

# Package: toonlite (via r-universe)

May 20, 2026

**Title** Read, Write, Validate, Stream, and Convert TOON Data

**Version** 0.1.0

**Description** A minimal-dependency, performance-first R package for reading, writing, validating, streaming, and converting TOON (Token-Oriented Object Notation) data. Optimized for very large tabular files with robust diagnostics. Supports lossless JSON conversion and tabular CSV/Parquet/Feather conversion.

**License** MIT + file LICENSE

**URL** <https://github.com/aljrjico/toonlite>

**BugReports** <https://github.com/aljrjico/toonlite/issues>

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**SystemRequirements** C++17

**Suggests** testthat (>= 3.0.0), jsonlite, arrow

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Repository** <https://aljrjico.r-universe.dev>

**Date/Publication** 2026-02-19 10:12:12 UTC

**RemoteUrl** <https://github.com/aljrjico/toonlite>

**RemoteRef** HEAD

**RemoteSha** 69033579bc60a17b1b0eb7b0ee0798821c55f705

## Contents

as_tabular_toon . . . . .	2
assert_toon . . . . .	3
conditions . . . . .	4
conversions . . . . .	4

core-io . . . . .	4
csv_to_toon . . . . .	5
feather_to_toon . . . . .	6
format_toon . . . . .	6
from_toon . . . . .	7
json_to_toon . . . . .	8
parquet_to_toon . . . . .	9
print.toon . . . . .	10
read_toon . . . . .	10
read_toon_df . . . . .	11
streaming . . . . .	12
tabular-io . . . . .	13
to_toon . . . . .	13
toon_info . . . . .	14
toon_peek . . . . .	15
toon_stream_items . . . . .	15
toon_stream_rows . . . . .	17
toon_stream_write_rows . . . . .	18
toon_to_csv . . . . .	19
toon_to_feather . . . . .	20
toon_to_json . . . . .	21
toon_to_parquet . . . . .	22
validate_toon . . . . .	23
validation . . . . .	24
write_toon . . . . .	24
write_toon_df . . . . .	25

**Index** **26**

---

as_tabular_toon	<i>Convert array-of-objects to tabular representation</i>
-----------------	---

---

**Description**

Convert array-of-objects to tabular representation

**Usage**

```
as_tabular_toon(x, strict = TRUE, warn = TRUE)
```

**Arguments**

x	List of lists/named lists (array of objects).
strict	Logical. If TRUE (default), validate input structure.
warn	Logical. If TRUE (default), emit warnings for type promotions.

**Details**

In permissive mode:

- Union schema across all records
- Missing fields filled with NA
- Type promotion to accommodate all values

**Value**

A data.frame suitable for write\_toon\_df().

**Examples**

```
# Convert array of objects to data.frame
records <- list(
  list(name = "Alice", age = 30),
  list(name = "Bob", age = 25)
)
df <- as_tabular_toon(records)
```

---

assert\_toon

*Assert TOON validity*

---

**Description**

Assert TOON validity

**Usage**

```
assert_toon(
  x,
  is_file = FALSE,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE
)
```

**Arguments**

**x** Character scalar containing TOON text, or file path if is\_file=TRUE.

**is\_file** Logical. If TRUE, x is treated as a file path.

**strict** Logical. If TRUE (default), enforce strict TOON syntax.

**allow\_comments** Logical. If TRUE (default), allow # and // comments.

**allow\_duplicate\_keys** Logical. If TRUE (default), allow duplicate keys.

**Details**

Throws `toonlite_parse_error` condition if invalid, with attached location and snippet information.

**Value**

Invisibly returns `TRUE` if valid.

**Examples**

```
# Valid TOON (returns invisibly)
assert_toon('name: "Alice"')

# Invalid TOON (throws error)
## Not run:
assert_toon('invalid: [')

## End(Not run)
```

---

conditions	<i>toonlite error and warning conditions</i>
------------	--

---

**Description**

Functions for creating and signaling toonlite-specific conditions.

---

conversions	<i>TOON Conversion Functions</i>
-------------	----------------------------------

---

**Description**

Functions for converting between TOON and other formats.

---

core-io	<i>Core TOON I/O functions</i>
---------	--------------------------------

---

**Description**

Functions for parsing and serializing TOON data.

