Preparing a Package to Be Used

Michael Van Sickle

@vansimke

Overview

Member visibility

Documenting packages

Designing a package

Interface strategies

Member Visibility

Public Scope

- Capitalize member
- Available to all consumers

Package Scope

- Lowercase member
- Only available within package

Internal Package

- Can use public- and package-level members
- Scoped to parent package and its descendants

Documenting a Package

Packages

Components

Package Comments

```go
// Copyright 2009 The Go Authors. All rights reserved.
// Use of this source code is governed by a BSD-style
// license that can be found in the LICENSE file.
```

Licensing

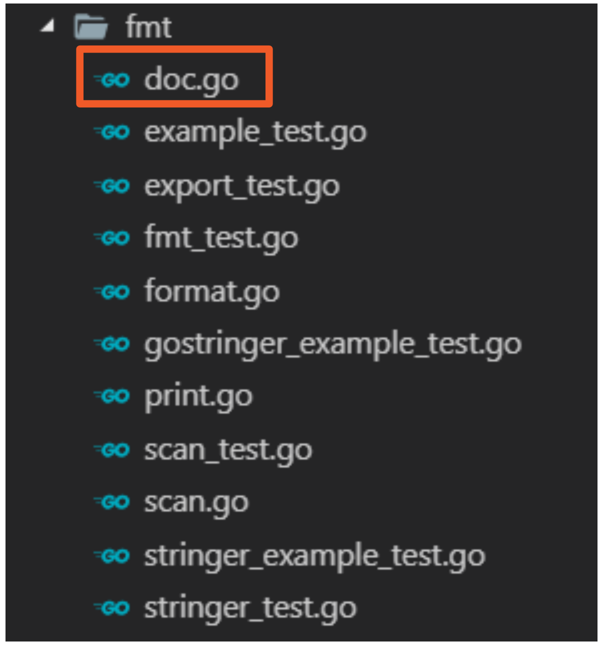
```go
// Package bufio implements buffered I/O. It wraps an io.Reader or io.Writer
// object, creating another object (Reader or Writer) that also implements
// the interface but provides buffering and some help for textual I/O.
```

Package Comment

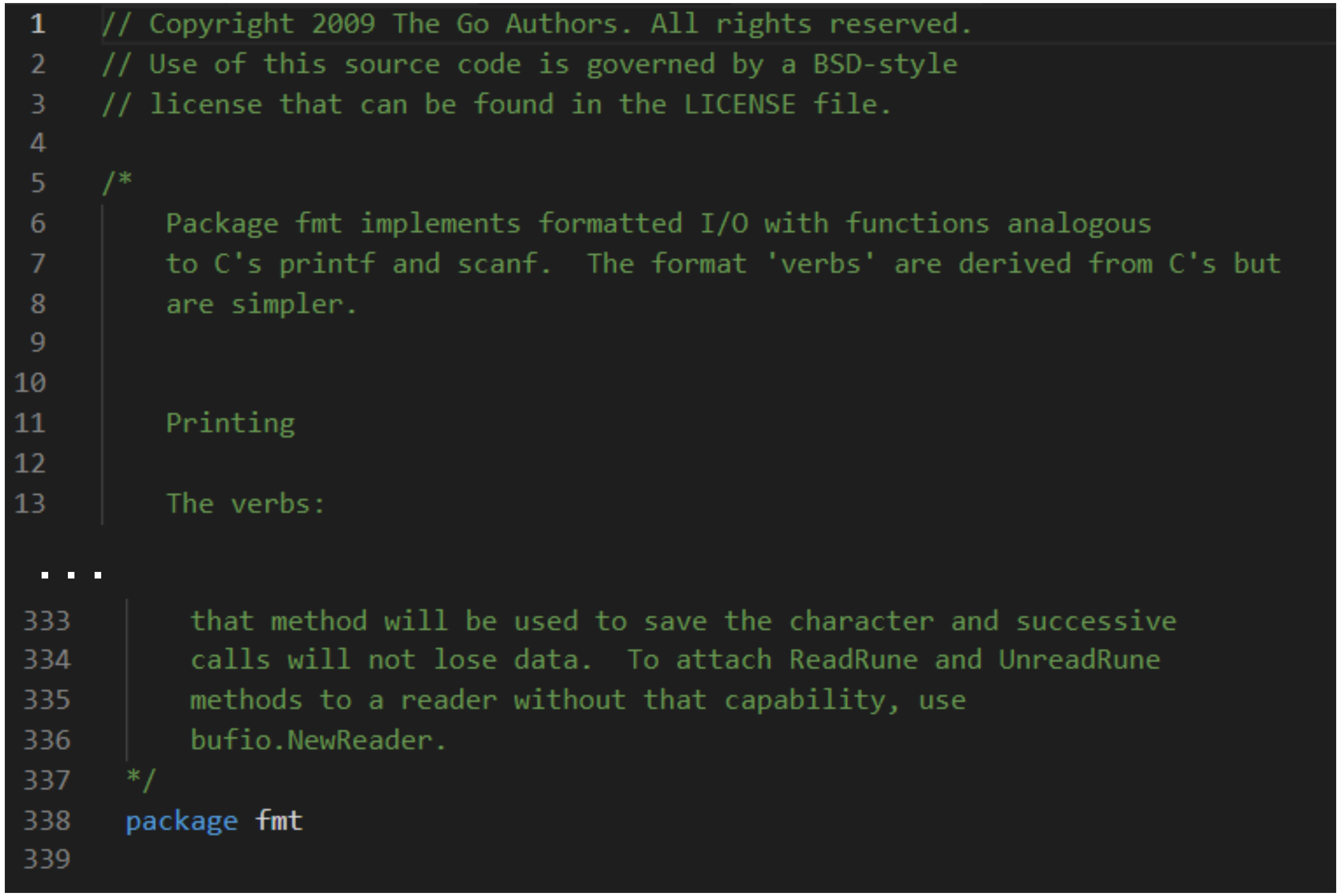
```go
package bufio
```

Declaration

Long Package Comments



```
▲ fmt
    doc.go
    example_test.go
    export_test.go
    fmt_test.go
    format.go
    gostringer_example_test.go
    print.go
    scan_test.go
    scan.go
    stringer_example_test.go
    stringer_test.go
```



```
// Copyright 2009 The Go Authors. All rights reserved.
// Use of this source code is governed by a BSD-style
// license that can be found in the LICENSE file.

/*
	Package fmt implements formatted I/O with functions analogous
	to C's printf and scanf.  The format 'verbs' are derived from C's but
	are simpler.


	Printing

	The verbs:
...
	that method will be used to save the character and successive
	calls will not lose data.  To attach ReadRune and UnreadRune
	methods to a reader without that capability, use
	bufio.NewReader.
*/
package fmt

```

Documenting Public Members

- ✓ Use complete sentences
- ✓ Start first sentence with element's name
- ✓ Write first sentence as a short description of the element

Designing a Package

Provide a clear solution

- Single responsibility
- Cohesive API

Focus on the consumer

- Simple to use
- Minimize API
- Encapsulate changes

Maximize reusability

- Reduce dependencies
- Minimize scope

Interface Strategies

concrete types
configuration

interfaces
behavior

concrete types
configuration and behavior

errors
avoid panics

Encapsulate changes

Maximize flexibility

Avoid abstracting too early

Interface Strategies

concrete types

net/http.Request

interfaces

net/http.Handler

Interface Strategies

concrete types

net/http.Response

errors

net/http.Get

Summary

Member visibility

Documenting packages

Designing a package

Interface strategies