Package: W3CMarkupValidator (via r-universe)

June 11, 2026

Version 0.2-4

Title R Interface to W3C Markup Validation Services

Description R interface to a W3C Markup Validation service. See <https://validator.w3.org/> for more information.

Depends R (>= 3.2.0)

Imports curl, utils, tools, parallel, jsonlite

License GPL-2

NeedsCompilation no

Author Kurt Hornik [aut, cre] (ORCID: <https://orcid.org/0000-0003-4198-9911>)

Maintainer Kurt Hornik <Kurt.Hornik@R-project.org>

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

Date/Publication 2026-06-11 11:52:03 UTC

RemoteUrl https://github.com/cran/W3CMarkupValidator

RemoteRef HEAD

RemoteSha 9b0c727c06bb3e6bf9c253efbb8a7ec20bf19c2d

Contents

`inspect` *Inspect* R *objects*

Description

Display R objects in a convenient and informative way.

Usage

```
inspect(x, ...)
## S3 method for class 'w3c_markup_validate'
inspect(x, details = TRUE, ...)
## S3 method for class 'w3c_markup_validate_db'
inspect(x, details = TRUE, full = FALSE, ...)
```

Arguments

`x`	an R object for the generic; objects inheriting from the respective classes for the methods.
`details`	a logical recycled to length two indicating whether to display detailed information on errors and warnings, respectively, or a character vector with elements partially matching 'error' or 'warning'.
`full`	a logical indicating whether to provide information about validation results with no errors or warnings.
`...`	arguments to be passed to and from methods.

Details

`inspect()` is a generic function.

The methods for objects inheriting from `"w3c_markup_validate"` or `"w3c_markup_validate_db"` (single results of markup validation using `w3c_markup_validate`, or collections of such results) conveniently summarize the problems found by the validation service as collections of tables with columns giving the line, column and a description of the problem.

`w3c_markup_validate` *Validate Markup of Web Documents using W3C Markup Validation Services*

Description

Check the markup validity of web documents in HTML, XHTML, etc., using a W3C Markup Validation service.

Usage

```
w3c_markup_validate(baseurl = w3c_markup_validate_baseurl(),
                    uri = NULL, file = NULL, string = NULL,
                    opts = list(), jar = NULL, concordance = FALSE)
```

Arguments

baseurl	a character string giving the URL of the W3C Markup Validation service to employ.
uri	a character string giving the URI to validate.
file	a character string giving the path of a file to validate.
string	a character string with the markup to validate.
opts	a named list with options to use for accessing the validation service.
jar	a character string giving the path to a 'vnu.jar' for direct local validation, or TRUE (see 'Details').
concordance	a logical indicating whether to look for R concordance information for mapping output to source locations.

Details

Exactly one of uri, file or string must be given.

Validation is then (unless jar is given) performed using the *new* W3C Markup Validation service at the given URL, as appropriate for programmatic checking of modern HTML (i.e., HTML5) documents. See https://validator.w3.org/docs/api.html for more information.

(Versions of **W3CMarkupValidator** up to 0.1-7 used the obsolete SOAP 1.2 API for the Markup Validator, which cannot handle HTML5.)

If the validation requests was successful (i.e., a 200 OK response status was returned), w3c_markup_validate() returns the information in the response organized into an object of class "w3c_markup_validate", which is a data frame with with rows giving the diagnostic messages and the following variables:

type a character string giving the type of the message: one of "info", "error" or "document-error".

subType a character string giving the subtype of the message. For type "info" this can be "warning" or missing; for type "error" this can be "fatal" or missing.

firstLine, firstColumn, lastLine, lastColumn integers indicating the range of source code associated with the message.

message a character string giving the diagnostic message text.

extract a character string representing an extract of the document source from around the point in source designated for the message by the line and column numbers.

If problems were found and concordance information is used and found, then variables srcFile and srcLine are added giving the mapping from output to source locations obtained by matchConcordance().

Alternatively, one can also use the underling Nu HTML checker directly (via invoking the Java interpreter) using a local 'vnu.jar' JAR file by giving the path to this file in the jar argument. See https://validator.github.io/validator/ for more information. Using the latest version of 'vnu.jar' from https://github.com/validator/validator/releases/tag/latest works

best. Installing package **vnu.jar** from the repository at https://datacube.wu.ac.at conveniently provides the JAR file, see the examples. Using `jar = TRUE` will use that JAR file.

This class has methods for `print` for compactly summarizing the results, and an `inspect` method for inspecting details.

Note

The validation service provided by the W3C used by default for validation is a shared and free resource, and the W3C asks (see https://validator.w3.org/docs/api.html) for considerate use and possibly installing a local instance of the validation service: excessive use of the service will be blocked. See https://validator.w3.org/docs/install.html for setting up a local service.

