

Package ‘DT2’

September 20, 2026

Type Package

Title 'DataTables' 2.x for R

Version 0.1.3

Date 2026-09-19

Description A modern R binding for 'DataTables' V2 with modular extension loading, 'Bootstrap 5' styling, 'Shiny' integration (proxy, events, inline inputs), server-side processing helpers, and standalone (non-Shiny) support. Configure 'DataTables' options directly via R lists, a 1:1 mapping to the 'JavaScript' API.

License MIT + file LICENSE

URL <https://github.com/StrategicProjects/DT2>,
<https://strategicprojects.github.io/DT2/>

Encoding UTF-8

Depends R (>= 4.1)

Imports htmlwidgets (>= 1.6.0), htmltools, jsonlite, cli

Suggests shiny, rlang, dplyr, lubridate, tibble, bslib, knitr,
rmarkdown, testthat (>= 3.1.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/Needs/website tidyverse/tidytemplate

NeedsCompilation no

Author Andre Leite [aut, cre],
Marcos Wasilew [aut],
Hugo Vasconcelos [aut],
Carlos Amorim [aut],
Diogo Bezerra [aut]

Maintainer Andre Leite <leite@castlab.org>

Repository CRAN

Date/Publication 2026-09-20 01:10:02 UTC

Contents

dt2	3
dt2_bind_server	5
dt2_buttons	6
dt2_check_updates	7
dt2_cols_align	8
dt2_cols_escape	8
dt2_cols_hide	9
dt2_cols_html	9
dt2_cols_render_js	10
dt2_cols_render_orthogonal	10
dt2_cols_width	11
dt2_col_button	12
dt2_col_checkbox	12
dt2_col_template	13
dt2_draw	13
dt2_extensions	14
dt2_format_datetime	14
dt2_format_number	15
dt2_format_number_abbrev	16
dt2_format_time_format	17
dt2_format_time_relative	17
dt2_language	18
dt2_length_menu	19
dt2_order	19
dt2_output	20
dt2_proxy	20
dt2_proxy_order	21
dt2_proxy_page	21
dt2_proxy_search	22
dt2_register_renderer	22
dt2_replace_data	23
dt2_search_global	23
dt2_select_rows	24
dt2_ssp_handler	24
dt2_state	25
dt2_theme	26
dt2_update_libs	27
dt2_use_buttons	28
dt2_use_renderer	29
observe_dt2_events	30
render_dt2	30

Index

Description

The main function for creating interactive DataTables. Works standalone (R Markdown, Quarto, Viewer) and inside Shiny.

Styling is controlled directly via theme, striped, hover, compact, font_scale – or a CSS class string for full control.

DataTables configuration goes in options (1:1 mapping to the JavaScript API). The two concerns are cleanly separated.

Usage

```
dt2(
  data,
  theme = "default",
  striped = NULL,
  hover = NULL,
  compact = NULL,
  font_scale = NULL,
  style = NULL,
  class = NULL,
  button_class = NULL,
  responsive = TRUE,
  options = list(),
  extensions = NULL,
  width = "100%",
  height = NULL,
  elementId = NULL
)
```

Arguments

data	A data.frame, tibble, or matrix.
theme	A theme preset name ("default", "clean", "minimal", "compact") or a dt2_theme() object. Default: "default".
striped, hover, compact	Logical; override the theme. NULL (default) = use theme value.
font_scale	Numeric; override the theme font-scale. NULL (default) = use theme value.
style	Styling framework: "bootstrap5" (default) or "core".
class	Optional CSS class string (e.g., "table table-dark"). If provided, overrides all theme-generated classes.
button_class	CSS class for Buttons extension buttons. Default: "btn btn-sm btn-outline-secondary". Examples: "btn btn-sm btn-primary", "btn btn-sm btn-outline-dark".

responsive	Logical; enable the Responsive extension so the table fills 100\ Default: TRUE. Set FALSE to disable.
options	List of DataTables options. See https://datatables.net/ref/core/option/ .
extensions	Character vector of extensions to load (e.g., c("Buttons", "Select")). Auto-detected from options when NULL.
width, height	CSS dimensions.
elementId	Optional HTML element ID.

