Table of Contents

double

This package provides comprehensive support for double-precision floating-point arithmetic, including basic operations, trigonometric functions, exponential and logarithmic functions, as well as utility functions for handling special values
.

Constants and Special Values

The package provides several important constants and special floating-point valu
es:

```
test "special values" {

  inspect(@double.infinity, content="Infinity")
  inspect(@double.neg_infinity, content="-Infinity")
  inspect(@double.not_a_number, content="NaN")


  inspect(@double.max_value, content="1.7976931348623157e+308")
  inspect(@double.min_value, content="-1.7976931348623157e+308")
  inspect(@double.min_positive, content="2.2250738585072014e-308")
}
```

Basic Operations

Basic mathematical operations and rounding functions:

```
test "basic operations" {

  inspect(@double.abs(-3.14), content="3.14")


  inspect(@double.floor(3.7), content="3")
  inspect(@double.ceil(3.2), content="4")
  inspect(@double.round(3.5), content="4")
  inspect(@double.trunc(3.7), content="3")


  inspect(2.0.pow(3), content="8")


  inspect((-3.14).signum(), content="-1")
  inspect(2.0.signum(), content="1")


  inspect(@double.from_int(42), content="42")
}
```

Special Value Testing

Functions for testing special floating-point values and comparing numbers:

```

test "special value testing" {

  inspect(@double.not_a_number.is_nan(), content="true")
  inspect(@double.infinity.is_inf(), content="true")
  inspect(@double.infinity.is_pos_inf(), content="true")
  inspect(@double.neg_infinity.is_neg_inf(), content="true")


  let relative_tolerance = 1.e-9
  inspect(@double.is_close(0.1 + 0.2, 0.3, relative_tolerance~), content
}
```

Binary Representation

Functions for converting doubles to their binary representation:

```

test "binary representation" {
  let num = 1.0



  inspect(
    num.to_be_bytes(),
    content=(
      #|b"\x3f\xf0\x00\x00\x00\x00\x00\x00"
    ),
  )
  inspect(
    num.to_le_bytes(),
    content=(
      #|b"\x00\x00\x00\x00\x00\x00\xf0\x3f"
    ),
  )
}
```

Note: Most methods can be called either as a method (d.to_be_bytes()) or as a package function (@double.to_be_bytes(d)).