Package 'selectr'

May 9, 2026

Type Package

Title Translate CSS Selectors to XPath Expressions

Version 0.5-1

License BSD_3_clause + file LICENCE

Depends R (>= 3.0)

Imports methods, stringr, R6

Suggests testthat, XML, xml2

URL https://sjp.co.nz/projects/selectr/

BugReports https://github.com/sjp/selectr/issues

Description Translates a CSS selector into an equivalent XPath expression. This allows us to use CSS selectors when working with the XML package as it can only evaluate XPath expressions. Also provided are convenience functions useful for using CSS selectors on XML nodes. This package is a port of the Python package 'cssselect' (<https://cssselect.readthedocs.io/>).

NeedsCompilation no

Author Simon Potter [aut, trl, cre],
Simon Sapin [aut],
Ian Bicking [aut]

Maintainer Simon Potter <simon@sjp.co.nz>

Repository CRAN

Date/Publication 2025-12-17 09:00:02 UTC

Contents

`css_to_xpath` *Translate a CSS selector to an equivalent XPath expression.*

Description

This function aims to create an XPath expression equivalent to what would be matched by the given CSS selector. The reason the translation is required is because the XML and xml2 packages, being a libxml2 wrappers, can only evaluate XPath expressions.

Using this function, it is possible to search an XML tree without the prerequisite of knowing XPath.

Usage

```
css_to_xpath(selector,
             prefix = "descendant-or-self::",
             translator = "generic")
```

Arguments

`selector`	A character vector of CSS selectors.
`prefix`	The prefixes to apply to the resulting XPath expressions. The default or `""` are most commonly used.
`translator`	The type of translator that will be used. Possible options are `generic` (the default), or `html` or `xhtml`.

Details

Each selector given to this function will be translated to an equivalent XPath expression. The resulting XPath expression can be given a prefix which determines the scope of the expression. The default prefix determines the scope to be the node itself and all descendants of the node. Most commonly the prefix is either the default or `""`, unless it is known what scope a particular XPath expression should have.

The translator used is usually unnecessary to specify as the default is sufficient for most cases. However, it is of use when creating expressions relating to (X)HTML pseudo elements and languages. In particular it qualifies the following pseudo selectors to apply only to relevant (X)HTML elements: `:checked`, `:disabled`, `:enabled` and `:link`.

When the translator is set to `html`, all elements and attributes will be converted to lower case. This restriction is removed when the translator is `xhtml` (or the default `generic` translator).

Value

A character vector of XPath expressions.

Author(s)

Simon Potter

References

CSS Selectors Level 4 https://www.w3.org/TR/selectors-4/, XPath https://www.w3.org/TR/xpath/.

Examples

```
css_to_xpath(".testclass")
css_to_xpath("#testid", prefix = "")
css_to_xpath("#testid .testclass")
css_to_xpath(":checked", translator = "html")
```

`querySelectorAll` *Find nodes that match a group of CSS selectors in an XML tree.*

Description

The purpose of these functions is to mimic the functionality of the `querySelector` and `querySelectorAll` functions present in Internet browsers. This is so we can succinctly query an XML tree for nodes matching a CSS selector.

Namespaced functions `querySelectorNS` and `querySelectorAllNS` are also provided to search relative to a given namespace.

Usage

```
querySelector(doc, selector, ns = NULL, ...)
querySelectorAll(doc, selector, ns = NULL, ...)
querySelectorNS(doc, selector, ns,
                prefix = "descendant-or-self::", ...)
querySelectorAllNS(doc, selector, ns,
                   prefix = "descendant-or-self::", ...)
```

Arguments

`doc`	The XML document or node to be evaluated against.
`selector`	A selector used to query `doc`. This must be a single character string.
`ns`	The namespace that the query will be filtered to. This is a named list or vector which has as its name a namespace, and its value is the namespace URI. This can be ignored for the un-namespaced functions.
`prefix`	The prefix to apply to the resulting XPath expression. The default or `""` are most commonly used.
`...`	Parameters to be passed onto `css_to_xpath`.

Details

The querySelectorNS and querySelectorAllNS functions are convenience functions for working with namespaced documents. They filter out all content that does not belong within the given namespaces. Note that when searching for particular elements in a selector, they must have a namespace prefix, e.g. "svg|g".