Value

An htmlwidget object.

Examples

```
# Just works – beautiful defaults
dt2(iris)

# Override style inline
dt2(iris, striped = FALSE)
dt2(iris, font_scale = 0.85, compact = FALSE)

# Theme presets
dt2(iris, theme = "minimal")
dt2(iris, theme = "compact")

# Reusable theme
my_theme <- dt2_theme("clean", compact = TRUE)
dt2(iris, theme = my_theme)

# Override a preset
dt2(iris, theme = "minimal", striped = TRUE)

# CSS class override (power users)
dt2(iris, class = "table table-bordered table-dark")

# DataTables options (separate from styling)
dt2(iris, options = list(pageLength = 5, searching = FALSE))

# Disable responsive (fixed-width columns)
dt2(iris, responsive = FALSE)

# Everything composes
dt2(mtcars,
  theme = "clean",
  compact = TRUE,
  options = list(pageLength = 25))

# Buttons
dt2(mtcars, options = list(
  buttons = list("copy", "csv", "excel"),
```

```

    layout = list(topEnd = "buttons")
  ))

  # Custom button style
  dt2(mtcars,
    button_class = "btn btn-sm btn-primary",
    options = list(
      buttons = list("copy", "csv", "excel"),
      layout = list(topEnd = "buttons")
    )
  )

```

dt2_bind_server	<i>Bind a DataTables v2 server-side endpoint to a widget id</i>
-----------------	---

Description

Bind a DataTables v2 server-side endpoint to a widget id

Usage

```

dt2_bind_server(
  id,
  data,
  session = shiny::getDefaultReactiveDomain(),
  handler = NULL,
  rows_all = TRUE
)

```

Arguments

id	Output id of the widget (e.g., "tbl").
data	A data.frame with the source data.
session	Shiny session (default: current).
handler	Optional custom handler function(data, req) -> list(...). A custom handler may include dt2_rows_all / dt2_rows_current (1-based row indices) in its result to feed input\$<id>_rows_all and input\$<id>_rows_current; see dt2_ssp_handler() .
rows_all	Passed to dt2_ssp_handler() when handler is NULL: whether the default handler ships the filtered row indices to the client.

Value

No return value, called for side effects. Registers a Shiny observer on session that responds to client-side server-processing requests for the given widget id.

dt2_buttons	<i>Configure DataTables Buttons and (optionally) move them to a custom container</i>
-------------	--

Description

Configure DataTables Buttons and (optionally) move them to a custom container

Usage

```
dt2_buttons(
  options = list(),
  buttons = c("copyHtml5", "csvHtml5", "excelHtml5", "pdfHtml5", "print"),
  target = NULL
)
```

Arguments

options	A DT2 options list you are building.
buttons	Character vector with button names (e.g. "copyHtml5", "csvHtml5", "excelHtml5", "pdfHtml5", "print"). You can also pass a list with full button objects.
target	Optional CSS selector (e.g. "#btn-slot" or ".my-toolbar") to receive the buttons container. If provided, DT2 will move the rendered buttons to that container after init.

Details

Requires the **Buttons** extension. For CSV/Excel/PDF you also need **JSZip** and **pdfMake** (incl. vfs_fonts).

Prefer [dt2_use_buttons\(\)](#) for the common case: it takes simple button ids, styles them with a CSS class, and places them in the layout. Use `dt2_buttons()` when you need full button objects or want to move the rendered buttons into a custom container via `target`.

Value

The modified options list.

See Also

[dt2_use_buttons\(\)](#)

dt2_check_updates	<i>Check for DataTables library updates</i>
-------------------	---

Description

Queries the npm registry to compare installed library versions against the latest available versions. Version constraints are enforced to prevent incompatible major version upgrades (e.g. jQuery 3.x will not jump to 4.x).

