avaliar () é uma função, mas está incluído neste capítulo sobre expressões porque realmente deveria ter
foi um operador.As versões mais antigas do idioma definiram uma função de avaliação () e desde então
Em seguida, designers de idiomas e escritores de intérpretes têm colocado restrições sobre ele que o tornam mais
e mais como operador.Interpretadores javascript modernos executam muita análise de código e
otimização.De um modo geral, se uma função chama de avaliação (), o intérprete não pode otimizar que
função.O problema de definir avaliar () em função é que ele pode receber outros nomes:

```
Seja f = aval;
Seja g = f;
```

Se isso for permitido, o intérprete não pode ter certeza de quais funções chamam de avaliação (), para que não possa
otimizar agressivamente.Esta questão poderia ter sido evitada se Eval () fosse um operador (e um
palavra reservada).Aprenderemos (em §4.12.2 e §4.12.3) sobre restrições colocadas em Eval () para fazê -lo
Mais como operador.

4.12.1 Eval ()

Eval () espera um argumento.Se você passar algum valor que não seja um
String, simplesmente retorna esse valor.Se você passar por uma corda, ele tenta
analise a string como código JavaScript, lançando um SyntaxError se falhar.Se
Ele analisa com sucesso a string e avalia o código e retorna
o valor da última expressão ou declaração na string ou
indefinido se a última expressão ou declaração não tivesse valor.Se o
String avaliada lança uma exceção, que a exceção se propaga de
a chamada para avaliar ().

A principal coisa sobre avaliar () (quando invocada assim) é que ele usa
O ambiente variável do código que o chama.Isto é, ele olha para cima
os valores das variáveis ??e define novas variáveis ??e funções no
Da mesma forma que o código local faz.Se uma função define uma variável local x
e depois chama Eval ("X"), ele obterá o valor do local
variável.Se ele chama de avaliar ("x = 1"), altera o valor do local
variável.E se a função chama avaliar ("var y = 3;"),