---

csv_to_toon	<i>Convert CSV to TOON</i>
-------------	----------------------------

---

### Description

Convert CSV to TOON

### Usage

```
csv_to_toon(  
  path_csv,  
  path_toon,  
  tabular = TRUE,  
  strict = TRUE,  
  col_types = NULL  
)
```

### Arguments

path_csv	Character scalar. Path to input CSV file.
path_toon	Character scalar. Path to output TOON file.
tabular	Logical. If TRUE (default), write as tabular TOON array.
strict	Logical. If TRUE (default), enforce strict syntax on output.
col_types	Named character vector specifying column types.

### Value

Invisibly returns NULL.

### Examples

```
## Not run:  
csv_to_toon("data.csv", "data.toon")  
  
## End(Not run)
```

---

feather_to_toon	<i>Convert Feather to TOON</i>
-----------------	--------------------------------

---

**Description**

Convert Feather to TOON

**Usage**

```
feather_to_toon(path_feather, path_toon, tabular = TRUE, strict = TRUE)
```

**Arguments**

path_feather	Character scalar. Path to input Feather file.
path_toon	Character scalar. Path to output TOON file.
tabular	Logical. If TRUE (default), write as tabular TOON array.
strict	Logical. If TRUE (default), enforce strict syntax on output.

**Details**

Requires the arrow package. If not installed, an error is thrown.

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:  
feather_to_toon("data.feather", "data.toon")  
  
## End(Not run)
```

---

format_toon	<i>Format/pretty-print TOON</i>
-------------	---------------------------------

---

**Description**

Format/pretty-print TOON

**Usage**

```
format_toon(
    x,
    is_file = FALSE,
    indent = 2L,
    canonical = FALSE,
    allow_comments = TRUE
)
```

**Arguments**

`x` Character scalar containing TOON text, or file path if `is_file=TRUE`.

`is_file` Logical. If TRUE, `x` is treated as a file path.

`indent` Integer. Number of spaces for indentation (default 2).

`canonical` Logical. If TRUE, use stable representation with lexicographic key ordering. Default FALSE (preserve original order).

`allow_comments` Logical. If TRUE (default), allow comments in input.

**Value**

Character scalar with formatted TOON. If `is_file=TRUE`, returns formatted text (does not rewrite file).

**Examples**

```
# Format with default indent
format_toon('name:"Alice"')

# Format with canonical key ordering
format_toon('b: 1\na: 2', canonical = TRUE)
```

---

from\_toon

*Parse TOON from string*


---

**Description**

Parse TOON from string

**Usage**

```
from_toon(
    text,
    strict = TRUE,
    simplify = TRUE,
    allow_comments = TRUE,
    allow_duplicate_keys = TRUE
)
```

**Arguments**

text	Character scalar or raw vector containing TOON data. If raw, treated as UTF-8 bytes.
strict	Logical. If TRUE (default), enforce strict TOON syntax.
simplify	Logical. If TRUE (default), simplify homogeneous arrays to atomic vectors.
allow_comments	Logical. If TRUE (default), allow # and // comments.
allow_duplicate_keys	Logical. If TRUE (default), allow duplicate keys in objects (last-one-wins semantics).

**Value**

R object representing the parsed TOON data:

- object -> named list
- array -> list or atomic vector (if simplified)
- primitives -> logical/integer/double/character/NULL

**Examples**

```
# Parse simple object
from_toon('name: "Alice"\nage: 30')

# Parse array
from_toon('[3]:\n - 1\n - 2\n - 3')

# Parse with comments
from_toon('# comment\nkey: "value"', allow_comments = TRUE)
```

---

json_to_toon	<i>Convert JSON to TOON</i>
--------------	-----------------------------

---

**Description**

Convert JSON to TOON

**Usage**

```
json_to_toon(json, pretty = TRUE, strict = TRUE, allow_comments = TRUE)
```

**Arguments**

json	Character scalar containing JSON.
pretty	Logical. If TRUE (default), use multi-line formatting.
strict	Logical. If TRUE (default), enforce strict syntax.
allow_comments	Logical. If TRUE (default), allow comments in output.

**Details**

Requires the jsonlite package. If not installed, an error is thrown.

**Value**

Character scalar with class "toon".

**Examples**

```
## Not run:
toon <- json_to_toon('{"name": "Alice", "age": 30}')

## End(Not run)
```

---

parquet_to_toon	<i>Convert Parquet to TOON</i>
-----------------	--------------------------------

---

**Description**

Convert Parquet to TOON

**Usage**

