

Le Langage Go

# Interface

## Définition d'un interface

# Syntaxe

```
type MyInterface interface {  
    Foo() error  
    Bar() string  
}
```

# Syntaxe

```
type MyInterface interface {  
    Foo() error  
    Bar() string  
}
```

```
var a MyInterface = ?
```

**Pour implémenter une interface**  
**Il faut redéfinir *toutes* ses fonctions**

# Redéfinition des fonctions

```
type MyInterface interface {  
    Foo() error  
    Bar() string  
}
```

# Redéfinition des fonctions

```
type MyInterface interface {  
    Foo() error  
    Bar() string  
}
```

```
type MyStruct struct { }
```

# Redéfinition des fonctions

```
type MyInterface interface {  
    Foo() error  
    Bar() string  
}  
  
type MyStruct struct { }  
  
func (m MyStruct) Foo() error { ... }  
func (m MyStruct) Bar() string { ... }
```

**Go déduit automatique que `MyStruct`  
est du type `MyInterface`**



# Utilisation de l'interface

```
var a MyInterface = ??
```

# Utilisation de l'interface

```
var a MyInterface = &MyStruct{}
```

# Utilisation de l'interface

```
var a MyInterface = &MyStruct{}  
err := a.Foo() // appelle l'implémentation de MyStruct
```

# En Go, on utilise le **duck typing**

(techniquement, c'est le structural typing)



**“ If it walks like a duck and it quacks like a duck,  
then it must be a duck**

*-- James Withcomb Riley*

”