|  |  |
| --- | --- |
| unlist {base} | R Documentation |

**Flatten Lists**

**Description**

Given a list structure x, unlist simplifies it to produce a vector which contains all the atomic components which occur in x.

**Usage**

unlist(x, recursive = TRUE, use.names = TRUE)

**Arguments**

|  |  |
| --- | --- |
| x | an **R** object, typically a list or vector. |
| recursive | logical. Should unlisting be applied to list components of x? |
| use.names | logical. Should names be preserved? |

**Details**

unlist is generic: you can write methods to handle specific classes of objects, see [InternalMethods](http://127.0.0.1:14695/library/base/help/InternalMethods), and note, e.g., [relist](http://127.0.0.1:14695/library/base/help/relist) with the unlist method for relistable objects.

If recursive = FALSE, the function will not recurse beyond the first level items in x.

Factors are treated specially. If all non-list elements of x are factors (or ordered factors) then the result will be a factor with levels the union of the level sets of the elements, in the order the levels occur in the level sets of the elements (which means that if all the elements have the same level set, that is the level set of the result).

x can be an atomic vector, but then unlist does nothing useful, not even drop names.

By default, unlist tries to retain the naming information present in x. If use.names = FALSE all naming information is dropped.

Where possible the list elements are coerced to a common mode during the unlisting, and so the result often ends up as a character vector. Vectors will be coerced to the highest type of the components in the hierarchy NULL < raw < logical < integer < double < complex < character < list < expression: pairlists are treated as lists.

A list is a (generic) vector, and the simplified vector might still be a list (and might be unchanged). Non-vector elements of the list (for example language elements such as names, formulas and calls) are not coerced, and so a list containing one or more of these remains a list. (The effect of unlisting an [lm](http://127.0.0.1:14695/library/base/help/lm) fit is a list which has individual residuals as components.) Note that unlist(x) now returns x unchanged also for non-vector x, instead of signalling an error in that case.

**Value**

NULL or an expression or a vector of an appropriate mode to hold the list components.

The output type is determined from the highest type of the components in the hierarchy NULL < raw < logical < integer < double < complex < character < list < expression, after coercion of pairlists to lists.

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

**See Also**

[c](http://127.0.0.1:14695/library/base/help/c), [as.list](http://127.0.0.1:14695/library/base/help/as.list), [relist](http://127.0.0.1:14695/library/base/help/relist).

**Examples**

unlist(options())

unlist(options(), use.names = FALSE)

l.ex <- list(a = list(1:5, LETTERS[1:5]), b = "Z", c = NA)

unlist(l.ex, recursive = FALSE)

unlist(l.ex, recursive = TRUE)

l1 <- list(a = "a", b = 2, c = pi+2i)

unlist(l1) # a character vector

l2 <- list(a = "a", b = as.name("b"), c = pi+2i)

unlist(l2) # remains a list

ll <- list(as.name("sinc"), quote( a + b ), 1:10, letters, expression(1+x))

utils::str(ll)

for(x in ll)

stopifnot(identical(x, unlist(x)))