Próximo:Aconselhando Funções Nomeadas, Acima:Funções de aconselhamento [Conteúdo][Índice]

13.11.1 Primitivos para manipular conselhos

Macro: add-function ***onde colocar a função e adereços opcionais***

Essa macro é a maneira prática de adicionar a função de aconselhamento à função armazenada no *local* (consulte Variáveis generalizadas).

onde determina como a *função* é composta com a função existente, por exemplo, se a *função* deve ser chamada antes ou depois da função original. Consulte Advice Combinators , para obter a lista de maneiras disponíveis de compor as duas funções.

Ao modificar uma variável (cujo nome geralmente termina com `-function`), você pode escolher se a *função* é usada globalmente ou apenas no buffer atual: se *place* é apenas um símbolo, então *function* é adicionada ao valor global de *place* . Considerando que se *place* estiver na forma , onde *símbolo* é uma expressão que retorna o nome da variável, então a *função* só será adicionada no buffer atual. Finalmente, se você quiser modificar uma variável léxica, terá que usar . `(local` *`symbol`*`)(var` *`variable`*`)`

Cada função adicionada com `add-function`pode ser acompanhada por uma lista de associação de *props* de propriedades . Atualmente, apenas duas dessas propriedades têm um significado especial:

`name`

Isso dá um nome ao conselho, que `remove-function`pode ser usado para identificar qual função remover. Normalmente usado quando *function* é uma função anônima.

`depth`

Isso especifica como solicitar o aviso, caso vários avisos estejam presentes. Por padrão, a profundidade é 0. Uma profundidade de 100 indica que este conselho deve ser mantido o mais profundo possível, enquanto uma profundidade de -100 indica que deve permanecer como a peça mais externa. Quando duas recomendações especificam a mesma profundidade, a mais recente adicionada será a mais externa.

Para `:before`aconselhamento, ser mais externo significa que este aconselhamento será executado primeiro, antes de qualquer outro aconselhamento, enquanto ser mais interno significa que ele será executado imediatamente antes da função original, sem nenhum outro aconselhamento executado entre ele e a função original. Da mesma forma, para o `:after`conselho o mais interno significa que ele será executado logo após a função original, sem nenhum outro aviso no meio, enquanto o mais externo significa que ele será executado logo no final após todos os outros avisos. Um aviso mais interno `:override`substituirá apenas a função original e outros avisos serão aplicados a ela, enquanto um `:override`aviso mais externo substituirá não apenas a função original, mas também todos os outros avisos aplicados a ela.

Se *a função* não for interativa, a função combinada herdará a especificação interativa, se houver, da função original. Caso contrário, a função combinada será interativa e usará a especificação interativa de *function* . Uma exceção: se a especificação interativa da *função* for uma função (ou seja, uma `lambda`expressão ou um `fbound` símbolo em vez de uma expressão ou uma string), então a especificação interativa da função combinada será uma chamada para essa função com o único argumento especificação da função original. Para interpretar a especificação recebida como argumento, use `advice-eval-interactive-spec`.

Nota: A especificação interativa de *function* será aplicada à função combinada e, portanto, deve obedecer à convenção de chamada da função combinada em vez da de *function* . Em muitos casos, isso não faz diferença, pois são idênticos, mas importa para `:around`, `:filter-args`, e `:filter-return`, onde *a função* recebe argumentos diferentes da função original armazenada no *local* .

Macro: função de remover a função de *lugar*

Esta macro remove a *função* da função armazenada no *local* . Isso só funciona se a *função* foi adicionada ao *lugar* usando `add-function`.

function é comparada com funções adicionadas ao *place* usando `equal`, para tentar fazê-la funcionar também com expressões lambda. Além disso, é comparado também com a `name`propriedade das funções adicionadas ao *lugar* , que pode ser mais confiável do que comparar expressões lambda usando `equal`.

Função: conselho-função-membro-p *conselho função-def*

Retorne não - `nil`se o *conselho* já estiver em *function-def* . Como `remove-function`acima, em vez de o *conselho* ser a função real, também pode ser o `name`conselho.

Função: conselho-função-mapc *f função-def*

Chame a função *f* para cada conselho que foi adicionado a *function-def* . *f* é chamado com dois argumentos: a função de aconselhamento e suas propriedades.

Função: especificação de *especificação* de conselho-eval-interativo

Avalie a *especificação* interativa da mesma forma que uma chamada interativa para uma função com essa especificação e, em seguida, retorne a lista correspondente de argumentos que foi criada. Por exemplo, `(advice-eval-interactive-spec "r\nP")`retornará uma lista de três elementos, contendo os limites da região e o argumento do prefixo atual.

Por exemplo, se você quiser fazer o prompt de comando `C-x m` ( ) para um '`compose-mail`A `partir de:`' cabeçalho, você poderia dizer algo assim:

```
(defun my-compose-mail-advice (args originais e restantes)
  "Ler de: endereço interativamente."
  (interativo
   (lambda (especificação)
     (let* ((usuário-endereço de e-mail
             (completando-leia "De: "
                               '("one.address@example.net"
                                 "alternative.address@example.net")))
            (de (mensagem-fazer-do usuário-nome-completo
                                   usuário-e-mail-endereço))
            (especificação (especificação do conselho-eval-especificação i
       ;; Coloque o cabeçalho From no argumento OTHER-HEADERS.
       (push (contra 'From from) (nth 2 spec))
       especificação)))
  (aplicar argumentos originais))

(advice-add 'compose-mail :around #'my-compose-mail-advice)
```

Próximo:Aconselhando Funções Nomeadas, Acima:Funções de aconselhamento [Conteúdo][Índice]