Objects

Objects are the fundamental way that all information is stored and manipulated in the system. An object has information and operations (properties) that describe some entity that a program works with. For example a receipt might be an object in a retail sales program. The receipt object would have information such as the date, which items were bought, and the sales price. The receipt object would also describe operations such as calculating the tax, updating inventory, and voiding the sale.

All information in the language is contained in objects. This includes ordinary data like numbers and lists, as well as complex data such as a sales receipt. Programming in this system consists of building objects with information and operations.

!!

An object is an entity in the system which can have state, receive messages and interact with other objects. Objects are created in response to user and program actions such as new and copy. Objects are destroyed automatically when they are no longer referenced by any other object.

Properties

An object can have one or more properties. A property is a named value or script in the object (like instance variables and methods in other object systems).

The name of a property is any valid symbol according to the following rules:

•

Any string of alpha-numeric and hyphen characters starting with a letter, case is disregarded

line-width
speed
the-1st-appointment
xjs5000

•

Any string of the graphic characters: !#$%&*+-/:<=>?@\^|~

(except the sequences :=, ?=, ?, !, and ???)

+
==
>=
&&

Inheritance

Each object can inherit properties from another object. These two objects form a parent/child relationship. This can be repeated: the child object can itself be the parent to other objects, and the parent object can be the child of yet some other object (cycles are not allowed). These parent/child relationships form a tree (also know as a directed, acyclic graph). At the root of the tree is an object that has no parent. Typically this object is the object called object[Footnote #1] [Footnote #2].

When something in the system attempts to reference a property of a given object, in general, first the object is checked to see if it has the property. If not then its parent is checked, then the parent's parent and so on until either the property is found, or there are no more parents to check. This process is called 'lookup'. If the property is found in one of the parents, then it is said to be an inherited property. There are some variations on this process, see property scopes, and self sends below.

Every object has a property called parent. The value of the parent property is the parent of the object. If the object doesn't have a parent (for example object doesn't) then the value of this property is nil.

Creating New Objects

New objects are created in response to user and program actions. There are two basic ways that an object can be created: coping and newing. All other ways in which cause an object to be created are simply variants of these two operations. Copying an object creates a new object that is like the original in all respects, except that it is new and distinct object. Newing an object creates a new object which is a child of the original. In this case the new object is generally like the original because it inherits all the properties of it. However, the new object has its own copies of each 'instance' property.

Property Scopes

Each property also has a scope. The scope determines how the property should be handled during the lookup process and during spawning. There are three scopes:

Instance properties are copied for each child during newing. Thus child objects don't need to inherit them as each have their own copy. (Instance properties are inherited in the rare cases where a child doesn't have its own copy.)

Class properties are accessible to all operations. Child objects inherit these properties. They have no special effect on newing.

rivate properties are accessible only for this object. Child objects do not inherit these

properties. They have no special effect on newing.

Values

A property of an object may either name a script or a value. When it names a value, this value be any other object, such as the number 17, the value true, or a sales receipt. This value is actually a reference to the other object, and thus two or more properties in any number of objects can all have the same value.

Garbage Collection

While objects are explicitly created in response to various user and program actions, objects are not explicitly destroyed. Instead, the system figures out when an object is no longer needed (because no property in no object refers to it anymore), and then destroys it. This process is called garbage collection and takes place in the background.