Package: xmlwriter (via r-universe)

May 27, 2026

Title Fast and Elegant XML Generation

Version 0.1.1.9000

Description Provides a fast and elegant interface for generating XML fragments and documents. It can be used in companion with R packages 'XML' or 'xml2' to generate XML documents. The fast XML generation is implemented using the 'Rcpp' package.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LinkingTo Rcpp

Imports Rcpp

Suggests xml2, tinytest

URL https://edwindj.github.io/xmlwriter/

BugReports https://github.com/edwindj/xmlwriter/issues

Repository https://edwindj.r-universe.dev

Date/Publication 2025-05-02 10:01:29 UTC

RemoteUrl https://github.com/edwindj/xmlwriter

RemoteRef HEAD

RemoteSha 87f69f1a9e2f6d71ff177dc9afcb86a0544da550

Contents

`xmlwriter-package` *Fast and elegant XML generation*

Description

`xmlwriter` is an R package that provides a fast and simple interface for creating XML documents and fragments from R. It has a simple elegant syntax for creating `xml_fragments`. `xmlwriter`'s XML generation from R lists is fast, implemented in C++ using `Rcpp`.

Details

`xmlwriter` can be used as a companion to R packages `XML` or `xml2` which are both wonderful packages optimized for parsing, querying and manipulating XML documents. Both `XML` and `xml2` provide several ways for creating XML documents, but they are not optimized for generating and writing XML.

Creating XML documents with `XML` and `xml2` can be a bit cumbersome, mostly because it forces the author to manipulate the XML document tree, obscuring the XML structure of the document, and making it hard to read the XML that is being generated. `xml2` does provide a way to create XML documents from R data structures using nested lists which is a powerful feature, but it is not optimized for speed or readability.

`xmlwriter` provides an intuitive interface for creating XML documents, that mimics how XML is written in a text editor. It has two different ways to create XML documents:

- a light weight R syntax using `tag()`, `frag()`, `+`, `/` and `data_frag()`, creating an `xml_fragment()`, that can be easily translated into a `character` or `xml2::xml_document` object, or be used as a flexible building block for generating a larger XML document.
- an `xmlbuilder()` object that allows you to create XML documents in a feed-forward manner, with `start` and `end` methods, giving you more control on the XML document structure, including XML comment, prolog etc.

It implements several `xml2` methods:

- `as_xml_document.xml_fragment()`
- `as_list.xml_fragment()`
- `write_xml.xml_fragment()`

Author(s)

Maintainer: Edwin de Jonge <edwindjonge@gmail.com> (ORCID)

See Also

Useful links:

- https://edwindj.github.io/xmlwriter/
- Report bugs at https://github.com/edwindj/xmlwriter/issues

Examples

```
doc <- xml_fragment(
  study = frag(
    .attr = c(id="1"),
    person = frag(
      .attr = c(id = "p1"),
      name = "John Doe",
      age = 30
    ),
    person = frag(
      name = "Jane Doe",
      age = 25,
      address = frag(street = "123 Main St", city = "Springfield"),
      "This is a text node"
    )
  )
)

print(doc)
if (require("xml2")){
  as_xml_document(doc)
}

# you can create a function to generate an xml fragment:
person_frag <- function(name, age, id){
  tag("person", id = id) > frag(
    name = name,
    age  = age,
    address = frag(
      street = "123 Main St",
      city = "Springfield"
    )
  )
}

# xml_doc is a xml_fragment with the restriction of having one root element
doc2 <- xml_doc("study") > (
  person_frag("John Doe", 30, "p1") +
  person_frag("Jane Doe", 25, "p2")
)
```

```
print(doc2)

if (require("xml2")){
  as_xml_document(doc2)
}

# a fragment can have multiple root elements
fgmt <- person_frag("John Doe", 30, id = "p1") +
  person_frag("Jane Doe", 25, id = "p2")


print(fgmt)

if (require("xml2")){
  # as_xml_document won't work because it expects a single root element,
  # so we retrieve a nodeset instead
  as_xml_nodeset(fgmt)
}

iris_xml <- xml_doc("fieldstudy", id = "iris", doi ="10.1111/j.1469-1809.1936.tb02137.x") /
  frag(
    source = "Fisher, R. A. (1936) The use of multiple measurements in
taxonomic problems. Annals of Eugenics, 7, Part II, 179-188.",
    data = data_frag(iris, row_tag = "obs")
  )

print(iris_xml, max_characters = 300)

if (require("xml2")){
  as_xml_document(iris_xml)
}
```

