

# Package ‘adaR’

February 8, 2024

**Title** A Fast 'WHATWG' Compliant URL Parser

**Version** 0.3.2

**Description** A wrapper for 'ada-url', a 'WHATWG' compliant and fast URL parser written in modern 'C++'. Also contains auxiliary functions such as a public suffix extractor.

**URL** <https://gesistsa.github.io/adaR/>, <https://github.com/gesistsa/adaR>

**BugReports** <https://github.com/gesistsa/adaR/issues>

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**LinkingTo** Rcpp

**Imports** Rcpp, triebeard

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Depends** R (>= 4.2)

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** David Schoch [aut, cre] (<<https://orcid.org/0000-0003-2952-4812>>),  
Chung-hong Chan [aut] (<<https://orcid.org/0000-0002-6232-7530>>),  
Yagiz Nizipli [ctb, cph] (author of ada-url :  
<<https://github.com/ada-url/ada>>),  
Daniel Lemire [ctb, cph] (author of ada-url :  
<<https://github.com/ada-url/ada>>)

**Maintainer** David Schoch <david@schochastics.net>

**Repository** CRAN

**Date/Publication** 2024-02-08 21:40:02 UTC

## R topics documented:

ada_clear_port . . . . .	2
ada_get_href . . . . .	3
ada_has_credentials . . . . .	4
ada_set_href . . . . .	5
ada_url_parse . . . . .	6
public_suffix . . . . .	7
url_decode2 . . . . .	7
<b>Index</b>	<b>9</b>

---

ada_clear_port	<i>Clear a specific component of URL</i>
----------------	--

---

### Description

These functions clears a specific component of URL.

### Usage

```
ada_clear_port(url, decode = TRUE)
ada_clear_hash(url, decode = TRUE)
ada_clear_search(url, decode = TRUE)
```

### Arguments

url	character. one or more URL to be parsed
decode	logical. Whether to decode the output (see <a href="#">utils::URLdecode()</a> ), default to TRUE

### Value

character, NA if not a valid URL

### Examples

```
url <- "https://user_1:password_1@example.org:8080/dir/./api?q=1#frag"
ada_clear_port(url)
ada_clear_hash(url)
ada_clear_search(url)
```

---

ada_get_href	<i>Get a specific component of URL</i>
--------------	--

---

### Description

These functions get a specific component of URL.

### Usage

```
ada_get_href(url, decode = TRUE)
ada_get_username(url, decode = TRUE)
ada_get_password(url, decode = TRUE)
ada_get_port(url, decode = TRUE)
ada_get_hash(url, decode = TRUE)
ada_get_host(url, decode = TRUE)
ada_get_hostname(url, decode = TRUE)
ada_get_pathname(url, decode = TRUE)
ada_get_search(url, decode = TRUE)
ada_get_protocol(url, decode = TRUE)
ada_get_domain(url, decode = TRUE)
ada_get_basename(url)
```

### Arguments

url	character. one or more URL to be parsed
decode	logical. Whether to decode the output (see <a href="#">utils::URLdecode()</a> ), default to TRUE

### Value

character, NA if not a valid URL

### Examples

```
url <- "https://user_1:password_1@example.org:8080/dir/./api?q=1#frag"
ada_get_href(url)
```

```
ada_get_username(url)
ada_get_password(url)
ada_get_port(url)
ada_get_hash(url)
ada_get_host(url)
ada_get_hostname(url)
ada_get_pathname(url)
ada_get_search(url)
ada_get_protocol(url)
ada_get_domain(url)
ada_get_basename(url)
## these functions are vectorized
urls <- c("http://www.google.com", "http://www.google.com:80", "noturl")
ada_get_port(urls)
```

---

ada\_has\_credentials    *Check if URL has a certain component*

---

### **Description**

These functions check if URL has a certain component.

### **Usage**

```
ada_has_credentials(url)
ada_has_empty_hostname(url)
ada_has_hostname(url)
ada_has_non_empty_username(url)
ada_has_non_empty_password(url)
ada_has_port(url)
ada_has_hash(url)
ada_has_search(url)
```

### **Arguments**

url                    character. one or more URL to be parsed

### **Value**

logical, NA if not a valid URL.

**Examples**

```
url <- c("https://user_1:password_1@example.org:8080/dir/./api?q=1#frag")
ada_has_credentials(url)
ada_has_empty_hostname(url)
ada_has_hostname(url)
ada_has_non_empty_username(url)
ada_has_non_empty_password(url)
ada_has_port(url)
ada_has_hash(url)
ada_has_search(url)
## these functions are vectorized
urls <- c("http://www.google.com", "http://www.google.com:80", "noturl")
ada_has_port(urls)
```

---

ada\_set\_href

*Set a specific component of URL*

---

**Description**

These functions set a specific component of URL.

