Próximo:Mutabilidade, Anterior:Tipo de predicados, Acima:Tipos de dados Lisp [Conteúdo][Índice]

2.8 Predicados de Igualdade

Aqui descrevemos funções que testam a igualdade entre dois objetos. Outras funções testam a igualdade de conteúdo entre objetos de tipos específicos, por exemplo, strings. Para esses predicados, consulte o capítulo apropriado que descreve o tipo de dados.

Função: eq *objeto1 objeto2*

Esta função retorna `t`se *objeto1* e *objeto2* forem o mesmo objeto e `nil`caso contrário.

Se *object1* e *object2* forem símbolos com o mesmo nome, normalmente são o mesmo objeto — mas consulte Criando símbolos para exceções. Para outros tipos não numéricos (por exemplo, listas, vetores, strings), dois argumentos com o mesmo conteúdo ou elementos não são necessariamente `eq`entre si: eles são `eq`apenas se forem o mesmo objeto, o que significa que uma mudança no conteúdo de um será refletido pela mesma mudança no conteúdo do outro.

Se *object1* e *object2* forem números com tipos ou valores diferentes, eles não poderão ser o mesmo objeto e `eq`retornarão `nil`. Se forem fixnums com o mesmo valor, são o mesmo objeto e `eq`retornam `t`. Se eles forem calculados separadamente, mas tiverem o mesmo valor e o mesmo tipo numérico não-fixnum, eles podem ou não ser o mesmo objeto e `eq`retornar `t`ou `nil` dependendo se o interpretador Lisp criou um ou dois objetos.

```
(eq 'foo' foo)
     ⇒ t

(eq ?A ?A)
     ⇒ t

(equivalente a 3,0 3,0)
     ⇒ t ou zero
;; Floats iguais podem ou não ser o mesmo objeto.

(eq (criar cadeia 3 ?A) (criar cadeia 3 ?A))
     ⇒ nada

(eq "asdf" "asdf")
     ⇒ t ou zero
;; Constantes de string iguais ou podem não ser o mesmo objeto.

(eq '(1 (2 (3))) '(1 (2 (3))))
     ⇒ nada

(setq foo '(1 (2 (3))))
     ⇒ (1 (2 (3)))
(eq foo foo)
     ⇒ t
(eq foo '(1 (2 (3))))
     ⇒ nada
```

```
(eq [(1 2) 3] [(1 2) 3])
     ⇒ nada


(eq (marcador de ponto) (marcador de ponto))
     ⇒ nada
```

A `make-symbol`função retorna um símbolo não interno, distinto do símbolo usado se você escrever o nome em uma expressão Lisp. Símbolos distintos com o mesmo nome não são `eq`. Consulte Criando Símbolos .

```
(eq (fazer-símbolo "foo") 'foo)
     ⇒ nada
```

O compilador de byte Emacs Lisp pode recolher objetos literais idênticos, como strings literais, em referências ao mesmo objeto, com o efeito de que o código compilado por byte irá comparar tais objetos como `eq`, enquanto a versão interpretada do mesmo código não. Portanto, seu código nunca deve depender de objetos com o mesmo conteúdo literal sendo `eq`ou não `eq`, ele deve usar funções que comparam o conteúdo do objeto, como `equal`, descrito abaixo. Da mesma forma, seu código não deve modificar objetos literais (por exemplo, colocar propriedades de texto em strings literais), pois isso pode afetar outros objetos literais com o mesmo conteúdo, se o compilador de bytes os recolher.

Função: igual *objeto1 objeto2*

Esta função retorna `t`se *objeto1* e *objeto2* tiverem componentes iguais e `nil`caso contrário. Enquanto `eq`testa se seus argumentos são o mesmo objeto, `equal`olha dentro de argumentos não idênticos para ver se seus elementos ou conteúdos são os mesmos. Então, se dois objetos são `eq`, eles são `equal`, mas a recíproca nem sempre é verdadeira.

```
(igual a 'foo' foo)
     ⇒ t


(igual 456 456)
     ⇒ t


(igual a "asdf" "asdf")
     ⇒ t
(eq "asdf" "asdf")
     ⇒ nada


(igual a '(1 (2 (3)))) '(1 (2 (3))))
     ⇒ t
(eq '(1 (2 (3))) '(1 (2 (3))))
     ⇒ nada


(igual [(1 2) 3] [(1 2) 3])
     ⇒ t
(eq [(1 2) 3] [(1 2) 3])
     ⇒ nada
```

```
(igual (marcador de ponto) (marcador de ponto))
     ⇒ t


(eq (marcador de ponto) (marcador de ponto))
     ⇒ nada
```

A comparação de strings faz distinção entre maiúsculas e minúsculas, mas não leva em conta as propriedades do texto — ela compara apenas os caracteres nas strings. Consulte Propriedades de texto . Use `equal-including-properties`também para comparar propriedades de texto. Por motivos técnicos, uma string unibyte e uma string multibyte são `equal`se e somente se contiverem a mesma sequência de códigos de caracteres e todos esses códigos estiverem no intervalo de 0 a 127 (ASCII).

```
(igual a "asdf" "ASDF")
     ⇒ nada
```

No entanto, dois buffers distintos nunca são considerados `equal`, mesmo que seus conteúdos textuais sejam os mesmos.

Para `equal`, a igualdade é definida recursivamente; por exemplo, dadas duas células cons *x* e *y* , retorna se e somente se ambas as expressões abaixo retornarem : `(equal` *x* *y*`)`tt

```
(igual (carro x ) (carro y ))
(igual (cdr x ) (cdr y ))
```

A comparação de listas circulares pode, portanto, causar uma recursão profunda que leva a um erro, e isso pode resultar em um comportamento contra-intuitivo, como `(equal a b)`retornar `t`enquanto `(equal b a)` sinaliza um erro.

Função: igual-incluindo-propriedades *object1 object2*

Esta função se comporta como `equal`em todos os casos, mas também requer que para que duas strings sejam iguais, elas tenham as mesmas propriedades de texto.

```
(igual a "asdf" (propriedade de "asdf" 'asdf t))
     ⇒ t
(propriedades iguais incluindo "asdf"
                            (propertize "asdf" 'asdf t))
     ⇒ nada
```

Próximo:Mutabilidade, Anterior:Tipo de predicados, Acima:Tipos de dados Lisp [Conteúdo][Índice]