`add_child_fragment` *Add a child fragment to an existing xml_fragment*

Description

Add a child fragment to an existing xml_fragment. The child fragment can be a named `frag` element in which case the name is used as the tag name, an unnamed element in which case the element is added as a text node. This functionality is equivalent with the / and > operator.

Usage

```
add_child_fragment(x, ..., .frag = frag(...))
```

Arguments

x	an `xml_fragment()` object
...	nest named elements and characters to include in the fragment (see example)
.frag	an xml_fragment to add as a child, overrides the ... argument

Value

the original xml_fragment() with the child added.

See Also

Other xml_fragment: as.character.xml_fragment(), as_frag(), as_xml_nodeset(), data_frag(), frag(), xml_fragment()

---

as.character.xml_fragment

Turn an xml_fragment into a character

Description

This function turns an xml_fragment into a character string, using a performant c++ implementation.

Usage

```
## S3 method for class 'xml_fragment'
as.character(x, use_prolog = FALSE, ...)

## S3 method for class 'xml_doc'
as.character(x, use_prolog = TRUE, ...)
```

Arguments

x	object to be coerced or tested.
use_prolog	if TRUE the xml prolog with be included. To suppress the prolog string either remove set use_prolog = FALSE.
...	further arguments passed to or from other methods.

Value

a character with the xml representation of the fragment.

See Also

Other xml_fragment: add_child_fragment(), as_frag(), as_xml_nodeset(), data_frag(), frag(), xml_fragment()

as_frag *Convert a list to an xml fragment*

Description

As frag is a helper function to convert a named list to an xml fragment, it transforms all values to character, and recursively transforms nested lists. `as_frag` can be used for flexible list created xml fragments, names of a list turn into tags, and values into text nodes.

Usage

```
as_frag(x, ..., .attr = list(...))
```

Arguments

`x`	named list that will be transformed into a fragment
`...`	optional attributes to be set on the parent of the fragment
`.attr`	a list of attributes to add to the parent of the fragment, overrides the `...` argument

Value

`xml_fragment()` object as if specified with `frag()`.

See Also

Other xml_fragment: `add_child_fragment()`, `as.character.xml_fragment()`, `as_xml_nodeset()`, `data_frag()`, `frag()`, `xml_fragment()`

as_xml_nodeset *Transforms an xml_fragment into an xml_nodeset*

Description

Using the `xml2` package, this function transforms an `xml_fragment` into an `xml_nodeset`

Usage

```
as_xml_nodeset(x, ...)
```

Arguments

`x`	an object created with `xml_fragment()`
`...`	reserved for future use

Value

an xml2::xml_nodeset object

See Also

Other xml_fragment: add_child_fragment(), as.character.xml_fragment(), as_frag(), data_frag(), frag(), xml_fragment()

Other xml2: list_as_xml_document(), list_as_xml_string()

---

data_frag *Create an xml_fragment from a data.frame*

Description

Create a xml_fragment() from a data.frame, in which each row is a set of xml elements (columns).