**Usage**

```
ada_set_href(url, input, decode = TRUE)
ada_set_username(url, input, decode = TRUE)
ada_set_password(url, input, decode = TRUE)
ada_set_port(url, input, decode = TRUE)
ada_set_host(url, input, decode = TRUE)
ada_set_hostname(url, input, decode = TRUE)
ada_set_pathname(url, input, decode = TRUE)
ada_set_protocol(url, input, decode = TRUE)
ada_set_search(url, input, decode = TRUE)
ada_set_hash(url, input, decode = TRUE)
```

**Arguments**

url	character. one or more URL to be parsed
input	character. containing new component for URL. Vector of length 1 or same length as url.

decode            logical. Whether to decode the output (see `utils::URLdecode()`), default to TRUE

### Value

character, NA if not a valid URL

### Examples

```
url <- "https://user_1:password_1@example.org:8080/dir/./api?q=1#frag"
ada_set_href(url, "https://google.de")
ada_set_username(url, "user_2")
ada_set_password(url, "hunter2")
ada_set_port(url, "1234")
ada_set_hash(url, "#section1")
ada_set_host(url, "example.de")
ada_set_hostname(url, "example.de")
ada_set_pathname(url, "path/")
ada_set_search(url, "q=2")
ada_set_protocol(url, "ws:")
```

---

ada\_url\_parse            *Use ada-url to parse a url*

---

### Description

Use ada-url to parse a url

### Usage

```
ada_url_parse(url, decode = TRUE)
```

### Arguments

url            character. one or more URL to be parsed

decode        logical. Whether to decode the output (see `utils::URLdecode()`), default to TRUE

### Details

For details on the returned components refer to the introductory vignette.

### Value

A data frame of the url components: href, protocol, username, password, host, hostname, port, pathname, search, and hash

### Examples

```
ada_url_parse("https://user_1:password_1@example.org:8080/dir/./api?q=1#frag")
```

---

public_suffix	<i>Extract the public suffix from a vector of domains or hostnames</i>
---------------	--

---

**Description**

Extract the public suffix from a vector of domains or hostnames

**Usage**

```
public_suffix(domains)
```

**Arguments**

domains            character. vector of domains or hostnames

**Value**

public suffixes of domains as character vector

**Examples**

```
public_suffix("http://example.com")

# doesn't work for general URLs
public_suffix("http://example.com/path/to/file")

# extracting hostname first does the trick
public_suffix(ada_get_hostname("http://example.com/path/to/file"))
```

---

url_decode2	<i>Function to percent-decode characters in URLs</i>
-------------	--

---

**Description**

Similar to [utils::URLdecode](#)

**Usage**

```
url_decode2(url)
```

**Arguments**

url                a character vector

**Value**

percent decoded URLs as character vector

**Examples**

```
url_decode2("Hello%20World")
```



# Index

ada\_clear\_hash (ada\_clear\_port), 2  
ada\_clear\_port, 2  
ada\_clear\_search (ada\_clear\_port), 2  
ada\_get\_basename (ada\_get\_href), 3  
ada\_get\_domain (ada\_get\_href), 3  
ada\_get\_hash (ada\_get\_href), 3  
ada\_get\_host (ada\_get\_href), 3  
ada\_get\_hostname (ada\_get\_href), 3  
ada\_get\_href, 3  
ada\_get\_password (ada\_get\_href), 3  
ada\_get\_pathname (ada\_get\_href), 3  
ada\_get\_port (ada\_get\_href), 3  
ada\_get\_protocol (ada\_get\_href), 3  
ada\_get\_search (ada\_get\_href), 3  
ada\_get\_username (ada\_get\_href), 3  
ada\_has\_credentials, 4  
ada\_has\_empty\_hostname  
    (ada\_has\_credentials), 4  
ada\_has\_hash (ada\_has\_credentials), 4  
ada\_has\_hostname (ada\_has\_credentials),  
    4  
ada\_has\_non\_empty\_password  
    (ada\_has\_credentials), 4  
ada\_has\_non\_empty\_username  
    (ada\_has\_credentials), 4  
ada\_has\_port (ada\_has\_credentials), 4  
ada\_has\_search (ada\_has\_credentials), 4  
ada\_set\_hash (ada\_set\_href), 5  
ada\_set\_host (ada\_set\_href), 5  
ada\_set\_hostname (ada\_set\_href), 5  
ada\_set\_href, 5  
ada\_set\_password (ada\_set\_href), 5  
ada\_set\_pathname (ada\_set\_href), 5  
ada\_set\_port (ada\_set\_href), 5  
ada\_set\_protocol (ada\_set\_href), 5  
ada\_set\_search (ada\_set\_href), 5  
ada\_set\_username (ada\_set\_href), 5  
ada\_url\_parse, 6  
  
public\_suffix, 7  
  
url\_decode2, 7  
utils::URLdecode, 7  
utils::URLdecode(), 2, 3, 6