DAKSH

Standard Library Reference

Phase 0 — Web Interpreter Edition

Document 03 of the Daksh Documentation Set

A reference for the standard-library modules currently implemented in the Daksh web interpreter.

Scope: This document records the currently established Phase 0 standard-library APIs. It does not invent APIs or syntax for unfinished features.

1. How to Use the Standard Library

Import a module before using its functions. Module functions are accessed with the module name followed by a dot and the function name.

```
import math
tell math.sqrt(144)
```

The currently documented Phase 0 modules are:

```
math
string
list
random
file
time
system
```

The browser-based runtime places normal browser security limits on operations that access the local machine or filesystem.

2. math Module

The **math** module provides common numeric operations and mathematical constants.

math.abs

Syntax: math.abs(value)

Purpose: Returns the absolute value.

Example:

```
tell math.abs(-25)
```

math.sqrt

Syntax: math.sqrt(value)

Purpose: Returns the square root.

Example:

```
tell math.sqrt(144)
```

math.pow

Syntax: math.pow(base, exponent)

Purpose: Raises a number to a power.

Example:

```
tell math.pow(2, 5)
```

math.floor

Syntax: math.floor(value)

Purpose: Rounds a numeric value down to the nearest integer.

Example:

```
tell math.floor(4.8)
```

math.ceil

Syntax: math.ceil(value)

Purpose: Rounds a numeric value up to the nearest integer.

Example:

```
tell math.ceil(4.2)
```

math.round

Syntax: math.round(value)

Purpose: Rounds a numeric value.

Example:

```
tell math.round(4.6)
```

math.max

Syntax: math.max(...values)

Purpose: Returns the largest supplied numeric value.

Example:

```
tell math.max(10, 20, 5)
```

math.min

Syntax: math.min(...values)

Purpose: Returns the smallest supplied numeric value.

Example:

```
tell math.min(10, 20, 5)
```

math.pi

Syntax: math.pi

Purpose: Provides the mathematical constant π.

Example:

```
tell math.pi
```

math.e

Syntax: math.e

Purpose: Provides Euler's number.

Example:

```
tell math.e
```

Math example

```
import math

set radius = 5
set area = math.pi * radius ** 2

tell math.sqrt(144)
tell area
```

3. string Module

The **string** module provides common text operations.

string.length

Syntax: string.length(text)

Purpose: Returns the length of a string.

Example:

```
tell string.length("Daksh")
```

string.upper

Syntax: string.upper(text)

Purpose: Returns the text in uppercase.

Example:

```
tell string.upper("daksh")
```

string.lower

Syntax: string.lower(text)

Purpose: Returns the text in lowercase.

Example:

```
tell string.lower("DAKSH")
```

string.trim

Syntax: string.trim(text)

Purpose: Removes surrounding whitespace.

Example:

```
tell string.trim("  Daksh  ")
```

string.contains

Syntax: string.contains(text, search)

Purpose: Checks whether text contains the search value.

Example:

```
tell string.contains("Daksh Language", "Language")
```

string.startsWith

Syntax: string.startsWith(text, prefix)

Purpose: Checks whether text begins with the supplied prefix.

Example:

```
tell string.startsWith("Daksh", "Da")
```

string.endsWith

Syntax: string.endsWith(text, suffix)

Purpose: Checks whether text ends with the supplied suffix.

Example:

```
tell string.endsWith("Daksh", "sh")
```

string.replace

Syntax: string.replace(text, search, replacement)

Purpose: Replaces matching text with replacement text.

Example:

```
tell string.replace("Hello Daksh", "Daksh", "World")
```

String processing example

```
import string

set rawName = "  manideep  "
set cleanName = string.trim(rawName)
set displayName = string.upper(cleanName)

tell displayName
tell string.length(cleanName)
tell string.contains(cleanName, "deep")
```

4. list Module

The **list** module provides common list operations. List indexing starts at zero.

list.length

Syntax: list.length(items)

Purpose: Returns the number of elements in a list.

Example:

```
tell list.length([10, 20, 30])
```

list.add

Syntax: list.add(items, value)

Purpose: Adds a value to the list.

Example:

```
set numbers = [10, 20]
list.add(numbers, 30)
tell numbers
```

list.remove

Syntax: list.remove(items, value)

Purpose: Removes the specified value from the list.

Example:

```
set numbers = [10, 20, 30]
list.remove(numbers, 20)
tell numbers
```

list.contains

Syntax: list.contains(items, value)

Purpose: Checks whether a value exists in the list.

Example:

```
tell list.contains([10, 20, 30], 20)
```

list.first

Syntax: list.first(items)

Purpose: Returns the first element.