Usage

```
dt2_check_updates(quiet = FALSE)
```

Arguments

quiet Logical. If TRUE, returns the result invisibly without printing. Default FALSE.

Value

A data.frame (invisibly) with columns: library, installed, latest, latest_ok, constraint, status.

Status values:

"ok" Library is up to date.

"UPDATE" A compatible update is available.

"PINNED" A new major version exists, but is blocked by the version constraint. The library is up to date within its allowed range.

"error" Lookup failed (check your internet connection).

Examples

```
dt2_check_updates()

# programmatic use
updates <- dt2_check_updates(quiet = TRUE)
updates[updates$status == "UPDATE", ]
```

dt2_cols_align	<i>Column align (Bootstrap 5 classes)</i>
----------------	---

Description

Column align (Bootstrap 5 classes)

Usage

```
dt2_cols_align(options = list(), cols, align = c("left", "center", "right"))
```

Arguments

- options Options list.
- cols Names or 1-based indices.
- align "left","center","right".

Value

Updated options.

dt2_cols_escape	<i>Escape/unescape columns content</i>
-----------------	--

Description

Controls whether cell content is HTML-escaped before display.

Usage

```
dt2_cols_escape(options = list(), cols, escape = TRUE)
```

Arguments

- options Options list.
- cols Names or indices.
- escape If TRUE (default), HTML special characters are escaped so the raw text is shown. If FALSE, the content is rendered as raw HTML (use with care; only for trusted content).

Value

Updated options.

dt2_cols_hide	<i>Hide columns</i>
---------------	---------------------

Description

Hide columns

Usage

```
dt2_cols_hide(options = list(), cols)
```

Arguments

options	Options list.
cols	Names or 1-based indices.

Value

Updated options.

dt2_cols_html	<i>Allow raw HTML rendering via columns.render</i>
---------------	--

Description

Mark columns to render raw HTML using a JS render function.

Usage

```
dt2_cols_html(options = list(), cols, js_render)
```

Arguments

options	Options list.
cols	Names or 1-based indices.
js_render	JS function (via htmlwidgets::JS) with signature (data, type, row, meta) returning a string of HTML when type == "display".

Value

Updated options.

dt2_cols_render_js	<i>Attach a raw JS render function to columns</i>
--------------------	---

Description

Provide a custom JS renderer for one or more columns. Use this when you need fine control over `columns.render`, including returning different outputs based on type (display/sort/filter/type).

Usage

```
dt2_cols_render_js(options = list(), col_specs, js_render)
```

Arguments

options	List returned, with columnDefs appended.
col_specs	Column names or indices.
js_render	A <code>htmlwidgets::JS()</code> function of signature <code>function(data, type, row, meta) { ... }</code> .

Value

Modified options.

See Also

<https://datatables.net/ref/core/option/columns.render>

dt2_cols_render_orthogonal	<i>Orthogonal render (display/sort/filter/type) per column</i>
----------------------------	--

Description

Supply different renderers for each orthogonal data request. Pass an object with keys `display`, `sort`, `filter`, `type` (all optional). Each value must be a JS function.

Usage

```
dt2_cols_render_orthogonal(
  options = list(),
  col_specs,
  display = NULL,
  sort = NULL,
  filter = NULL,
  type = NULL
)
```

Arguments

options	Options list to modify.
col_specs	Column names or indices.
display	Optional JS renderer for UI display.
sort	Optional JS renderer used for ordering.
filter	Optional JS renderer used for searching.
type	Optional JS renderer used for type detection.

Value

Modified options.

Examples

```
opts <- list(columns = names(iris))
opts <- dt2_cols_render_orthogonal(
  opts, "Sepal.Length",
  display = htmlwidgets::JS("function(d,t,row,meta){ return d + ' cm'; }"),
  sort    = htmlwidgets::JS("function(d,t,row,meta){ return parseFloat(d); }")
)
```

dt2_cols_width	<i>Column widths (CSS)</i>
----------------	----------------------------

Description

Column widths (CSS)

Usage

```
dt2_cols_width(options = list(), map_named)
```

Arguments

options	Options list.
map_named	Named character vector: c(Col="120px", ...).