Usage

```
data_frag(df, row_tags = "row", .attr = NULL)
```

Arguments

df	data frame that will be stored as set of xml elements
row_tags	character the tag name that is used for each row. Note that this can be a single value or a vector of length equal to the number of rows in the data.frame.
.attr	optional data.frame with xml row attributes

Value

xml_fragment() object

See Also

Other xml_fragment: add_child_fragment(), as.character.xml_fragment(), as_frag(), as_xml_nodeset(), frag(), xml_fragment()

Examples

```
persons <- data.frame(
  name = c("John Doe", "Jane Doe"),
  age = c(30, 25),
  stringsAsFactors = FALSE
)

df <- data_frag(persons, row_tag = "person")
print(df)
```

```
# setting ids on rows
persons <- data.frame(
  name = c("John Doe", "Jane Doe"),
  age = c(30, 25),
  id = c("p1", "p2"),
  stringsAsFactors = FALSE
)

df <- data_frag(
  persons[c("name", "age")],
  row_tag = "person",
  .attr = persons[c("id")]
)

print(df)

# turning it into a document
doc <-
  xml_doc("study", id = "1") / frag(
  source = "homeless db",
  data = df
)

cat(as.character(doc))
```

`elem` *add an element to an xmlbuilder object*

Description

add an element to an xmlbuilder object

Usage

```
elem(tag, text = NULL, ...)
```

Arguments

`tag`	name of element
`text`	text contents of element
`...`	additional xml. attributes to be set

Value

an xmlbuilder object

Examples

```
xb <- elem("homeless") /
  elem("person") / (
     elem("name","John Doe") +
     elem("age",35)
  ) +
  elem("person") /(
    elem("name","Jane Doe") +
    elem("age", 30)
  ) +
  elem("person") / (
    elem("name","Jim Doe") +
    elem("age", 25) +
    elem("address") / (
      elem("street", "123 Main St") +
      elem("city", "Anytown") +
      elem("state", "CA") +
      elem("zip", 12345)
    )
  )

print(xb)
xb$end()
xb$end()


doc <- xb |> xml2::as_xml_document()
doc |> as.character() |> cat()
```

`frag` *Create a frag xml_fragment*

Description

Create a `frag` xml_fragment, that allows for multiple elements and nested `frag`s.

Usage

```
frag(..., .attr = NULL)
```

Arguments

`...`	nest named elements and characters to include in the fragment (see example)
`.attr`	a list of attributes to add to the parent of the fragment

Value

an `xml_fragment()` object

See Also

Other xml_fragment: add_child_fragment(), as.character.xml_fragment(), as_frag(), as_xml_nodeset(), data_frag(), xml_fragment()

Examples

```
doc <- xml_fragment(
  study = frag(
    .attr = c(id="1"),
    person = frag(
      .attr = c(id = "p1"),
      name = "John Doe",
      age = 30
    ),
    person = frag(
      name = "Jane Doe",
      age = 25,
      address = frag(street = "123 Main St", city = "Springfield"),
      "This is a text node"
    )
  )
)

print(doc)
if (require("xml2")){
  as_xml_document(doc)
}

# you can create a function to generate an xml fragment:
person_frag <- function(name, age, id){
  tag("person", id = id) > frag(
    name = name,
    age  = age,
    address = frag(
      street = "123 Main St",
      city = "Springfield"
    )
  )
}

# xml_doc is a xml_fragment with the restriction of having one root element
doc2 <- xml_doc("study") > (
  person_frag("John Doe", 30, "p1") +
  person_frag("Jane Doe", 25, "p2")
)

print(doc2)

if (require("xml2")){
  as_xml_document(doc2)
}
```

```
# a fragment can have multiple root elements
fgmt <- person_frag("John Doe", 30, id = "p1") +
  person_frag("Jane Doe", 25, id = "p2")


print(fgmt)

if (require("xml2")){
  # as_xml_document won't work because it expects a single root element,
  # so we retrieve a nodeset instead
  as_xml_nodeset(fgmt)
}

iris_xml <- xml_doc("fieldstudy", id = "iris", doi ="10.1111/j.1469-1809.1936.tb02137.x") /
  frag(
    source = "Fisher, R. A. (1936) The use of multiple measurements in
taxonomic problems. Annals of Eugenics, 7, Part II, 179-188.",
    data = data_frag(iris, row_tag = "obs")
  )

print(iris_xml, max_characters = 300)

if (require("xml2")){
  as_xml_document(iris_xml)
}
```

`list_as_xml_document` *Convert a list to an xml_document*

Description

`list_as_xml_document` is fast and efficient way to convert a list to an `xml2::xml_document`. The preferred interface is to use `xml_fragment()` and `xml_doc()` to create xml fragments.

