Code Organization

This lesson explains how to organize code in Go using an example

WE'LL COVER THE FOLLOWING

- Example

Example

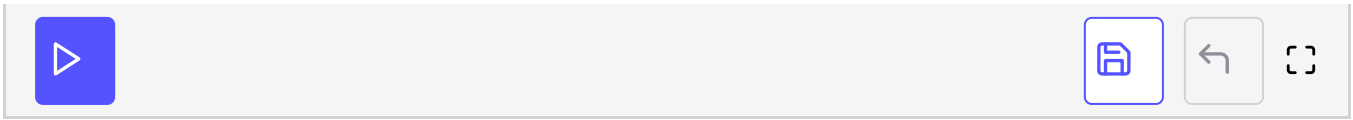
Methods can be defined on any file in the package, but my recommendation is to organize the code as shown below:

Environment Variables

Key:	Value:
GOPATH	/go

```
package main

// list of packages to import
import (
        "fmt"
)

// list of constants
const (
        ConstExample = "const before vars"
)

// list of variables
var (
        ExportedVar    = 42
        nonExportedVar = "so say we all"
)

// Main type(s) for the file,
// try to keep the lowest amount of structs per file when possible.
type User struct {
        FirstName, LastName string
        Location            *UserLocation
}

type UserLocation struct {
```

```
        City    string
        Country string
}

// List of functions
func NewUser(firstName, lastName string) *User {
        return &User{FirstName: firstName,
                LastName: lastName,
                Location: &UserLocation{
                        City:    "Santa Monica",
                        Country: "USA",
                },
        }
}

// List of methods
func (u *User) Greeting() string {
        return fmt.Sprintf("Dear %s %s", u.FirstName, u.LastName)
}

func main() {
  us:=User {FirstName: "Matt",
                LastName: "Damon",
                Location: &UserLocation{
                        City:    "Santa Monica",
                        Country: "USA",}}
  fmt.Println(us.Greeting())
}
```



In fact, you can define a method on `any` type you define in your package, not just structs. You cannot define a method on a type from another package, or on a basic type.

In the next lesson, we will take a look at *type aliasing*.