Value

Updated options.

dt2_col_button	Action button per row
----------------	-----------------------

Description

Action button per row

Usage

```
dt2_col_button(  
  options = list(),  
  col,  
  label = "Action",  
  input_id_prefix = "row_btn_"  
)
```

Arguments

- options Options list.
- col Target column (name or 1-based index).
- label Button label.
- input_id_prefix Prefix for element ids (e.g., "row_btn_").

Value

Updated options.

dt2_col_checkbox	Checkbox input per row
------------------	------------------------

Description

Checkbox input per row

Usage

```
dt2_col_checkbox(  
  options = list(),  
  col,  
  input_id_prefix = "row_chk_",  
  value_col = NULL  
)
```

Arguments

options	Options list.
col	Target column (name or 1-based index).
input_id_prefix	Prefix for element ids (e.g., "row_chk_").
value_col	Optional boolean column to define initial state.

Value

Updated options.

dt2_col_template	<i>Simple HTML template per column (replace {{VAL}})</i>
------------------	--

Description

Simple HTML template per column (replace {{VAL}})

Usage

```
dt2_col_template(options = list(), col, template)
```

Arguments

options	Options list.
col	Name or index of target column.
template	HTML string with {{VAL}} placeholder.

Value

Updated options.

dt2_draw	<i>Redraw the table (proxy)</i>
----------	---------------------------------

Description

Redraw the table (proxy)

Usage

```
dt2_draw(proxy)
```

Arguments

proxy [dt2_proxy\(\)](#).

Value

The proxy object, returned invisibly.

dt2_extensions	<i>List available DataTables extensions</i>
----------------	---

Description

List available DataTables extensions

Usage

```
dt2_extensions()
```

Value

A data.frame with columns name, version, dir.

Examples

```
dt2_extensions()
```

dt2_format_datetime	<i>Format date/time columns (DataTables renderer: datetime)</i>
---------------------	---

Description

Use DataTables' built-in `DataTable.render.datetime` to transform date/time strings for display (and preserve sortability).

Usage

```
dt2_format_datetime(  
  options = list(),  
  col_specs,  
  from = NULL,  
  to = "DD/MM/YYYY",  
  locale = NULL,  
  def = NULL  
)
```

Arguments

options	List of options (returned with columnDefs updated).
col_specs	Column names or 1-based indices.
from	Input format (e.g., "YYYY-MM-DD" or ISO8601 by default).
to	Output format (e.g., "DD/MM/YYYY").
locale	Optional locale (e.g., "pt-BR").
def	Optional default output if input is invalid.

Value

Modified options.

Examples

```
opts <- list(columns = c("when", "val"))
opts <- dt2_format_datetime(opts, "when", from = "YYYY-MM-DD",
                           to = "DD/MM/YYYY", locale = "pt-BR")
```

dt2_format_number	<i>Format numeric columns (DataTables renderer: number)</i>
-------------------	---

Description

Add a number renderer to one or more columns using DataTables' built-in `DataTable.render.number`.

Usage

```
dt2_format_number(
  options = list(),
  col_specs,
  thousands = NULL,
  decimal = NULL,
  digits = 0,
  prefix = "",
  prefix_right = ""
)
```

Arguments

options	List of options (returned, with columnDefs updated).
col_specs	Column names or 1-based indices to format.
thousands	Thousands separator (character or NULL for auto).
decimal	Decimal separator (character or NULL for auto).
digits	Number of decimal places.
prefix, prefix_right	String to prepend/append (e.g., currency symbol).

Value

Modified options.

Examples

```
opts <- list(columns = names(iris))
opts <- dt2_format_number(opts, "Sepal.Length", thousands = ".", decimal = ",",
  digits = 2, prefix = "", prefix_right = "")
```

dt2_format_number_abbrev

Abbreviate large numbers with fixed decimals (k / M / B)

Description

Adds a `columns.render` function that displays numbers as 1.2k, 3.4M, etc. This renderer **lets you control** the number of decimal places via `digits`. Use this when you want a fixed, compact style independent of locale rules.