```
parquet_to_toon(path_parquet, path_toon, tabular = TRUE, strict = TRUE)
```

**Arguments**

path_parquet	Character scalar. Path to input Parquet file.
path_toon	Character scalar. Path to output TOON file.
tabular	Logical. If TRUE (default), write as tabular TOON array.
strict	Logical. If TRUE (default), enforce strict syntax on output.

**Details**

Requires the arrow package. If not installed, an error is thrown.

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:
parquet_to_toon("data.parquet", "data.toon")

## End(Not run)
```

---

print.toon	<i>Print method for toon class</i>
------------	------------------------------------

---

**Description**

Print method for toon class

**Usage**

```
## S3 method for class 'toon'  
print(x, ...)
```

**Arguments**

x	TOON string object
...	Additional arguments (ignored)

**Value**

The input x, invisibly.

**Examples**

```
x <- to_toon(list(a = 1, b = 2))  
print(x)
```

---

read_toon	<i>Read TOON from file</i>
-----------	----------------------------

---

**Description**

Read TOON from file

**Usage**

```
read_toon(  
  file,  
  strict = TRUE,  
  simplify = TRUE,  
  allow_comments = TRUE,  
  allow_duplicate_keys = TRUE,  
  encoding = "UTF-8"  
)
```

**Arguments**

`file` Character scalar. Path to TOON file.  
`strict` Logical. If TRUE (default), enforce strict TOON syntax.  
`simplify` Logical. If TRUE (default), simplify homogeneous arrays to atomic vectors.  
`allow_comments` Logical. If TRUE (default), allow # and // comments.  
`allow_duplicate_keys` Logical. If TRUE (default), allow duplicate keys in objects.  
`encoding` Character. File encoding (default "UTF-8").

**Value**

R object representing the parsed TOON data.

**Examples**

```
# Read from file
## Not run:
data <- read_toon("config.toon")

## End(Not run)
```

---

read_toon_df	<i>Read tabular TOON to data.frame</i>
--------------	--

---

**Description**

Read tabular TOON to data.frame

**Usage**

```
read_toon_df(  
  file,  
  key = NULL,  
  strict = TRUE,  
  allow_comments = TRUE,  
  allow_duplicate_keys = TRUE,  
  warn = TRUE,  
  col_types = NULL,  
  ragged_rows = c("expand_warn", "error"),  
  n_mismatch = c("warn", "error"),  
  max_extra_cols = Inf  
)
```

**Arguments**

file	Character scalar. Path to TOON file.
key	Character scalar or NULL. If non-NULL, extract tabular array at root[key] (root must be object).
strict	Logical. If TRUE (default), enforce strict TOON syntax.
allow_comments	Logical. If TRUE (default), allow # and // comments.
allow_duplicate_keys	Logical. If TRUE (default), allow duplicate keys.
warn	Logical. If TRUE (default), emit warnings for schema changes.
col_types	Named character vector specifying column types: "logical", "integer", "double", or "character".
ragged_rows	Character. How to handle rows with different field counts: <ul style="list-style-type: none"> <li>• "expand_warn" (default): Fill missing fields with NA, expand schema for extra fields, emit aggregated warning.</li> <li>• "error": Require all rows have same field count.</li> </ul>
n_mismatch	Character. How to handle declared row count mismatch: <ul style="list-style-type: none"> <li>• "warn" (default): Accept mismatch, return observed rows, emit warning.</li> <li>• "error": Require observed row count equals declared [N].</li> </ul>
max_extra_cols	Numeric. Maximum new columns allowed via schema expansion. Default Inf (no limit).

**Value**

A base data.frame.

**Examples**

```
## Not run:
# Read tabular TOON file
df <- read_toon_df("data.toon")

# Read nested tabular array
df <- read_toon_df("config.toon", key = "records")

## End(Not run)
```

---

streaming

*Streaming TOON I/O*


---

**Description**

Functions for streaming TOON data in batches.

---

tabular-io	<i>Tabular TOON I/O (data.frame)</i>
------------	--------------------------------------

---

### Description

Functions for reading and writing tabular TOON data as data.frames.

---

to_toon	<i>Serialize R object to TOON</i>
---------	-----------------------------------

---

### Description

Serialize R object to TOON

### Usage

```
to_toon(x, pretty = TRUE, indent = 2L, strict = TRUE, allow_comments = FALSE)
```

### Arguments

x	R object to serialize.
pretty	Logical. If TRUE (default), use multi-line formatting.
indent	Integer. Number of spaces for indentation (default 2).
strict	Logical. If TRUE (default), reject NaN/Inf values.
allow_comments	Logical. For future use (writers generally should not emit comments).

