Package 'htmxr'

March 4, 2026

Title Build Modern Web Applications with 'htmx' and 'plumber2'

Version 0.1.1

Description A lightweight framework for building server-driven web applications in 'R'. 'htmxr' combines the simplicity of 'htmx' for partial page updates with the power of 'plumber2' for non-blocking HTTP endpoints. Build interactive dashboards and data applications without writing 'JavaScript', using familiar 'R' patterns inspired by 'Shiny'. For more information on 'htmx', see <https://htmx.org>.

License MIT + file LICENSE

URL https://hyperverse-r.github.io/htmxr/,
https://github.com/hyperverse-r/htmxr

BugReports https://github.com/hyperverse-r/htmxr/issues

Depends R (>= 4.1.0)

Imports htmltools, plumber2

Suggests dplyr, ggplot2, knitr, pak, purrr, rmarkdown, svglite, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Arthur Bréant [aut, cre]

Maintainer Arthur Bréant <arthur@thinkr.fr>

Repository CRAN

Date/Publication 2026-03-04 10:30:02 UTC

Contents

`htmxr_is_htmx` *Detect if a request comes from htmx*

Description

Checks whether the incoming HTTP request was made by htmx by inspecting the `HX-Request` header. htmx sends this header with every AJAX request.

Usage

```
htmxr_is_htmx(request)
```

Arguments

`request`	A request object (e.g. from a plumber2 handler). Must have a `headers` element — a named list or character vector of HTTP headers (lowercase keys, as provided by plumber2).

Value

`TRUE` if the request was made by htmx, `FALSE` otherwise.

Examples

```
# Simulated htmx request
req <- list(headers = list(`hx-request` = "true"))
htmxr_is_htmx(req)

# Regular request
req <- list(headers = list())
htmxr_is_htmx(req)
```

hx_button *Button element*

Description

Creates a <button> element with optional htmx attributes.

Usage

```
hx_button(
  label,
  id = NULL,
  class = NULL,
  get = NULL,
  post = NULL,
  target = NULL,
  swap = NULL,
  trigger = NULL,
  indicator = NULL,
  swap_oob = NULL,
  confirm = NULL,
  ...
)
```

Arguments

label	Button label (text or HTML content).
id	Optional element id.
class	Optional CSS class(es).
get	URL for hx-get.
post	URL for hx-post.
target	CSS selector for hx-target.
swap	Swap strategy for hx-swap.
trigger	Trigger specification for hx-trigger.
indicator	CSS selector for hx-indicator.
swap_oob	Out-of-band swap targets for hx-swap-oob.
confirm	Confirmation message for hx-confirm.
...	Additional HTML attributes passed to the <button> element.

Value

An htmltools::tags object.

Examples

```
# Simple button
hx_button("Click me")

# Button with htmx GET request
hx_button("Load data", get = "/api/data", target = "#result")

# Button with confirmation
hx_button("Delete", post = "/api/delete", confirm = "Are you sure?")
```

---

hx_head *Specify additional head elements for an htmxr page*

Description

Wraps tags to be included in the page head when passed to hx_page().

Usage

```
hx_head(..., title = "htmxr page")
```

Arguments

...	tags to include in the head (stylesheets, scripts, meta, etc.)
title	page title

Value

A list with class hx_head, to be passed to hx_page().

Examples

```
hx_head(title = "My app")

hx_head(
  title = "My app",
  tags$link(rel = "stylesheet", href = "/style.css")
)
```

hx_page *Generate a complete HTML page with htmx*

Description

Generate a complete HTML page with htmx

Usage

```
hx_page(..., lang = "en", html_attrs = list())
```

Arguments

`...`	page content. Use `hx_head()` to add elements to the head.
`lang`	language code for the `<html>` element (default `"en"`).
`html_attrs`	a named list of additional attributes to set on the `<html>` element (e.g. `list("data-theme" = "cupcake")` for DaisyUI).

Value

A length-one character string containing the full HTML document (including `<!DOCTYPE html>`), ready to be served as an HTTP response.

Examples

```
hx_page(tags$h1("Hello, htmxr!"))

hx_page(
  hx_head(title = "My app"),
  tags$p("Hello, world!")
)
```

hx_run_example *Run an htmxr example*

Description

Launches an example API that demonstrates htmxr features. Call `hx_run_example()` without arguments to list available examples.