Usage

```
dt2_format_number_abbrev(
  options = list(),
  col_specs,
  digits = 1,
  locale = NULL
)
```

Arguments

<code>options</code>	A DataTables options list to be modified.
<code>col_specs</code>	Column names or 1-based indices to format.
<code>digits</code>	Integer, decimal places for the abbreviated display (default 1).
<code>locale</code>	Optional BCP-47 locale string (e.g. "pt-BR"). If provided, the non-abbreviated part uses <code>toLocaleString(locale)</code> for grouping.

Value

The modified options list.

Examples

```
opts <- list(columns = names(mtcars))
opts <- dt2_format_number_abbrev(opts, c("hp", "qsec"), digits = 1, locale = "pt-BR")
```

`dt2_format_time_format`*Format a date/time using DataTables' datetime renderer, with locale*

Description

Format a date/time using DataTables' datetime renderer, with locale

Usage

```
dt2_format_time_format(  
  options = list(),  
  col_specs,  
  from = NULL,  
  to = "L",  
  locale = "pt-br"  
)
```

Arguments

<code>options</code>	Options list (returned modified).
<code>col_specs</code>	Column names or indices to format.
<code>from</code>	Input format (e.g. 'YYYY-MM-DDTHH:mm:ssZ' or NULL for ISO).
<code>to</code>	Output format (e.g. 'L LTS'). See moment.js docs.
<code>locale</code>	Locale string, e.g. 'pt-br'.

Value

Modified options.

`dt2_format_time_relative`*Relative time using moment.fromNow(), with locale*

Description

Relative time using moment.fromNow(), with locale

Usage

```
dt2_format_time_relative(options = list(), col_specs, locale = "pt-br")
```

Arguments

options	list of options (returned updated)
col_specs	names or indices to format
locale	e.g. "pt-br" (requires moment-with-locales)

Value

The modified options list with an updated columnDefs entry.

dt2_language	<i>Language helper (either list or JSON url)</i>
--------------	--

Description

Language helper (either list or JSON url)

Usage

```
dt2_language(options = list(), lang_list = NULL, lang_url = NULL)
```

Arguments

options	Options list.
lang_list	Named list of language keys.
lang_url	URL to a JSON translation file.

Value

Updated options.

Examples

```
# Inline translation
opts <- dt2_language(list(), lang_list = list(search = "Buscar:"))
dt2(iris, options = opts)

# Or load a ready-made translation file from the DataTables CDN
opts <- dt2_language(list(),
  lang_url = "https://cdn.datatables.net/plug-ins/2.3.3/i18n/pt-BR.json")
```

dt2_length_menu	<i>Length menu helper</i>
-----------------	---------------------------

Description

Configures the entries-per-page dropdown.

Usage

```
dt2_length_menu(options = list(), values = c(10, 25, 50, -1), labels = NULL)
```

Arguments

options	Options list.
values	Numeric vector of page lengths (e.g., c(10, 25, 50, -1)). Use -1 for "show all".
labels	Optional character vector of labels. If NULL, numeric values are used as-is and -1 becomes "All" automatically via language.lengthLabels.

Value

Updated options.

Examples

```
opts <- dt2_length_menu(values = c(5, 10, 25, -1))
dt2(iris, options = opts)

opts <- dt2_length_menu(values = c(10, 50, 100), labels = c("10", "50", "100"))
dt2(iris, options = opts)
```

dt2_order	<i>Define initial ordering (option order)</i>
-----------	---

Description

Define initial ordering (option order)

Usage

```
dt2_order(options = list(), ...)
```

Arguments

options	Options list.
...	Vectors like c(col, "asc"/"desc"). col may be name or 1-based index.

Value

Updated options.

Examples

```
opts <- list(columns = names(mtcars))
opts <- dt2_order(opts, c("mpg", "desc"))
dt2(mtcars, options = opts)
```

dt2_output	<i>Shiny output for DT2</i>
------------	-----------------------------

Description

Place a DT2 table in a Shiny UI.