### Details

Type conversions:

- factor -> character
- Date -> ISO 8601 string (YYYY-MM-DD)
- POSIXct -> ISO 8601 string (YYYY-MM-DDTHH:MM:SSZ)
- NA values -> null

### Value

Character scalar with class "toon" containing the TOON representation.

**Examples**

```
# Serialize simple list
to_toon(list(name = "Alice", age = 30))

# Serialize vector
to_toon(c(1, 2, 3))

# Compact format
to_toon(list(x = 1, y = 2), pretty = FALSE)
```

---

toon\_info

*Get TOON file info*


---

**Description**

Get TOON file info

**Usage**

```
toon_info(file, allow_comments = TRUE)
```

**Arguments**

`file` Character scalar. Path to TOON file.  
`allow_comments` Logical. If TRUE (default), allow comments.

**Value**

A list with components:

- `array_count`: Integer. Number of arrays in file.
- `object_count`: Integer. Number of objects in file.
- `has_tabular`: Logical. Whether file contains tabular arrays.
- `declared_rows`: Integer or NA. Declared row count if tabular.

**Examples**

```
## Not run:
info <- toon_info("data.toon")
cat("Arrays:", info$array_count, "\n")
cat("Has tabular:", info$has_tabular, "\n")

## End(Not run)
```

---

toon_peek	<i>Peek at TOON file structure</i>
-----------	------------------------------------

---

**Description**

Peek at TOON file structure

**Usage**

```
toon_peek(file, n = 50L, allow_comments = TRUE)
```

**Arguments**

`file` Character scalar. Path to TOON file.  
`n` Integer. Number of lines to preview (default 50).  
`allow_comments` Logical. If TRUE (default), allow comments.

**Value**

A list with components:

- `type`: Character. Top-level type ("object", "array", "tabular\_array", "unknown").
- `first_keys`: Character vector. First few keys if top-level is object.
- `preview`: Character vector. First n lines.

**Examples**

```
## Not run:
info <- toon_peek("data.toon")
cat("Type:", info$type, "\n")
cat("Preview:\n", info$preview, sep = "\n")

## End(Not run)
```

---

toon_stream_items	<i>Stream non-tabular array items</i>
-------------------	---------------------------------------

---

**Description**

Stream non-tabular array items

**Usage**

```
toon_stream_items(
  file,
  key = NULL,
  callback,
  batch_size = 1000L,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  warn = TRUE,
  simplify = TRUE
)
```

**Arguments**

file	Character scalar. Path to TOON file.
key	Character scalar or NULL. If non-NULL, extract array at root[key].
callback	Function. Called with each batch as a list.
batch_size	Integer. Number of items per batch (default 1000).
strict	Logical. If TRUE (default), enforce strict TOON syntax.
allow_comments	Logical. If TRUE (default), allow # and // comments.
allow_duplicate_keys	Logical. If TRUE (default), allow duplicate keys.
warn	Logical. If TRUE (default), emit warnings.
simplify	Logical. If TRUE (default), simplify homogeneous batches.

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:
# Stream array items
toon_stream_items("items.toon",
  callback = function(batch) {
    cat("Got", length(batch), "items\n")
  }
)
## End(Not run)
```

---

toon_stream_rows	<i>Stream tabular TOON rows</i>
------------------	---------------------------------

---

## Description

Stream tabular TOON rows

## Usage

```
toon_stream_rows(
  file,
  key = NULL,
  callback,
  batch_size = 10000L,
  strict = TRUE,
  allow_comments = TRUE,
  allow_duplicate_keys = TRUE,
  warn = TRUE,
  col_types = NULL,
  ragged_rows = c("expand_warn", "error"),
  n_mismatch = c("warn", "error"),
  max_extra_cols = Inf
)
```

## Arguments

file	Character scalar. Path to TOON file.
key	Character scalar or NULL. If non-NULL, extract tabular array at root[key].
callback	Function. Called with each batch as a data.frame. Return value is ignored; exceptions propagate and abort parsing.
batch_size	Integer. Number of rows per batch (default 10000).
strict	Logical. If TRUE (default), enforce strict TOON syntax.
allow_comments	Logical. If TRUE (default), allow # and // comments.
allow_duplicate_keys	Logical. If TRUE (default), allow duplicate keys.
warn	Logical. If TRUE (default), emit warnings.
col_types	Named character vector specifying column types.
ragged_rows	Character. How to handle rows with different field counts.
n_mismatch	Character. How to handle declared row count mismatch.
max_extra_cols	Numeric. Maximum new columns allowed.

**Details**

Guarantees:

- Batches are independent and may be garbage collected
- Stable column order within stream

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:
# Stream large file in batches
toon_stream_rows("large.toon",
  callback = function(batch) {
    cat("Processing", nrow(batch), "rows\n")
  },
  batch_size = 10000
)

## End(Not run)
```