Example:

```
tell list.first([10, 20, 30])
```

list.last

Syntax: list.last(items)

Purpose: Returns the last element.

Example:

```
tell list.last([10, 20, 30])
```

List processing example

```
import list

set numbers = [10, 20, 30]

tell list.length(numbers)
tell list.first(numbers)
tell list.last(numbers)

list.add(numbers, 40)
tell list.contains(numbers, 40)

list.remove(numbers, 20)
tell numbers
```

5. random Module

The **random** module provides random numeric values and random selection.

random.number

Syntax: random.number()

Purpose: Returns a random number.

Example:

```
tell random.number()
```

random.integer

Syntax: random.integer(min, max)

Purpose: Returns a random integer in the supplied range.

Example:

```
tell random.integer(1, 10)
```

random.choice

Syntax: random.choice(items)

Purpose: Returns a random element from a list.

Example:

```
set colors = ["red", "green", "blue"]
tell random.choice(colors)
```

Random example — number guessing

```
import random

set secret = random.integer(1, 10)

tell "A number between 1 and 10 was selected."
tell secret
```

For a real guessing game, combine this with input(), integer(), and conditions.

6. file Module

The **file** module is designed for browser-safe file operations. Browser security means Daksh cannot freely inspect arbitrary filesystem paths.

file.read

Syntax: file.read()

Purpose: Opens a browser file picker and reads a selected local file.

Example:

```
import file
set content = file.read()
tell content
```

file.write

Syntax: file.write(content)

Purpose: Uses a browser save flow to write content to a file.

Example:

```
import file
file.write("Hello from Daksh!")
```

file.download

Syntax: file.download(filename, content)

Purpose: Downloads generated content through the browser.

Example:

```
import file
file.download("hello.txt", "Hello from Daksh!")
```

file.exists

Syntax: file.exists(...)

Purpose: Reserved for existence checks, but arbitrary local-path inspection is restricted in the browser runtime.

Example:

```
import file
file.exists("example.txt")
```

Notes: In the current browser implementation this operation intentionally reports that arbitrary path inspection is not available.

File module safety note

A browser page does not receive unrestricted access to the user's filesystem. The current Daksh file API therefore uses browser file-picker/save/download mechanisms rather than pretending that arbitrary local paths are available.

7. time Module

The **time** module provides date/time values, formatting, date arithmetic, comparisons, and date-part helpers.

time.today

Syntax: time.today()

Purpose: Returns the current date.

Example:

```
import time
tell time.today()
```

time.now

Syntax: time.now()

Purpose: Returns the current time.

Example:

```
import time
tell time.now()
```

time.nowDateTime

Syntax: time.nowDateTime()

Purpose: Returns the current date and time.

Example:

```
import time
tell time.nowDateTime()
```

time.format

Syntax: time.format(date, pattern)

Purpose: Formats a date/time value using the supported pattern tokens.

Example:

```
import time
set today = time.today()
tell time.format(today, "YYYY-MM-DD")
```

Notes: Current formatting tokens: YYYY, MM, DD, HH, mm, ss.

time.addDays

Syntax: time.addDays(date, days)

Purpose: Adds a number of days.

Example:

```
import time
set future = time.addDays(time.today(), 7)
tell future
```

time.removeDays

Syntax: time.removeDays(date, days)

Purpose: Removes a number of days.

Example:

```
import time
set previous = time.removeDays(time.today(), 1)
tell previous
```

time.isBefore

Syntax: time.isBefore(date1, date2)

Purpose: Checks whether date1 is before date2.

Example:

```
import time
tell time.isBefore(time.removeDays(time.today(), 1), time.today())
```

time.isAfter

Syntax: time.isAfter(date1, date2)

Purpose: Checks whether date1 is after date2.

Example:

```
import time
tell time.isAfter(time.addDays(time.today(), 1), time.today())
```

time.dayName

Syntax: time.dayName(date)

Purpose: Returns the day name.

Example:

```
import time
tell time.dayName(time.today())
```

time.monthName

Syntax: time.monthName(date)

Purpose: Returns the month name.

Example:

```
import time
tell time.monthName(time.today())
```

time.yearNumber

Syntax: time.yearNumber(date)

Purpose: Returns the year number.

Example:

```
import time
tell time.yearNumber(time.today())
```

time.monthNumber

Syntax: time.monthNumber(date)

Purpose: Returns the month number.

Example:

```
import time
tell time.monthNumber(time.today())
```

time.dayNumber

Syntax: time.dayNumber(date)

Purpose: Returns the day number.

