

DEFINITION

- The contextual constraints of a language are rules about this language that cannot be expressed grammatically.

SCOPING

Definitions

- The **scope of a declaration** is the portion of the program over which the declaration takes effect.
- The **scope of an identifier** in a program is the scope of its declaration; i.e. the area in the program where that identifier is meaningful.
- A **block** is any program phrase that delimits the scope of declarations within it.
- The **visibility** of an identifier in a program are the sections of that program where that identifier can be accessed.
- The **extent** or **lifetime** of an identifier describes when in a program's execution a variable has a value.
- The **binding** of an identifier is the association of the name of a variable to its location.

Scoping Structures

- **Monolithic block structure:**
 - The only block is the entire program.
 - All declarations are global in scope (i.e. all identifiers are global)
- **Flat block structure:**
 - A program is partitioned into several disjoint blocks.
 - There are 2 scope levels: global or local to block
- **Nested block structure:**
 - Blocks may be nested one within another.
 - There are multiple scope levels:
 - Scope levels are numbered from the outside in.
 - The scope of a declaration is the block that contains that declaration including its sub-blocks
 - If an identifier is redeclared in a subblock of its original scope, it is not visible anymore in that subblock.

Scoping Mechanisms

- **Static (Lexical) Scoping:**
 - An identifier always refers to its nearest enclosing declaration.
 - This is decided statically at compile time based on the block structure of the program.
- **Dynamic Scoping:**
 - The scope of an identifier is extended to include its lifetime. Bindings are resolved when the identifier is used at run time.
- **Inheritance (OOP):**
 - In addition to the static scoping rules, the scope of an identifier is extended to subclasses of the class in which the identifier is declared.
- **Visibility modifiers (OOP)**
 - The visibility of class and instance identifiers in OOP languages can be modified to extend the scope of that identifier outside of the class where it is declared.

TYPING

- **Static typing**
 - Each identifier has a type: i.e the values that this identifier can take are restricted to a given set.
- **Dynamic Typing**
 - Values are types, but identifiers are not; i.e. they can change type throughout the lifetime of the program.
- **Polymorphism**
 - Identifiers can refer to variables of multiple types simultaneously.
 - Applies mostly to functions and operators.

VARIABLE BINDING

- **Static Binding**
 - Binding of identifier name to location occurs before the program is run.
- **Dynamic Binding**
 - Binding of identifier name to location occurs at run time.