Rules for Evaluation
The rules you (and Scheme) use for evaluating expressions are:

1. self-evaluating: return the value
2. name: return the value associated with the name
3. special form: do something special
4. combination:
   1. evaluate all subexpressions in the combination (order undefined; assume left to right unless specified)
   2. apply operator (value of first subexpression) to operands (values of other subexpressions) and return the result

*Note that evaluation is recursive.

Rules for Application
The rules for application of a procedure are:

1. primitive: just do it
2. compound: evaluate body of procedure with formal parameters replaced by actual argument values

Example

((lambda (x) (* x x)) 10)

expression is an anonymous lambda applied to the value 10
evaluate body of the lambda with x replaced by 10

((lambda (x) (* x x)) 10)
(* 10 10) => 100