Usage

```
hx_run_example(example = NULL, port = 8080)
```

Arguments

example	name of the example to run. If NULL, lists available examples.
port	port to run the API on.

Value

Called for side effects. When example is NULL, returns the available example names invisibly. Otherwise does not return (the server blocks).

Examples

```
hx_run_example() # list available examples
if (interactive()) {
  hx_run_example("hello") # run the hello example
}
```

---

hx_select_input *Select input*

Description

Creates a <select> element with optional htmx attributes. When label is provided, the input is wrapped in a containing a <label> element linked via the for attribute.

Usage

```
hx_select_input(
  id,
  label = NULL,
  choices,
  selected = NULL,
  multiple = FALSE,
  name = id,
  class = NULL,
  get = NULL,
  post = NULL,
  target = NULL,
  swap = NULL,
  trigger = NULL,
  indicator = NULL,
  swap_oob = NULL,
  confirm = NULL,
  ...
)
```

Arguments

id	Element id. Also used as name by default.
label	Optional label text. When provided, the input is wrapped in a with a <label>.
choices	Named or unnamed character vector of choices. If unnamed, values are used as labels. If named, names are used as labels and values as option values (same convention as Shiny).
selected	Optional value(s) to pre-select.
multiple	Logical. If TRUE, adds the multiple attribute to allow multi-selection.
name	Form field name. Defaults to id.
class	Optional CSS class(es) for the <select> element.
get	URL for hx-get.
post	URL for hx-post.
target	CSS selector for hx-target.
swap	Swap strategy for hx-swap.
trigger	Trigger specification for hx-trigger.
indicator	CSS selector for hx-indicator.
swap_oob	Out-of-band swap targets for hx-swap-oob.
confirm	Confirmation message for hx-confirm.
...	Additional HTML attributes passed to the <select> element.

Value

An htmltools::tags object.

Examples

```
# Simple select without label
hx_select_input("cut", choices = c("Fair", "Good", "Ideal"))

# Select with label and named choices
hx_select_input(
  "cut",
  label = "Filter by cut:",
  choices = c("All" = "all", "Fair", "Good", "Ideal"),
  selected = "all"
)

# Select with htmx attributes
hx_select_input(
  "cut",
  label = "Filter by cut:",
  choices = c("All" = "all", "Fair", "Good"),
  get = "/rows",
  trigger = "change",
```

```
  target = "#tbody"
)
```

`hx_serve_assets` *Serve htmxr static assets*

Description

Configures a plumber2 API to serve htmxr's static assets (htmx JavaScript library) at `/htmxr/assets/`.

Usage

```
hx_serve_assets(api)
```

Arguments

`api`	a plumber2 API object

Value

the API object (modified, for piping)

Examples

```
plumber2::api() |>
  hx_serve_assets()
```

`hx_set` *Add htmx attributes to any HTML tag*

Description

A generic modifier that appends htmx attributes to an existing htmltools::tags object. Works with any HTML element.

Usage

```
hx_set(
  tag,
  get = NULL,
  post = NULL,
  target = NULL,
  swap = NULL,
  trigger = NULL,
  indicator = NULL,
```

```
  swap_oob = NULL,
  confirm = NULL
)
```

Arguments

tag	An htmltools::tags object to modify.
get	URL for hx-get.
post	URL for hx-post.
target	CSS selector for hx-target.
swap	Swap strategy for hx-swap.
trigger	Trigger specification for hx-trigger.
indicator	CSS selector for hx-indicator.
swap_oob	Out-of-band swap targets for hx-swap-oob.
confirm	Confirmation message for hx-confirm.

Value

The input tag with htmx attributes appended.

Examples

```
tags$div(id = "plot") |>
  hx_set(get = "/plot", trigger = "load", target = "#plot", swap = "innerHTML")

hx_set(
  tags$div(id = "result", class = "container"),
  get = "/data",
  trigger = "load"
)
```

---

hx_slider_input *Slider input*

Description

Creates an <input type="range"> element with optional htmx attributes. When label is provided, the input is wrapped in a containing a <label> element linked via the for attribute.