Usage

```
list_as_xml_document(x, ...)
```

Arguments

`x`	a list as returned by `xml2::as_list()`
`...`	reserved for future use

Details

`list_to_xml_document` is a much faster implementation of `xml2::as_xml_document.list()` method. It writes the xml directly to a string buffer and then reads it back into an `xml2::xml_document`.

The function can be used in tandem with `xml2::as_list()` to convert R data structures.

Value

an xml2::xml_document

See Also

Other xml2: as_xml_nodeset(), list_as_xml_string()

Examples

```
data <-
  list(
    study = list(
      person = list(
        name = "John Doe",
        age = "30"
      ),
      person = list(
        name = "Jane Doe",
        age = "25"
      )
    )
  )

list_as_xml_string(data)
if (require("xml2")){
  list_as_xml_document(data)
}

#note the xml_fragment function is more powerful to create lists

data <- xml_doc("study", id = "1") /
  frag(
    person = frag(
      name = "John Doe",
      age = "30"
    ),
    person = frag(
      name = "Jane Doe",
      age = "25"
    ),
    "This is a text node"
)

list_as_xml_string(data)
```

---

list_as_xml_string *Convert a list to an xml string*

Description

`list_to_xml_string` is fast and efficient way to convert a specific list to an xml string. The preferred interface is to use `xml_fragment()` and `xml_doc()` to create xml fragments.

Usage

```
list_as_xml_string(x, use_prolog = FALSE, ...)
```

Arguments

`x`	a list as returned by `xml2::as_list()`
`use_prolog`	logical, should the xml declaration be included in the output?
`...`	reserved for future use

Details

This function is the working horse for turning `xml_fragment()`, `xml_doc()` and list object into `character` xml strings and `xml2::xml_document` objects.

The input list format is identical to the format returned by `xml2::as_list()` function, but much faster in generating an xml string from it. It writes the xml directly to a string buffer.

This function allows for easy conversion of R data structures into xml format by creating the list structures in R and then converting them to xml. The function can be used in tandem with `xml2::as_list()` to convert R data structures.

Value

a character string with the xml representation of the list

See Also

Other xml2: `as_xml_nodeset()`, `list_as_xml_document()`

Examples

```
data <-
  list(
    study = list(
      person = list(
        name = "John Doe",
        age = "30"
      ),
      person = list(
        name = "Jane Doe",
        age = "25"
      )
    )
  )

list_as_xml_string(data)
```

```
if (require("xml2")){
  list_as_xml_document(data)
}

#note the xml_fragment function is more powerful to create lists

data <- xml_doc("study", id = "1") /
  frag(
    person = frag(
      name = "John Doe",
      age = "30"
    ),
    person = frag(
      name = "Jane Doe",
      age = "25"
    ),
    "This is a text node"
)

list_as_xml_string(data)
```

`read_fragment` *Read an XML fragment from a string*

Description

Reads a xml fragment from a string, a connection or a raw vector using `xml2::read_xml()`, and turns it into a `xml_fragment()`.