Usage

```
dt2_output(outputId, width = "100%", height = "auto")
```

Arguments

- outputId Output ID (must match the render_dt2() call in server).
- width, height CSS dimensions.

Value

An htmlwidgets Shiny output (HTML container) suitable for inclusion in a Shiny UI definition.

dt2_proxy	<i>Create a proxy for a DT2 table</i>
-----------	---------------------------------------

Description

Create a proxy for a DT2 table

Usage

```
dt2_proxy(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

- id Widget id used in [dt2_output\(\)](#).
- session Shiny session.

Value

A "DT2Proxy" object.

dt2_proxy_order	<i>Order the table (proxy)</i>
-----------------	--------------------------------

Description

Order the table (proxy)

Usage

```
dt2_proxy_order(proxy, ..., columns = NULL)
```

Arguments

proxy	dt2_proxy() .
...	Vectors c(col, "asc"/"desc"). If you pass columns, names will be resolved to indices.
columns	Optional character vector of column names to resolve names to indices.

Value

The proxy object, returned invisibly.

dt2_proxy_page	<i>Page navigation (proxy)</i>
----------------	--------------------------------

Description

Page navigation (proxy)

Usage

```
dt2_proxy_page(
  proxy,
  page = c("first", "previous", "next", "last", "number"),
  number = NULL
)
```

Arguments

proxy	A dt2_proxy() object.
page	Navigation action: "first", "previous", "next", "last", or "number" (go to a specific page).
number	Page number (1-based). Only used when page = "number".

Value

The proxy, invisibly.

dt2_proxy_search	<i>Global search (proxy)</i>
------------------	------------------------------

Description

Global search (proxy)

Usage

```
dt2_proxy_search(  
  proxy,  
  value,  
  regex = FALSE,  
  smart = TRUE,  
  caseInsensitive = TRUE  
)
```

Arguments

- proxy A [dt2_proxy\(\)](#) object.
- value Search string.
- regex Logical; treat value as a regular expression? Default: FALSE.
- smart Logical; use DataTables smart search? Default: TRUE.
- caseInsensitive Logical; case-insensitive search? Default: TRUE.

Value

The proxy, invisibly.

dt2_register_renderer	<i>Register a named JS renderer</i>
-----------------------	-------------------------------------

Description

Register a named JS renderer

Usage

```
dt2_register_renderer(name, js)
```

Arguments

- name Unique name (character scalar).
- js A `htmlwidgets::JS()` function or a JSON helper expression.

Value

Invisibly, the name.

dt2_replace_data	<i>Replace all data in the table (proxy)</i>
------------------	--

Description

Replace all data in the table (proxy)

Usage

```
dt2_replace_data(proxy, data)
```

Arguments

proxy	dt2_proxy() .
data	New data.frame (will be serialized).

Value

The proxy object, returned invisibly.

dt2_search_global	<i>Set global search (option search)</i>
-------------------	--

Description

Set global search (option search)

Usage

```
dt2_search_global(  
  options = list(),  
  value,  
  regex = FALSE,  
  smart = TRUE,  
  caseInsensitive = TRUE  
)
```

Arguments

options	Options list.
value	Text.
regex, smart, caseInsensitive	Search flags.

Value

Updated options.

Examples

```
opts <- dt2_search_global(list(), value = "Toyota")
dt2(mtcars, options = opts)
```

dt2_select_rows	Select rows (proxy; Select extension)
-----------------	---------------------------------------

Description

Select rows (proxy; Select extension)

Usage

```
dt2_select_rows(proxy, indexes, reset = TRUE)
```

Arguments

- proxy [dt2_proxy\(\)](#).
- indexes 1-based row indices.
- reset If TRUE, clear selection before selecting.

Value

The proxy object, returned invisibly.

dt2_ssp_handler	Default server-side handler (filter/order/page)
-----------------	---

Description

Default server-side handler (filter/order/page)

Usage