Usage

```
hx_slider_input(
  id,
  label = NULL,
  value = 50,
  min = 0,
  max = 100,
  step = 1,
  name = id,
  class = NULL,
  get = NULL,
  post = NULL,
  target = NULL,
  swap = NULL,
  trigger = NULL,
  indicator = NULL,
  swap_oob = NULL,
  confirm = NULL,
  ...
)
```

Arguments

id	Element id. Also used as name by default.
label	Optional label text. When provided, the input is wrapped in a with a <label>.
value	Initial value (default 50).
min	Minimum value (default 0).
max	Maximum value (default 100).
step	Step increment (default 1).
name	Form field name. Defaults to id.
class	Optional CSS class(es) for the <input> element.
get	URL for hx-get.
post	URL for hx-post.
target	CSS selector for hx-target.
swap	Swap strategy for hx-swap.
trigger	Trigger specification for hx-trigger.
indicator	CSS selector for hx-indicator.
swap_oob	Out-of-band swap targets for hx-swap-oob.
confirm	Confirmation message for hx-confirm.
...	Additional HTML attributes passed to the <input> element.

Value

An htmltools::tags object.

Examples

```
# Simple slider
hx_slider_input("bins", label = "Number of bins:", min = 1, max = 50)

# Slider with htmx attributes
hx_slider_input(
  "bins",
  label = "Number of bins:",
  value = 30, min = 1, max = 50,
  get = "/plot",
  trigger = "input changed delay:300ms",
  target = "#plot"
)
```

---

hx_table *Table with htmx-powered tbody*

Description

Builds a complete <table> element with a <thead> and a <tbody>. htmx attributes are applied to the <tbody>, making it the swap target. When data is NULL (the default), the <tbody> is empty and its content is loaded lazily via htmx (e.g. trigger = "load").

Usage

```
hx_table(
  columns,
  data = NULL,
  id = NULL,
  col_labels = NULL,
  col_classes = NULL,
  class = NULL,
  thead_class = NULL,
  get = NULL,
  post = NULL,
  target = NULL,
  swap = NULL,
  trigger = NULL,
  indicator = NULL,
  swap_oob = NULL,
  confirm = NULL,
  ...
)
```

Arguments

columns	Character vector of column names to display. Defines the <thead> structure (required).
data	Optional data frame. If provided, rows are rendered in the <tbody> via hx_table_rows(). If NULL, the <tbody> is empty.
id	id attribute applied to the <tbody>.
col_labels	Labels for the <thead>. If NULL, column names are used as-is. Can be a named vector (c(price = "Price ($)")) to override specific columns, or an unnamed positional vector to replace all labels.
col_classes	Named list of CSS classes for <td> cells, passed to hx_table_rows() when data is provided.
class	CSS class(es) for the <table> element.
thead_class	CSS class(es) for the <thead> element.
get	URL for hx-get (applied to <tbody>).
post	URL for hx-post (applied to <tbody>).
target	CSS selector for hx-target (applied to <tbody>).
swap	Swap strategy for hx-swap (applied to <tbody>).
trigger	Trigger specification for hx-trigger (applied to <tbody>).
indicator	CSS selector for hx-indicator (applied to <tbody>).
swap_oob	Out-of-band swap targets for hx-swap-oob (applied to <tbody>).
confirm	Confirmation message for hx-confirm (applied to <tbody>).
...	Additional HTML attributes passed to the <table> element.

Value

An htmltools::tags object (<table>).

Examples

```
# Lazy-load table (empty tbody, content loaded on trigger)
hx_table(
  columns = c("cut", "color", "price"),
  col_labels = c("Cut", "Color", "Price"),
  id = "tbody",
  get = "/rows",
  trigger = "load",
  swap = "innerHTML"
)

# Table with data pre-rendered
df <- data.frame(cut = c("Fair", "Good"), price = c(326L, 400L))
hx_table(columns = c("cut", "price"), data = df)
```

hx_table_rows *Table rows fragment*

Description

Converts a data frame into a tagList of <tr> elements, one per row. Designed to be used as a fragment endpoint response — the output replaces a <tbody> via htmx swap.

Usage

```
hx_table_rows(data, columns = NULL, col_classes = NULL)
```

Arguments

data	A data frame.
columns	Character vector of column names to include (and their order). If NULL, all columns are used.
col_classes	Named list of CSS classes to add to <td> elements, keyed by column name. Example: list(price = "text-end fw-bold").

Value

A htmltools::tagList of <tr> tags.

Examples

```
df <- data.frame(cut = c("Fair", "Good"), price = c(326L, 400L))
hx_table_rows(df, columns = c("cut", "price"))

# With CSS classes on specific columns
hx_table_rows(df, col_classes = list(price = "text-end fw-bold"))
```

Index