---

toon\_stream\_write\_rows

*Stream write tabular rows*

---

**Description**

Stream write tabular rows

**Usage**

```
toon_stream_write_rows(
  file,
  schema,
  row_source,
  batch_size = 10000L,
  indent = 2L
)
```

**Arguments**

file	Character scalar. Path to output file.
schema	Character vector of column names.
row_source	Function that returns next batch as data.frame, or NULL to end.
batch_size	Integer. Hint for batch size (passed to row_source).
indent	Integer. Number of spaces for indentation (default 2).

**Details**

Writes a tabular TOON array without holding all rows in memory.

**Value**

Invisibly returns the number of rows written.

**Examples**

```
## Not run:
# Stream write rows
i <- 0
row_source <- function() {
  i <<- i + 1
  if (i > 3) return(NULL)
  data.frame(x = i, y = i * 2)
}
toon_stream_write_rows("output.toon",
  schema = c("x", "y"),
  row_source = row_source
)

## End(Not run)
```

---

toon\_to\_csv

*Convert TOON to CSV*


---

**Description**

Convert TOON to CSV

**Usage**

```
toon_to_csv(
  path_toon,
  path_csv,
  key = NULL,
  strict = TRUE,
  allow_comments = TRUE,
  warn = TRUE
)
```

**Arguments**

path_toon	Character scalar. Path to input TOON file.
path_csv	Character scalar. Path to output CSV file.
key	Character scalar or NULL. If non-NULL, extract tabular array at root[key].

strict	Logical. If TRUE (default), enforce strict syntax.
allow_comments	Logical. If TRUE (default), allow comments.
warn	Logical. If TRUE (default), emit warnings.

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:
toon_to_csv("data.toon", "data.csv")

## End(Not run)
```

---

toon_to_feather	<i>Convert TOON to Feather</i>
-----------------	--------------------------------

---

**Description**

Convert TOON to Feather

**Usage**

```
toon_to_feather(
  path_toon,
  path_feather,
  key = NULL,
  strict = TRUE,
  allow_comments = TRUE,
  warn = TRUE
)
```

**Arguments**

path_toon	Character scalar. Path to input TOON file.
path_feather	Character scalar. Path to output Feather file.
key	Character scalar or NULL. If non-NULL, extract tabular array.
strict	Logical. If TRUE (default), enforce strict syntax.
allow_comments	Logical. If TRUE (default), allow comments.
warn	Logical. If TRUE (default), emit warnings.

**Details**

Requires the arrow package. If not installed, an error is thrown.

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:
toon_to_feather("data.toon", "data.feather")

## End(Not run)
```

---

toon_to_json	<i>Convert TOON to JSON</i>
--------------	-----------------------------

---

**Description**

Convert TOON to JSON

**Usage**

```
toon_to_json(toon, pretty = FALSE, strict = TRUE, allow_comments = TRUE)
```

**Arguments**

toon	Character scalar containing TOON.
pretty	Logical. If TRUE, use multi-line JSON formatting (default FALSE).
strict	Logical. If TRUE (default), enforce strict syntax.
allow_comments	Logical. If TRUE (default), allow comments in input.

**Details**

Requires the jsonlite package. If not installed, an error is thrown.

**Value**

Character scalar containing JSON.

**Examples**

```
## Not run:
json <- toon_to_json('name: "Alice"\nage: 30')

## End(Not run)
```

---

toon_to_parquet	<i>Convert TOON to Parquet</i>
-----------------	--------------------------------

---

### Description

Convert TOON to Parquet

### Usage

```
toon_to_parquet(  
  path_toon,  
  path_parquet,  
  key = NULL,  
  strict = TRUE,  
  allow_comments = TRUE,  
  warn = TRUE  
)
```

### Arguments

path_toon	Character scalar. Path to input TOON file.
path_parquet	Character scalar. Path to output Parquet file.
key	Character scalar or NULL. If non-NULL, extract tabular array.
strict	Logical. If TRUE (default), enforce strict syntax.
allow_comments	Logical. If TRUE (default), allow comments.
warn	Logical. If TRUE (default), emit warnings.