Usage

```
read_fragment(x, ...)
```

Arguments

`x`	A string, a connection or a raw vector
`...`	passed to `xml2::read_xml()`

Value

an object of class `xml_fragment`

tag *Create a tag fragment*

Description

Create a tag fragment with optional text and attributes

Usage

```
tag(tag, text = NULL, ..., .attr = list(...))
```

Arguments

tag	character, the name of the tag
text	character, the text to include in the tag
...	additional attributes to add to the tag
.attr	a list of additional attributes to add to the tag, overrides the ... argument

Value

an xml_fragment with the new tag added

Examples

```
tag("greeting", "hi", id = "hi")

tag("person", id = "1") / (tag("name", "John Doe") + tag("age", 35))

xml_fragment(person = frag(
  .attr = c(id = 1),
  name = "John Doe",
  age = 30
))   / tag("address", "Unknown")


a <- tag("person", id = 1) /
  xml_fragment(
    name ="John Doe",
    age = 30,
    address = frag(
      street = "123 Main St",
      city = "Springfield"
    )
  )

cat(as.character(a))
```

xml_doc *Create an xml_fragment with a root element, (kind of tag)*

Description

Create an xml_fragment with a root element, (kind of tag)

Usage

```
xml_doc(root, ..., .attr = list(...))
```

Arguments

root	the name of the root element
...	additional attributes to add to the tag
.attr	a list of additional attributes to add to the tag, overrides the ... argument

Value

an xml_fragment with the root element

Examples

```
tag("greeting", "hi", id = "hi")

tag("person", id = "1") / (tag("name", "John Doe") + tag("age", 35))

xml_fragment(person = frag(
  .attr = c(id = 1),
  name = "John Doe",
  age = 30
))   / tag("address", "Unknown")


a <- tag("person", id = 1) /
  xml_fragment(
    name ="John Doe",
    age = 30,
    address = frag(
      street = "123 Main St",
      city = "Springfield"
    )
  )

cat(as.character(a))
```

`xml_fragment` *Create an XML fragment*

Description

Create an XML fragment using readable R syntax, that can be used to create a string, an `xml2::xml_document` or as a building block for more complex XML documents.

Usage

```
xml_fragment(...)
```

Arguments

`...`	nest named elements and characters to include in the fragment (see example)

Details

An `xml_fragment` is built using:

- named `frag` elements, each name is a tag name, and the value is the contents of the tag, e.g. `name = "value"` becomes `<name>value</name>`. The value can be a nested `frag` object, a character string or a numeric value.
- `.attr` attributes, which is set on current element, or on the `frag` where it is specified
- unnamed elements, which are added as text nodes.
- `data_frag()` function that can be used to convert a data.frame to an xml fragment, in which each row is a set of xml elements (columns).
- `tag()` function that can be used to create a tag with attributes and (optional) text.

An `xml_doc` is a special case of an `xml_fragment` that contains exactly one root element, and errors when this is not the case.

A resulting `xml_fragment` object can be converted to an `xml2::xml_document` with `xml2::as_xml_document()` or to a character string with `as.character()`. Both methods are fast using a performant c++ implementation.

Value

an `xml_fragment`, list object that can be converted to an `xml2::xml_document` or `character` string

See Also

Other xml_fragment: `add_child_fragment()`, `as.character.xml_fragment()`, `as_frag()`, `as_xml_nodeset()`, `data_frag()`, `frag()`

