Próximo:Acesso de hash, Acima:Tabelas de hash [Conteúdo][Índice]

8.1 Criando tabelas de hash

A função principal para criar uma tabela de hash é `make-hash-table`.

Função: make-hash-table ***&rest palavra-chave-args***

Esta função cria uma nova tabela de hash de acordo com os argumentos especificados. Os argumentos devem consistir em palavras-chave alternadas (símbolos específicos reconhecidos especialmente) e valores correspondentes a elas.

Várias palavras-chave fazem sentido em `make-hash-table`, mas as únicas duas que você realmente precisa saber são `:teste` `:weakness`.

`:test` ***`test`***

Isso especifica o método de pesquisa de chave para esta tabela de hash. O padrão é `eql`; `eqe` `equal`são outras alternativas:

`eql`

As chaves que são números são iguais se forem `equal`, ou seja, se forem iguais em valor e ambas forem inteiras ou ambas forem de ponto flutuante; caso contrário, dois objetos distintos nunca são os mesmos.

`eq`

Quaisquer dois objetos Lisp distintos são diferentes como chaves.

`equal`

Dois objetos Lisp são iguais, como chaves, se forem iguais de acordo com `equal`.

Você pode usar `define-hash-table-test`(consulte Definindo Hash) para definir possibilidades adicionais para *test* .

`:weakness` ***`weak`***

A fraqueza de uma tabela de hash especifica se a presença de uma chave ou valor na tabela de hash a preserva da coleta de lixo.

O valor, *fraco* , deve ser um de `nil`, `key`, `value`, `key-or-value`, `key-and-value`, ou `t` que seja um alias para `key-and-value`. Se *fraco* for `key` , a tabela de hash não impede que suas chaves sejam coletadas como lixo (se não forem referenciadas em nenhum outro lugar); se uma chave específica for coletada, a associação correspondente será removida da tabela de hash.

Se *fraco* for `value`, a tabela de hash não impedirá que os valores sejam coletados como lixo (se não forem referenciados em nenhum outro lugar); se um valor específico for coletado, a associação correspondente será removida da tabela de hash.

Se *fraco* for `key-and-value`ou `t`, tanto a chave quanto o valor devem estar ativos para preservar a associação. Assim, a tabela de hash não protege nem as chaves nem os valores da coleta de lixo; se qualquer um deles for coletado como lixo, isso removerá a associação.

Se *fraco* for `key-or-value`, tanto a chave quanto o valor podem preservar a associação. Assim, as associações são removidas da tabela de hash quando sua chave e valor seriam coletados como lixo (se não para referências de tabelas de hash fracas).

O padrão para *fraco* é `nil`, para que todas as chaves e valores referenciados na tabela de hash sejam preservados da coleta de lixo.

`:size` *`size`*

Isso especifica uma dica de quantas associações você planeja armazenar na tabela de hash. Se você souber o número aproximado, poderá tornar as coisas um pouco mais eficientes especificando-o dessa maneira. Se você especificar um tamanho muito pequeno, a tabela de hash aumentará automaticamente quando necessário, mas isso levará algum tempo extra.

O tamanho padrão é 65.

`:rehash-size` *`rehash-size`*

Quando você adiciona uma associação a uma tabela de hash e a tabela está cheia, ela cresce automaticamente. Esse valor especifica como aumentar a tabela de hash naquele momento.

Se *rehash-size* for um número inteiro, ele deverá ser positivo e a tabela de hash aumentará adicionando aproximadamente isso ao tamanho nominal. Se *rehash-size* for ponto flutuante, é melhor que seja maior que 1, e a tabela de hash cresce multiplicando o tamanho antigo por aproximadamente esse número.

O valor padrão é 1,5.

`:rehash-threshold` *`threshold`*

Isso especifica o critério para quando a tabela de hash está cheia (portanto, deve ser maior). O valor, *threshold* , deve ser um número de ponto flutuante positivo, não maior que 1. A tabela de hash está cheia sempre que o número real de entradas exceder o tamanho nominal multiplicado por uma aproximação desse valor. O padrão para o *limite* é 0,8125.

Você também pode criar uma nova tabela de hash usando a representação impressa para tabelas de hash. O leitor Lisp pode ler essa representação impressa, desde que cada elemento na tabela de hash especificada tenha uma sintaxe de leitura válida (consulte Representação impressa). Por exemplo, o seguinte especifica uma nova tabela de hash contendo as chaves `key1`e `key2`(ambos os símbolos) associados a `val1` (um símbolo) e `300`(um número), respectivamente.

```
#s(tamanho da tabela de hash 30 dados (chave1 val1 chave2 300))
```

A representação impressa para uma tabela de hash consiste em '`#s`' seguido por uma lista começando com '`tabela de hash`'. O restante da lista deve consistir em zero ou mais pares de valor de propriedade especificando as propriedades da tabela de hash e o conteúdo inicial. As propriedades e valores são lidos literalmente. Os nomes de propriedade válidos são `size`, `test`, `weakness`, `rehash-size`, `rehash-threshold`e `data`. A `data`propriedade deve ser uma lista de pares chave-valor para o conteúdo inicial; as outras propriedades têm os mesmos significados que as `make-hash-table`palavras-chave correspondentes (`:size`, `:test`, etc.), descritas acima.

Observe que você não pode especificar uma tabela de hash cujo conteúdo inicial inclua objetos que não tenham sintaxe de leitura, como buffers e quadros. Esses objetos podem ser adicionados à tabela de hash após sua criação.

Próximo:Acesso de hash, Acima:Tabelas de hash [Conteúdo][Índice]