Próximo:Dicas para definir, Anterior:Variáveis nulas, Acima:Variáveis [Conteúdo][Índice]

12.5 Definindo Variáveis Globais

Uma *definição de variável* é uma construção que anuncia sua intenção de usar um símbolo como uma variável global. Ele usa os formulários especiais `defvar`ou `defconst`, que estão documentados abaixo.

Uma definição de variável serve a três propósitos. Primeiro, ele informa às pessoas que lêem o código que o símbolo deve *ser* usado de uma determinada maneira (como uma variável). Segundo, informa isso ao sistema Lisp, fornecendo opcionalmente um valor inicial e uma string de documentação. Terceiro, fornece informações para ferramentas de programação como `etags`, permitindo que encontrem onde a variável foi definida.

A diferença entre `defconst`e `defvar`é principalmente uma questão de intenção, servindo para informar os leitores humanos sobre se o valor deve mudar. O Emacs Lisp não impede que você altere o valor de uma variável definida com `defconst`. Uma diferença notável entre as duas formas é que `defconst` inicializa incondicionalmente a variável, enquanto `defvar` a inicializa somente se for originalmente nula.

Para definir uma variável personalizável, você deve usar `defcustom` (que chama `defvar`como uma sub-rotina). Consulte Definições de Variáveis .

Forma especial: símbolo defvar *[valor [doc-string]]*

> Esta forma especial define o *símbolo* como uma variável. Observe que o *símbolo* não é avaliado; o símbolo a ser definido deve aparecer explicitamente no `defvar`formulário. A variável é marcada como *special* , o que significa que ela deve sempre ser vinculada dinamicamente (consulte Variable Scoping).
>
> Se *value* for especificado e *symbol* for void (ou seja, não tem valor vinculado dinamicamente; consulte Void Variables), então *value* será avaliado e *symbol* será definido para o resultado. Mas se o *símbolo* não for nulo, o *valor* não será avaliado e o valor do *símbolo* permanecerá inalterado. Se o *valor* for omitido, o valor do *símbolo* não será alterado em nenhum caso.
>
> Observe que especificar um valor, mesmo `nil`, marca a variável como especial permanentemente. Considerando que se o *valor* for omitido, a variável será marcada apenas como especial localmente (ou seja, dentro do escopo lexical atual, ou arquivo se estiver no nível superior). Isso pode ser útil para suprimir avisos de compilação de bytes, consulte Erros do compilador .
>
> Se o *símbolo* tiver uma associação de buffer-local no buffer atual, `defvar`atua no valor padrão, que é independente de buffer, em vez da associação de buffer-local. Ele define o valor padrão se o valor padrão for nulo. Consulte Variáveis Locais de Buffer .
>
> Se o *símbolo* já estiver vinculado lexicalmente (por exemplo, se o `defvar` formulário ocorrer em um `let`formulário com vinculação léxica habilitada), `defvar`define o valor dinâmico. A ligação léxica permanece em vigor até que sua construção de ligação seja encerrada. Consulte Escopo variável .
>
> `defvar`Quando você avalia um formulário de nível superior com o `C-M-x`modo Emacs Lisp (`eval-defun`), um recurso especial de `eval-defun`organiza para definir a variável incondicionalmente, sem testar se seu valor é nulo.
>
> Se o argumento *doc-string* for fornecido, ele especifica a string de documentação para a variável (armazenada na `variable-documentation`propriedade do símbolo). Consulte Documentação .

Aqui estão alguns exemplos. Este formulário define `foo`, mas não inicializa:

```
(defvar foo)
     ⇒ fo
```

Este exemplo inicializa o valor de `barto 23`e fornece uma string de documentação:

```
(barra defvar 23
  "O peso normal de uma barra.")
     ⇒ barra
```

O `defvar`formulário retorna *símbolo* , mas normalmente é usado no nível superior em um arquivo onde seu valor não importa.

Para obter um exemplo mais elaborado de uso `defvar`sem valor, consulte Exemplo defvar local .

Forma especial: valor do símbolo defconst *[doc-string]*

Essa forma especial define o *símbolo* como um valor e o inicializa. Ele informa a pessoa que está lendo seu código que o *símbolo* possui um valor global padrão, estabelecido aqui, que não deve ser alterado pelo usuário ou por outros programas. Observe que o *símbolo* não é avaliado; o símbolo a ser definido deve aparecer explicitamente no arquivo `defconst`.

O `defconst`formulário, como `defvar`, marca a variável como *especial* , o que significa que ela sempre deve ser vinculada dinamicamente (consulte Variable Scoping). Além disso, marca a variável como arriscada (consulte File Local Variables).

`defconst`sempre avalia *value* e define o valor de *symbol* para o resultado. Se o *símbolo* tiver uma ligação de buffer-local no buffer atual, `defconst`define o valor padrão, não o valor de buffer-local. (Mas você não deve fazer ligações de buffer-local para um símbolo definido com `defconst`.)

Um exemplo do uso de `defconst`é a definição do Emacs de `float-pi`—a constante matemática *pi* , que não deve ser alterada por ninguém (apesar das tentativas da Legislatura do Estado de Indiana). Como o segundo formulário ilustra, no entanto, `defconst`é apenas consultivo.

```
(defconst float-pi 3.141592653589793 "O valor de Pi.")
     ⇒ float-pi
(setq float-pi 3)
     ⇒ float-pi
float-pi
     ⇒ 3
```

Aviso: Se você usar um formulário `defconst`ou `defvar` especial enquanto a variável tiver uma ligação local (feita com `let`, ou um argumento de função), ela define a ligação local em vez da ligação global. Isso não é o que você geralmente quer. Para evitar isso, use esses formulários especiais no nível superior em um arquivo, onde normalmente nenhuma ligação local está em vigor, e certifique-se de carregar o arquivo antes de fazer uma ligação local para a variável.

Próximo:Dicas para definir, Anterior:Variáveis nulas, Acima:Variáveis [Conteúdo][Índice]