Examples

```
doc <- xml_fragment(
  study = frag(
    .attr = c(id="1"),
    person = frag(
      .attr = c(id = "p1"),
      name = "John Doe",
      age = 30
    ),
    person = frag(
      name = "Jane Doe",
      age = 25,
      address = frag(street = "123 Main St", city = "Springfield"),
      "This is a text node"
    )
  )
)

print(doc)
if (require("xml2")){
  as_xml_document(doc)
}

# you can create a function to generate an xml fragment:
person_frag <- function(name, age, id){
  tag("person", id = id) > frag(
    name = name,
    age  = age,
    address = frag(
      street = "123 Main St",
      city = "Springfield"
    )
  )
}

# xml_doc is a xml_fragment with the restriction of having one root element
doc2 <- xml_doc("study") > (
  person_frag("John Doe", 30, "p1") +
  person_frag("Jane Doe", 25, "p2")
)

print(doc2)

if (require("xml2")){
  as_xml_document(doc2)
}

# a fragment can have multiple root elements
fgmt <- person_frag("John Doe", 30, id = "p1") +
  person_frag("Jane Doe", 25, id = "p2")
```

```
print(fgmt)

if (require("xml2")){
  # as_xml_document won't work because it expects a single root element,
  # so we retrieve a nodeset instead
  as_xml_nodeset(fgmt)
}

iris_xml <- xml_doc("fieldstudy", id = "iris", doi ="10.1111/j.1469-1809.1936.tb02137.x") /
  frag(
    source = "Fisher, R. A. (1936) The use of multiple measurements in
taxonomic problems. Annals of Eugenics, 7, Part II, 179-188.",
    data = data_frag(iris, row_tag = "obs")
  )

print(iris_xml, max_characters = 300)

if (require("xml2")){
  as_xml_document(iris_xml)
}
```

`xmlbuilder` *Create a fast feed-forward XML builder*

Description

This function creates an XML builder that allows you to create XML documents in a feed-forward manner. `xmlbuilder` returns an object that has methods to create XML elements, text nodes, comments, and more.

Usage

```
xmlbuilder(
  allow_fragments = TRUE,
  use_prolog = !allow_fragments,
  strict = FALSE
)
```

Arguments

`allow_fragments`	logical. Should a warning be issued if the XML document has multiple root elements? Set to `FALSE` to suppress when creating multiple xml fragments.
`use_prolog`	logical. Should the XML prolog be included in the output? Default is `TRUE`, which generate an UTF-8 xml prolog. Set to `FALSE` if you want to generate an xml fragment or manually prepend the prolog.
`strict`	logical. Should the builder check for dangling nodes, default is `FALSE`.

Details

- `$start(tag, ...)` (or `$start_element`) starts an element with a given tag and attributes.
- `$end()` (or `$end_element`) ends the current element.
- `$element(tag, text, ...)` creates an element with a given tag, text, and attributes.
- `$text(text)` creates a text node.
- `$fragment(..., .attr)` writes an xml fragment to the.
- `$comment(comment)` creates a comment node.
- `$to_xml_string()` returns the XML document or fragments(s) as a character vector.

Value

An object of class 'xmlbuilder

Examples

```
b <-xmlbuilder()

b$start("root")
  b$element("child1", "text1", attr1 = "value1")
  b$element("child2", "text2", attr2 = "value2")
  b$start("child3", attr3 = "value3")
    b$text("text3")
    b$element("child4", "text3", attr4 = "value4")
  b$end("child3")
b$end("root")

print(b)

if (require("xml2")) {
  # a builder can be converted to an xml_document using
  doc <- as_xml_document(b)

  # or equivalentlty
  doc <-
    b$to_xml_string() |>
    read_xml()
}

# build some xml fragments
fms <- xmlbuilder(allow_fragments = TRUE)

fms$start("person", id = "1")
  fms$element("name", "John Doe")
  fms$element("age", 30)
fms$end("person")

fms$start("person", id = "2")
  fms$element("name", "Jane Doe")
  fms$element("age", 25)
fms$end("person")
```

```
fms$start("person", id = "3")
  fms$element("name", "Jim Doe")
  fms$element("age", 35)
fms$end("person")

s <- fms$to_xml_string()
as.character(fms)
length(s) # three fragments

# print xml string of the second fragment
print(s[2])

if (require("xml2")){
  # convert to xml_nodes
  nodes <- fms$to_xml_node_list()
  length(nodes) # three nodes
  # show the second xml_node
  print(nodes[[2]])
}

# use fragments
xb <- xmlbuilder()

xb$start("study")
xb$fragment(
  person = frag(
    name = "John Doe",
    age = 30
  ),
  person = frag(
    name = "Jane Doe",
    age = 25
  )
)
xb$end("study")
xb
```

Index