### Details

Requires the arrow package. If not installed, an error is thrown.

### Value

Invisibly returns NULL.

### Examples

```
## Not run:  
toon_to_parquet("data.toon", "data.parquet")  
  
## End(Not run)
```

---

validate_toon	<i>Validate TOON</i>
---------------	----------------------

---

**Description**

Validate TOON

**Usage**

```
validate_toon(  
  x,  
  is_file = FALSE,  
  strict = TRUE,  
  allow_comments = TRUE,  
  allow_duplicate_keys = TRUE  
)
```

**Arguments**

x	Character scalar containing TOON text, or file path if is_file=TRUE.
is_file	Logical. If TRUE, x is treated as a file path.
strict	Logical. If TRUE (default), enforce strict TOON syntax.
allow_comments	Logical. If TRUE (default), allow # and // comments.
allow_duplicate_keys	Logical. If TRUE (default), allow duplicate keys.

**Details**

Never throws unless there's an internal error (e.g., file unreadable). May warn for permissive recoveries.

**Value**

Logical scalar.

- TRUE if valid.
- FALSE if invalid, with attribute attr(result, "error") containing a structured error object with components: type, message, line, column, snippet, file.

**Examples**

```
# Valid TOON  
validate_toon('key: "value"')  
  
# Invalid TOON (returns FALSE with error attribute)  
result <- validate_toon('key: {invalid}')  
if (!result) print(attr(result, "error")$message)
```

---

validation	<i>TOON Validation and Formatting</i>
------------	---------------------------------------

---

**Description**

Functions for validating and formatting TOON data.

---

write_toon	<i>Write R object to TOON file</i>
------------	------------------------------------

---

**Description**

Write R object to TOON file

**Usage**

```
write_toon(x, file, pretty = TRUE, indent = 2L, strict = TRUE)
```

**Arguments**

x	R object to serialize.
file	Character scalar. Path to output file.
pretty	Logical. If TRUE (default), use multi-line formatting.
indent	Integer. Number of spaces for indentation (default 2).
strict	Logical. If TRUE (default), reject NaN/Inf values.

**Value**

Invisibly returns NULL.

**Examples**

```
## Not run:  
write_toon(list(x = 1, y = 2), "output.toon")  
  
## End(Not run)
```

---

write_toon_df	<i>Write data.frame to tabular TOON</i>
---------------	---

---

### Description

Write data.frame to tabular TOON

### Usage

```
write_toon_df(  
  df,  
  file,  
  tabular = TRUE,  
  pretty = TRUE,  
  indent = 2L,  
  strict = TRUE  
)
```

### Arguments

df	A data.frame to write.
file	Character scalar. Path to output file.
tabular	Logical. If TRUE (default), write as tabular TOON array.
pretty	Logical. If TRUE (default), use multi-line formatting.
indent	Integer. Number of spaces for indentation (default 2).
strict	Logical. If TRUE (default), reject NaN/Inf values.

### Value

Invisibly returns NULL.

### Examples

```
## Not run:  
# Write data.frame as tabular TOON  
write_toon_df(mtcars[1:3, 1:4], "cars.toon")  
  
## End(Not run)
```

# Index

[as\\_tabular\\_toon](#), 2  
[assert\\_toon](#), 3

[conditions](#), 4  
[conversions](#), 4  
[core-io](#), 4  
[csv\\_to\\_toon](#), 5

[feather\\_to\\_toon](#), 6  
[format\\_toon](#), 6  
[from\\_toon](#), 7

[json\\_to\\_toon](#), 8

[parquet\\_to\\_toon](#), 9  
[print.toon](#), 10

[read\\_toon](#), 10  
[read\\_toon\\_df](#), 11

[streaming](#), 12

[tabular-io](#), 13  
[to\\_toon](#), 13  
[toon\\_info](#), 14  
[toon\\_peek](#), 15  
[toon\\_stream\\_items](#), 15  
[toon\\_stream\\_rows](#), 17  
[toon\\_stream\\_write\\_rows](#), 18  
[toon\\_to\\_csv](#), 19  
[toon\\_to\\_feather](#), 20  
[toon\\_to\\_json](#), 21  
[toon\\_to\\_parquet](#), 22

[validate\\_toon](#), 23  
[validation](#), 24

[write\\_toon](#), 24  
[write\\_toon\\_df](#), 25