type *Declaration*

Syntax:

`(type` *typespec* {*var*}*`)`

`(`*typespec* {*var*}*`)`

Arguments:

typespec—a *type specifier*.

var—a *variable name*.

Valid Context:

declaration or *proclamation*

Binding Types Affected:

variable

Description:

Affects only variable *bindings* and specifies that the *vars* take on values only of the specified *typespec*. In particular, values assigned to the variables by **setq**, as well as the initial values of the *vars* must be of the specified *typespec*. **type** declarations never apply to function *bindings* (see **ftype**).

A type declaration of a *symbol* defined by **symbol-macrolet** is equivalent to wrapping a **the** expression around the expansion of that *symbol*, although the *symbol*'s *macro expansion* is not actually affected.

The meaning of a type declaration is equivalent to changing each reference to a variable (*var*) within the scope of the declaration to `(the` *typespec var*`)`, changing each expression assigned to the variable (*new-value*) within the scope of the declaration to `(the` *typespec new-value*`)`, and executing `(the` *typespec var*`)` at the moment the scope of the declaration is entered.

A *type* declaration is valid in all declarations. The interpretation of a type declaration is as follows:

1. During the execution of any reference to the declared variable within the scope of the declaration, the consequences are undefined if the value of the declared variable is not of the declared *type*.

2. During the execution of any **setq** of the declared variable within the scope of the declaration, the consequences are undefined if the newly assigned value of the declared variable is not of the declared *type*.

3. At the moment the scope of the declaration is entered, the consequences are undefined if