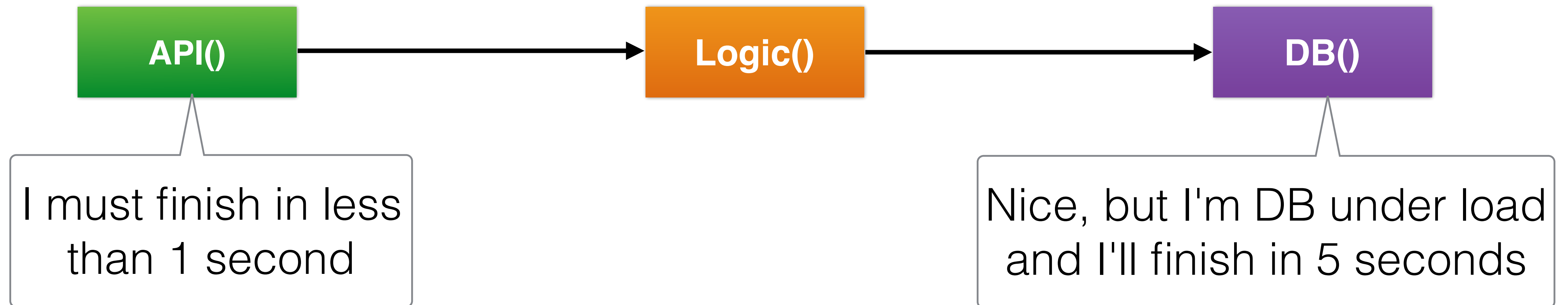
Quick Recap

- You need to instrument your code to work with Context

- In our case, func2() and/or func3() should listen for ctx.Done() channel

- When the channel is closed, they should cancel and release resources
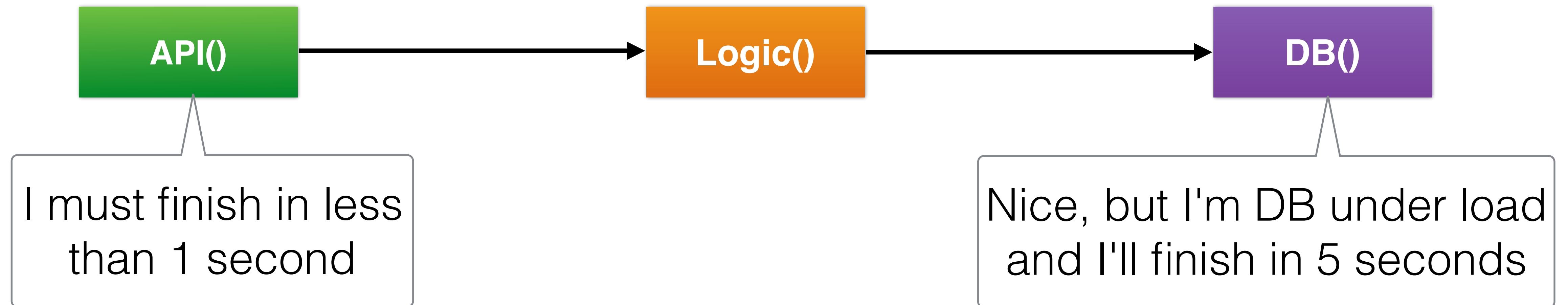
Context for DB calls

