Go 'syntax' Paketi Özeti

Fonksiyon / Tip	Açıklama	Örnek Kod	Notlar
ParseFile	Bir Go dosyasını/parçayı parse edip *syntax.File döner.	file, err := syntax.ParseFile("example.go", src, nil, 0)	Kaynak kodu io.Reader veya dosya adı ile verilebilir.
File	Bir .go dosyasını temsil eder (paket adı, importlar, fonksiyonlar, değişkenler vs.)	fmt.Println(file.PkgName.Value)	AST'in en üst düğümü.
Node	Tüm AST düğümlerinin ortak arayüzü.	Kullanıcı genelde Walk ile görür.	Tür iddiaları ile (x.(*syntax.FuncDecl)) kullanılabilir.
Decl	Bildirim (declaration) türü: FuncDecl, ImportDecl, VarDecl vs.	for _, d := range file.DeclList { ... }	Kaynak dosyadaki global bildirileri içerir.
FuncDecl	Bir fonksiyon bildirimini temsil eder.	if fn, ok := d.(*syntax.FuncDecl); ok { fmt.Println(fn.Name.Value) }	Fonksiyon adı, parametreleri, gövdesi gibi alanlara erişilir.
ImportDecl	Bir import bildirimini temsil eder.	imp := decl.(*syntax.ImportDecl) fmt.Println(imp.Path.Value)	Import edilen paketlerin yolunu verir.
Expr	İfade düğümlerinin ana türü. Alt tipler: CallExpr, Name, BasicLit vb.	expr := call.Fun.(*syntax.Name)	İfadeleri çözümlemek için kullanılır.
CallExpr	Fonksiyon çağrılarını temsil eder.	fmt.Println("Çağrı:", call.Fun)	fmt.Println("...") gibi ifadeler.
BasicLit	Sayılar, stringler gibi temel literal değerleri temsil eder.	fmt.Println(lit.Value)	Kod içindeki sabit değerler.
Stmt	Tüm ifade cümleleri (statement) için temel arayüz.	Örn: IfStmt, ForStmt, ReturnStmt.	Kod akışını temsil eder.
Walk	AST üzerinde DFS dolaşmı yapar.	syntax.Walk(file, func(n syntax.Node) bool { ...; return true })	Tüm düğümlere ulaşmak için ziyaretçi deseni.
ParserMode	Parse seçenekleri: CheckBranches, AllowGenerics, AllowTypeParams	syntax.ParseFile("x.go", f, nil, syntax.AllowGenerics)	Derleyiciye hangi kuralları dikkate alacağını söyler.