The namespace argument, ns, is simply passed on to getNodeSet or xml_find_all if it is necessary to use a namespace present within the document. This can be ignored for content lacking a namespace, which is usually the case when using querySelector or querySelectorAll.

Value

For querySelector, the result is a single node that represents the first matched node from a selector. If no matching nodes are found, NULL is returned.

For querySelectorAll, the result is a list of XML nodes. This list may be empty in the case that no match is found.

The querySelectorNS and querySelectorAllNS functions return the same type of content as their un-namespaced counterparts.

Author(s)

Simon Potter

References

CSS Selectors Level 4 https://www.w3.org/TR/selectors-4/, XPath https://www.w3.org/TR/xpath/, querySelectorAll https://developer.mozilla.org/en-US/docs/DOM/Document.querySelectorAll and https://www.w3.org/TR/selectors-api/#interface-definitions.

Examples

```
hasXML <- require(XML)
hasxml2 <- require(xml2)

if (!hasXML && !hasxml2)
  return() # can't demo without XML or xml2 packages present

parseFn <- if (hasXML) xmlParse else read_xml
# Demo for working with the XML package (if present, otherwise xml2)
exdoc <- parseFn('<a><b class="aclass"/><c id="anid"/></a>')
querySelector(exdoc, "#anid")   # Returns the matching node
querySelector(exdoc, ".aclass") # Returns the matching node
querySelector(exdoc, "b, c")    # First match from grouped selection
querySelectorAll(exdoc, "b, c") # Grouped selection
querySelectorAll(exdoc, "b")    # A list of length one
querySelector(exdoc, "d")       # No match
querySelectorAll(exdoc, "d")    # No match

# Read in a document where two namespaces are being set:
# SVG and MathML
```

```
svgdoc <- parseFn(system.file("demos/svg-mathml.svg",
                              package = "selectr"))
# Search for <script/> elements in the SVG namespace
querySelectorNS(svgdoc, "svg|script",
                c(svg = "http://www.w3.org/2000/svg"))
querySelectorAllNS(svgdoc, "svg|script",
                   c(svg = "http://www.w3.org/2000/svg"))
# MathML content is *within* SVG content,
# search for <mtext> elements within the MathML namespace
querySelectorNS(svgdoc, "math|mtext",
                c(math = "http://www.w3.org/1998/Math/MathML"))
querySelectorAllNS(svgdoc, "math|mtext",
                   c(math = "http://www.w3.org/1998/Math/MathML"))
# Search for *both* SVG and MathML content
querySelectorAllNS(svgdoc, "svg|script, math|mo",
                   c(svg = "http://www.w3.org/2000/svg",
                     math = "http://www.w3.org/1998/Math/MathML"))

if (!hasXML)
  return() # already demo'd xml2

# Demo for working with the xml2 package
exdoc <- read_xml('<a><b class="aclass"/><c id="anid"/></a>')
querySelector(exdoc, "#anid")   # Returns the matching node
querySelector(exdoc, ".aclass") # Returns the matching node
querySelector(exdoc, "b, c")    # First match from grouped selection
querySelectorAll(exdoc, "b, c") # Grouped selection
querySelectorAll(exdoc, "b")    # A list of length one
querySelector(exdoc, "d")       # No match
querySelectorAll(exdoc, "d")    # No match

# Read in a document where two namespaces are being set:
# SVG and MathML
svgdoc <- read_xml(system.file("demos/svg-mathml.svg",
                               package = "selectr"))
# Search for <script/> elements in the SVG namespace
querySelectorNS(svgdoc, "svg|script",
                c(svg = "http://www.w3.org/2000/svg"))
querySelectorAllNS(svgdoc, "svg|script",
                   c(svg = "http://www.w3.org/2000/svg"))
# MathML content is *within* SVG content,
# search for <mtext> elements within the MathML namespace
querySelectorNS(svgdoc, "math|mtext",
                c(math = "http://www.w3.org/1998/Math/MathML"))
querySelectorAllNS(svgdoc, "math|mtext",
                   c(math = "http://www.w3.org/1998/Math/MathML"))
# Search for *both* SVG and MathML content
querySelectorAllNS(svgdoc, "svg|script, math|mo",
                   c(svg = "http://www.w3.org/2000/svg",
                     math = "http://www.w3.org/1998/Math/MathML"))
```

Index