

# Package: webutils (via r-universe)

November 2, 2024

**Type** Package

**Title** Utility Functions for Developing Web Applications

**Version** 1.2.2

**Description** Parses http request data in application/json, multipart/form-data, or application/x-www-form-urlencoded format. Includes example of hosting and parsing html form data in R using either 'httpuv' or 'Rhttpd'.

**License** MIT + file LICENSE

**URL** <https://jeroen.r-universe.dev/webutils>

**BugReports** <https://github.com/jeroen/webutils/issues>

**Imports** curl (>= 2.5), jsonlite

**Suggests** httpuv, testthat

**RoxygenNote** 7.3.2.9000

**Language** en-US

**Roxygen** list(markdown = TRUE)

**Encoding** UTF-8

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

**RemoteUrl** <https://github.com/jeroen/webutils>

**RemoteRef** HEAD

**RemoteSha** cd2ba42240a8ab23d7ae57fef87d2c81fff9b0e

## Contents

demo_httpuv . . . . .	2
demo_rhttpd . . . . .	2
parse_http . . . . .	3
parse_multipart . . . . .	3
parse_query . . . . .	4

<b>Index</b>	<b>5</b>
--------------	----------

demo\_httpuv

*Demo multipart parser with httpuv*

---

**Description**

Starts the httpuv web server and hosts a simple form including a file upload to demo the multipart parser.

**Usage**

```
demo_httpuv(port = 9359)
```

**Arguments**

port                    which port number to run the http server

**See Also**

Other demo: [demo\\_rhttpd\(\)](#)

---

demo\_rhttpd

*Demo multipart parser with rhttpd*

---

**Description**

Starts the Rhttpd web server and hosts a simple form including a file upload to demo the multipart parser.

**Usage**

```
demo_rhttpd()
```

**See Also**

Other demo: [demo\\_httpuv\(\)](#)

---

parse\_http                    *Parse http request*

---

### Description

Parse the body of a http request, based on the Content-Type request header. Currently supports the three most important content types: application/x-www-form-urlencoded with [parse\\_query\(\)](#), multipart/form-data with [parse\\_multipart\(\)](#), and application/json with [jsonlite::fromJSON\(\)](#).

### Usage

```
parse_http(body, content_type, ...)
```

### Arguments

body	request body of the http request
content_type	content-type http request header as specified by the client
...	additional arguments passed to parser function

### Examples

```
# Parse json encoded payload:
parse_http('{"foo":123, "bar":true}', 'application/json')

# Parse url-encoded payload
parse_http("foo=1%2B1%3D2&bar=yin%26yang", "application/x-www-form-urlencoded")

## Not run: use demo app to parse multipart/form-data payload
demo_rhttpd()

## End(Not run)
```

---

parse\_multipart                *Parse a multipart/form-data request*

---

### Description

Parse a multipart/form-data request, which is usually generated from a HTML form submission. The parameters can include both text values as well as binary files. They can be distinguished from the presence of a filename attribute.

### Usage

```
parse_multipart(body, boundary)
```

**Arguments**

body                body of the HTTP request. Must be raw or character vector.  
 boundary            boundary string as specified in the Content-Type request header.

**Details**

A multipart/form-data request consists of a single body which contains one or more values plus meta-data, separated using a boundary string. This boundary string is chosen by the client (e.g. the browser) and specified in the Content-Type header of the HTTP request. There is no escaping; it is up to the client to choose a boundary string that does not appear in one of the values.

The parser is written in pure R, but still pretty fast because it uses the regex engine.

**Examples**

```
## Not run: example form
demo_rhttpd()

## End(Not run)
```

---

parse_query	<i>Parse query string</i>
-------------	---------------------------

---

**Description**

Parse http parameters from a query string. This includes unescaping of url-encoded values.

**Usage**

```
parse_query(query)
```

**Arguments**

query                a url-encoded query string

**Details**

For http GET requests, the query string is specified in the URL after the question mark. For http POST or PUT requests, the query string can be used in the request body when the Content-Type header is set to application/x-www-form-urlencoded.

**Examples**

```
q <- "foo=1%2B1%3D2&bar=yin%26yang"
parse_query(q)
```

# Index

## \* **demo**

demo\_httpuv, 2

demo\_rhttpd, 2

demo\_httpuv, 2, 2

demo\_rhttpd, 2, 2

jsonlite::fromJSON(), 3

parse\_http, 3

parse\_multipart, 3

parse\_multipart(), 3

parse\_query, 4

parse\_query(), 3