sql and context

```
API()  →  Logic()  →  DB()
```
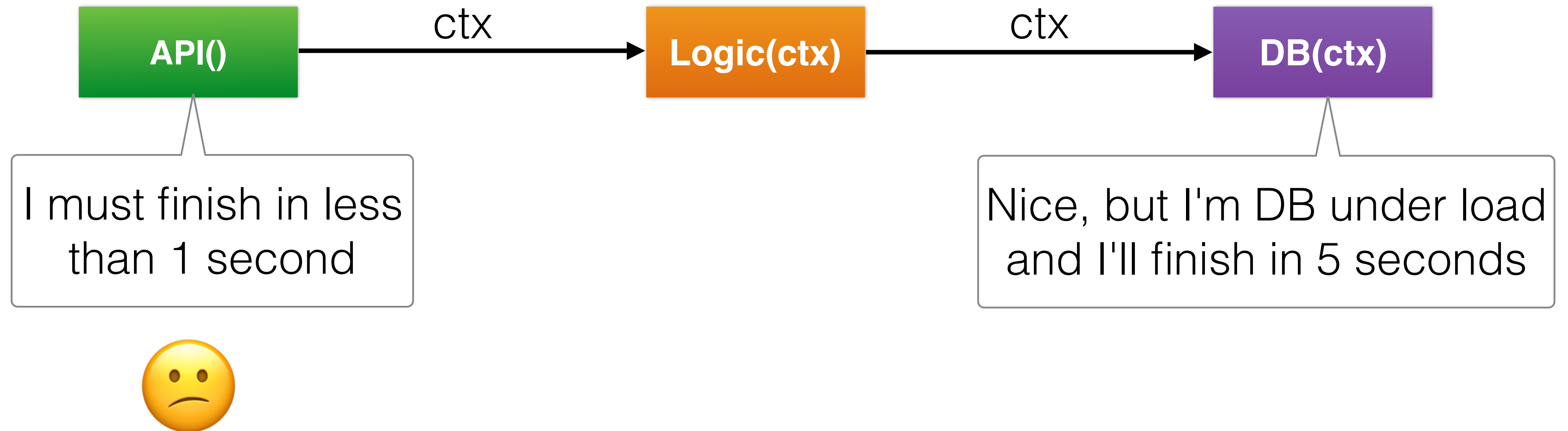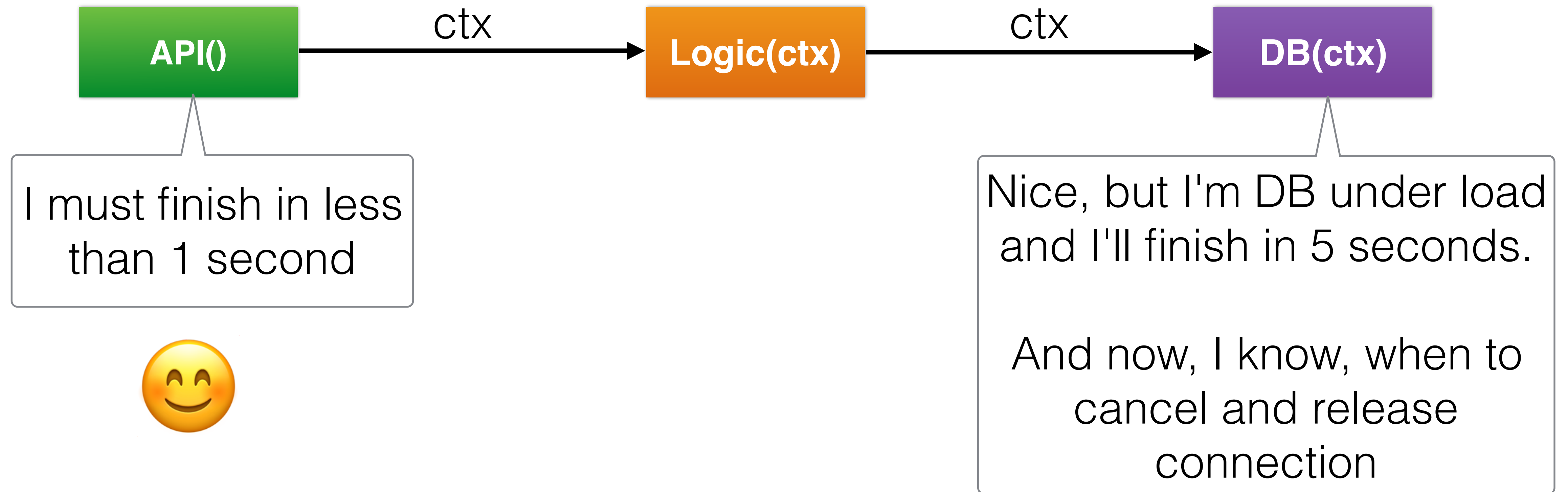
I must finish in less than 1 second

sql and context

sql and context

sql and context

API() → ctx → **Logic(ctx)** → ctx → **DB(ctx)**

I must finish in less than 1 second

☹️

Nice, but I'm DB under load and I'll finish in 5 seconds

sql and context

API() → ctx → **Logic(ctx)** → ctx → **DB(ctx)**

I must finish in less than 1 second

😊

Nice, but I'm DB under load and I'll finish in 5 seconds.

And now, I know, when to cancel and release connection

sql and context

- For types sql.DB, sql.Stmt and sql.Tx:

  - Exec(...) → ExecContext(ctx, ...)

  - Query(...) → QueryContext(ctx, ...)

  - QueryRow(...) → QueryRowContext(ctx, ...)

  - Prepare(...) → PrepareContext(ctx, ...)

- For sql.DB:

  - Begin(...) → BeginTx(ctx, opts *TxOptions)

DEMO

DEMO

https://gist.github.com/divan/
eb11ddc97aab765fb9b093864410fd25

sql and context

- The new Context query methods work for all drivers, but Context cancelation is not responsive unless the driver has been updated to use them.

Drivers status:

MySQL https://github.com/go-sql-driver/mysql and https://github.com/ziutek/mymysql	**WIP**
Postgres https://github.com/lib/pq	**Yes**
SQLite3 https://github.com/mattn/go-sqlite3	**Yes**
Oracle https://github.com/mattn/go-oci8 and https://github.com/rana/ora	**Yes**
Microsoft ActiveX Object DataBase https://github.com/mattn/go-adodb	**Yes**
MSSQL https://github.com/denisenkom/go-mssqldb	**Yes**
Apache Phoenix/Avatica https://github.com/Boostport/avatica	**Yes**
Yandex ClickHouse https://github.com/kshvakov/clickhouse	**Yes**

MySQL WIP

https://github.com/go-sql-driver/mysql/pull/551

Links

- [Go 1.8 changes to database/sql](#)

- [Go Concurrency Patterns: Context](#)

Thank you