Example:

```
import time
tell time.dayNumber(time.today())
```

Time example

```
import time

set today = time.today()
set nextWeek = time.addDays(today, 7)

tell today
tell nextWeek
tell time.format(today, "YYYY-MM-DD")
tell time.dayName(today)
tell time.monthName(today)
tell time.isAfter(nextWeek, today)
```

8. system Module

The **system** module exposes runtime and environment information plus selected system operations. Some operations are naturally constrained by the browser environment.

system.osName

Syntax: system.osName()

Purpose: Returns the operating-system name exposed by the runtime.

Example:

```
import system
tell system.osName()
```

system.osVersion

Syntax: system.osVersion()

Purpose: Returns the operating-system version information exposed by the runtime.

Example:

```
import system
tell system.osVersion()
```

system.deviceName

Syntax: system.deviceName()

Purpose: Returns the device/host name exposed by the runtime.

Example:

```
import system
tell system.deviceName()
```

system.userName

Syntax: system.userName()

Purpose: Returns the user name exposed by the runtime.

Example:

```
import system
tell system.userName()
```

system.language

Syntax: system.language()

Purpose: Returns the runtime/browser language information.

Example:

```
import system
tell system.language()
```

system.architecture

Syntax: system.architecture()

Purpose: Returns architecture information exposed by the runtime.

Example:

```
import system
tell system.architecture()
```

system.dakshVersion

Syntax: system.dakshVersion()

Purpose: Returns the current Daksh runtime version value.

Example:

```
import system
tell system.dakshVersion()
```

system.memory

Syntax: system.memory()

Purpose: Returns runtime memory information available to the implementation.

Example:

```
import system
tell system.memory()
```

system.batteryLevel

Syntax: system.batteryLevel()

Purpose: Returns available battery information when supported by the browser/device.

Example:

```
import system
tell system.batteryLevel()
```

system.isOnline

Syntax: system.isOnline()

Purpose: Checks the browser/runtime online status.

Example:

```
import system
tell system.isOnline()
```

system.sleep

Syntax: system.sleep(seconds)

Purpose: Pauses asynchronously for the requested number of seconds.

Example:

```
import system
tell "Before"
system.sleep(1)
tell "After"
```

Notes: This is asynchronous in the web interpreter.

system.exit

Syntax: system.exit()

Purpose: Requests runtime exit behavior.

Example:

```
import system
system.exit()
```

Notes: Actual browser behavior can be constrained by browser security.

system.restart

Syntax: system.restart()

Purpose: Requests runtime restart behavior.

Example:

```
import system
system.restart()
```

Notes: Actual browser behavior can be constrained by browser security.

9. Combined Standard-Library Program

This example uses math, string, and list together with Daksh functions, foreach, objects, conditions, and assignment.

```
import math
import string
import list
```

```
function processStudent(name, marks) {
    set cleanName = string.upper(string.trim(name))
    set total = 0

    foreach mark in marks {
        set total += mark
    }

    set average = total / list.length(marks)

    if average >= 90 {
        return {
            name: cleanName,
            average: average,
            grade: "A"
        }
    } else if average >= 75 {
        return {
            name: cleanName,
            average: average,
            grade: "B"
        }
    } else {
        return {
            name: cleanName,
            average: average,
            grade: "C"
        }
    }
}

set marks = [85, 90, 95]
set student = processStudent("  manideep  ", marks)

tell student.name
tell student.average
tell student.grade

set student.grade = "A+"
tell student.grade

tell math.sqrt(144)
```

Expected output:

```
MANIDEEP
90
A
A+
12
```

10. Module Quick Reference

```
math
  abs, sqrt, pow, floor, ceil, round, max, min, pi, e

string
  length, upper, lower, trim, contains,
  startsWith, endsWith, replace

list
  length, add, remove, contains, first, last

random
  number, integer, choice

file
  read, write, download, exists

time
```

```
  today, now, nowDateTime, format,
  addDays, removeDays, isBefore, isAfter,
  dayName, monthName, yearNumber,
  monthNumber, dayNumber

system
  osName, osVersion, deviceName, userName,
  language, architecture, dakshVersion,
  memory, batteryLevel, isOnline,
  sleep, exit, restart
```

11. Browser Runtime Notes

Daksh Phase 0 is currently implemented as a web interpreter. Standard-library functions that interact with the operating system, battery, network state, or files depend on what the browser exposes. The API should not be documented as unrestricted native OS access.

12. API Stability Note

This document describes the Phase 0 APIs that were established and tested. Future modules or additional functions should be added as a new language/library decision rather than silently changing this baseline.

Daksh — Standard Library Reference • Phase 0