Description

Create, coerce to or test for a double-precision vector.

Usage

```
double(length = 0)
as.double(x, ...)
is.double(x)

single(length = 0)
as.single(x, ...)
```

Arguments

length	A non-negative integer specifying the desired length. Double values will be coerced to integer: supplying an argument of length other than one is an error.
x	object to be coerced or tested.
...	further arguments passed to or from other methods.

Details

`double` creates a double-precision vector of the specified length. The elements of the vector are all equal to `0`. It is identical to `numeric`.

`as.double` is a generic function. It is identical to `as.numeric`. Methods should return an object of base type `"double"`.

`is.double` is a test of double type.

R *has no single precision data type. All real numbers are stored in double precision format.* The functions `as.single` and `single` are identical to `as.double` and `double` except they set the attribute `Csingle` that is used in the `.C` and `.Fortran` interface, and they are intended only to be used in that context.

Value

`double` creates a double-precision vector of the specified length. The elements of the vector are all equal to `0`.

`as.double` attempts to coerce its argument to be of double type: like `as.vector` it strips attributes including names. (To ensure that an object is of double type without stripping attributes, use `storage.mode`.) Character strings containing optional whitespace followed by either a decimal representation or a hexadecimal representation (starting with `0x` or `0X`) can be converted, as can special values such as `"NA"`, `"NaN"`, `"Inf"` and `"infinity"`, irrespective of case.

`as.double` for factors yields the codes underlying the factor levels, not the numeric representation of the labels, see also `factor`.

`is.double` returns `TRUE` or `FALSE` depending on whether its argument is of double type or not.

Double-precision values

All R platforms are required to work with values conforming to the IEC 60559 (also known as IEEE 754) standard. This basically works with a precision of 53 bits, and represents to that precision a range of absolute values from about 2×10^{-308} to 2×10^{308}. It also has special values `NaN` (many of them), plus and minus infinity and plus and minus zero (although R acts as if these are the same). There are also *denormal(ized)* (or *subnormal*) numbers with values below the range given above but represented to less precision.

See `.Machine` for precise information on these limits. Note that ultimately how double precision numbers are handled is down to the CPU/FPU and compiler.

In IEEE 754-2008/IEC60559:2011 this is called 'binary64' format.

Note on names

It is a historical anomaly that R has two names for its floating-point vectors, `double` and `numeric` (and formerly had `real`).

`double` is the name of the type. `numeric` is the name of the mode and also of the implicit class. As an S4 formal class, use `"numeric"`.

The potential confusion is that R has used *mode* `"numeric"` to mean 'double or integer', which conflicts with the S4 usage. Thus `is.numeric` tests the mode, not the class, but `as.numeric` (which is identical to `as.double`) coerces to the class.

References

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

https://en.wikipedia.org/wiki/IEEE_754-1985, https://en.wikipedia.org/wiki/IEEE_754-2008, https://en.wikipedia.org/wiki/IEEE_754-2019, https://en.wikipedia.org/wiki/Double_precision, https://en.wikipedia.org/wiki/Denormal_number.

See Also

`integer`, `numeric`, `storage.mode`.

Examples

```
is.double(1)
all(double(3) == 0)
```