Blocks and Scopes

...

Block

A block is a possibly empty sequence of declarations and statements within matching brace brackets.

Block = "{" StatementList "}" .
StatementList = { Statement ";" } .

Block - explicit

```
package main
import “fmt”
func main() {
    { // start outer block
        a := 1
        fmt.Println(a)
        { // start inner block
            b := 2
            fmt.Println(b)
        } // end inner block
    } // end outer block
}
```

Block - implicit

- The universe block encompasses all Go source text.
-
- Each package has a package block containing all Go source text for that package.
-
- Each file has a file block containing all Go source text in that file.

Block - implicit

- Each "if", "for", and "switch" statement is considered to be in its own implicit block.

```
if i := 0; i < 5; i++ {
    fmt.Println(i)
}

for i := 0; i >= 0 {
    fmt.Println(i)
}

switch i := 2; i * 4 {
case 8:
    fmt.Println(i)
default:
    fmt.Println(“default”)
}
```

Block - implicit

- Each clause in a "switch" or "select" statement acts as an implicit block.

```
switch i := 2; i * 4 {
case 8:
    j := 0
    fmt.Println(i, j)
default:
    // "j" is undefined here
    fmt.Println(“default”)
}
// "j" is undefined here
```

Scope

The scope of a declared identifier is the extent of source text in which the identifier denotes the specified constant, type, variable, function, label, or package.

Scope

The scope of a predeclared identifier is the universe block.
The scope of an identifier denoting a constant, type, variable, or function (but not method) declared at top level (outside any function) is the package block.
The scope of the package name of an imported package is the file block of the file containing the import declaration.
The scope of an identifier denoting a method receiver, function parameter, or result variable is the function body.
The scope of a constant or variable identifier declared inside a function begins at the end of the ConstSpec or VarSpec (ShortVarDecl for short variable declarations) and ends at the end of the innermost containing block.
The scope of a type identifier declared inside a function begins at the identifier in the TypeSpec and ends at the end of the innermost containing block.

shadowing...

An identifier declared in a block may be redeclared in an inner block. While the identifier of the inner declaration is in scope, it denotes the entity declared by the inner declaration.

```
package main

import "fmt"

func main() {
	x := 100
	for i := 0; i < 5; i++ {
		x := i
		fmt.Println(x)
	}
	fmt.Println(x)
}
```

Scopes - Label

Labels are declared by labeled statements and are used in the "break", "continue", and "goto" statements. It is illegal to define a label that is never used. In contrast to other identifiers, labels are not block scoped and do not conflict with identifiers that are not labels. The scope of a label is the body of the function in which it is declared and excludes the body of any nested function.

```
  x := 1
    goto x
x:
    fmt.Println(x)
```