On Debian-based systems, for a long time a local instance could conveniently be installed via the system command `apt-get install w3c-markup-validator` and following the instructions for providing the validator as a web service. Unfortunately, as of 2025-08 this package is orphaned and still provides the old SOAP service which cannot handle modern HTML documents.

One can use the environment variable `W3C_MARKUP_VALIDATOR_BASEURL` to specify the service to be employed by default.

See Also

`w3c_markup_validate_baseurl` for getting and setting the URL of the validation service.

`w3c_markup_validate_db` for combining and analyzing collections of single validation results.

Examples

```
## Not much to show with this as it should validate ok
## (provided that the validation service is accessible):
tryCatch(w3c_markup_validate(uri = "https://CRAN.R-project.org"),
         error = identity)
## Use a local 'vnu.jar' provided by the 'vnu.jar' package from the
## repository at <https://datacube.wu.ac.at>, installed via
##   install.packages("vnu.jar", repos = "https://datacube.wu.ac.at/",
##                    type = "source")
## to install.  To be on the safe side:
if(nzchar(jar <- system.file("java", "vnu.jar", package = "vnu.jar"))) {
    file <- system.file("examples", "file2_novalid.html",
                        package = "W3CMarkupValidator")
    results <- w3c_markup_validate(file = file, jar = jar)
    results
}
## Alternatively, with the 'vnu.jar' package installed, one can simply
## use w3c_markup_validate(file = file, jar = TRUE).
```

`w3c_markup_validate_baseurl`

URL of W3C Markup Validation Service

Description

Get or set the URL of the W3C Markup Validation service to employ.

Usage

```
w3c_markup_validate_baseurl(new)
```

Arguments

`new`	a character string with the URL of the the W3C Markup Validation service to employ, or `NULL` indicating to use the W3C service as specified by the environment variable `W3C_MARKUP_VALIDATOR_BASEURL`, or if this is unset, at https://validator.w3.org/nu/?.

Details

If no argument is given, the current URL is returned. Otherwise, the URL is set to the given one or (if `new` is `NULL`) the default one.

`w3c_markup_validate_db`

Collections of W3C Markup Validation Results

Description

Create and manipulate collections of W3C markup validation results.

Usage

```
w3c_markup_validate_db(x, names = NULL)

w3c_markup_validate_files(files,
                          baseurl = w3c_markup_validate_baseurl(),
                          opts = list(),
                          jar = NULL,
                          concordance = FALSE,
                          verbose = interactive(),
                          Ncpus = getOption("Ncpus", 1L))

w3c_markup_validate_uris(uris,
```

```
                    baseurl = w3c_markup_validate_baseurl(),
                    opts = list(),
                    jar = NULL,
                    concordance = FALSE,
                    verbose = interactive(),
                    Ncpus = getOption("Ncpus", 1L))
```

Arguments

x	a list of w3c_markup_validate results.
names	a character vector of names for the elements in x, or NULL (default), indicating to use the names of x or to auto-generate names if these are NULL.
files	a character vector giving the names of files to validate.
uris	a character vector giving URIs to validate.
baseurl	a character string giving the URL of the W3C Markup Validation service to employ.
opts, jar, concordance	see w3c_markup_validate.
verbose	a logical indicating if some "progress report" should be given.
Ncpus	the number of parallel processes to use for validating in parallel.

Details

w3c_markup_validate_db() creates a db (data base) of w3c_markup_validate results as a list of these results with class "w3c_markup_validate_db". This class has methods for print and c for compactly summarizing and combining results, an inspect method for inspecting details, and an as.data.frame method for collapsing the errors and warnings into a "flat" data frame useful for further analyses.

w3c_markup_validate_files() and w3c_markup_validate_uris() validate the markup in the given files or URIs, with results combined into such results db objects. For files or URIs for which validation failed (which can happen for example when these contain characters invalid in SGML), the corresponding error condition objects are gathered into the "failures" attribute of the results db returned.

See Also

w3c_markup_validate_baseurl for getting and setting the URL of the validation service.

Examples

```
## Test files provided with this package:
dir <- system.file("examples", package = "W3CMarkupValidator")
files <- Sys.glob(file.path(dir, "*.html"))
if(!startsWith(w3c_markup_validate_baseurl(),
               "https://validator.w3.org")) {
    ## Validate.
    results <- w3c_markup_validate_files(files)
    results
```

```
    ## In case of failures, inspect the error messages:
    lapply(attr(results, "failures"), conditionMessage)
    ## Inspect validation results:
    inspect(results)
    inspect(results, full = TRUE)
    ## Turn results into a data frame:
    df <- as.data.frame(results)
    ## Tabulate error messages:
    table(substring(df$message, 1L, 60L))
    ## Inspect a particular set of error messages:
    df[startsWith(df$message, "Unclosed element"), ]
    ## Conveniently view the full records (modulo HTML markup):
    write.dcf(df)
}
```

Index