```
dt2_ssp_handler(names, rows_all = TRUE)
```

Arguments

names	character() column names in display order.
rows_all	Logical. If TRUE (default), the response also carries the 1-based indices (into the source data) of all rows that survive the current filter (dt2_rows_all) and of the rows on the current page (dt2_rows_current). dt2.js forwards them to input\$<id>_rows_all and input\$<id>_rows_current, mirroring the client-side behaviour. Set to FALSE for very large tables, where shipping the full index vector on every draw is too costly; the two inputs are then NULL.

Value

function(data, req) -> list(draw, recordsTotal, recordsFiltered, data, and optionally dt2_rows_all, dt2_rows_current)

dt2_state

Access the current state snapshot of a DT2 table

Description

Returns a list with reason, order, search, page, selected, rows_all, rows_current, rows_selected and state reflecting the current client-side table state.

Usage

```
dt2_state(input, id)
```

Arguments

input	Shiny input object.
id	Widget ID.

Details

The widget pushes input\$<id>_state on every init, draw, order, search, page, select and deselect event. Besides that snapshot, it also sets three standalone inputs with the same names as the DT package, all **1-based** row indices into the original data:

- input\$<id>_rows_all – rows that survive the current filters (global search and per-column searches, e.g. from the ColumnControl extension), in display order.
- input\$<id>_rows_current – the rows shown on the current page.
- input\$<id>_rows_selected – the selected rows (Select extension).

The same vectors are available inside the snapshot as rows_all, rows_current and rows_selected. The legacy selected element is kept unchanged (0-based, as DataTables reports it).

With server-side processing the client only holds the current page, so rows_all / rows_current are taken from the server response: the default `dt2_ssp_handler()` ships them unless rows_all = FALSE (see `dt2_bind_server()`); a custom handler must return dt2_rows_all / dt2_rows_current

itself, otherwise those inputs are NULL. In server mode rows_selected is mapped through rows_current (so it is also NULL without it), and only rows of the current page can be selected, because DataTables discards server-side selections on every redraw. The order, search and page snapshots fire before the server replies, so they carry NULL row vectors; the draw snapshot that follows has the fresh values (the standalone input<id>_rows_* inputs only change on that draw).

Value

A list with the current table state.

dt2_theme	Create a reusable DT2 theme
-----------	-----------------------------

Description

Creates a theme object that can be passed to dt2() via the theme parameter. Useful when you want the same look across many tables.

For quick one-off styling, you can also pass arguments directly to dt2() (e.g., dt2(iris, striped = FALSE)).

Usage

```
dt2_theme(  
  preset = "default",  
  striped = NULL,  
  hover = NULL,  
  compact = NULL,  
  font_scale = NULL,  
  style = NULL,  
  button_class = NULL  
)
```

Arguments

preset	A named preset to start from: "default", "clean", "minimal", or "compact". Remaining arguments override the preset.
striped	Logical; alternate row colours.
hover	Logical; highlight rows on hover.
compact	Logical; reduce cell padding.
font_scale	Numeric; font-size multiplier (e.g., 0.85 = 85%).
style	Styling framework: "bootstrap5" (default) or "core" (plain DataTables).
button_class	CSS class string for Buttons extension buttons. Default: "btn btn-sm btn-outline-secondary". See Bootstrap 5 button classes for options (e.g., "btn btn-sm btn-primary").

Value

A dt2_theme object (a named list).

Examples

```
# Create and reuse
my_theme <- dt2_theme("clean", compact = TRUE)

dt2(iris, theme = my_theme)
dt2(mtcars, theme = my_theme)

# Custom button style
dt2_theme("default", button_class = "btn btn-sm btn-primary")

# Presets
dt2_theme("minimal")
dt2_theme("compact")
```

dt2_update_libs

Update DataTables JS/CSS libraries (developer tool)

Description

Checks for updates (respecting version constraints), patches version numbers in the source files, and optionally runs tools/get-dt2-libs.sh to download the new files.

Usage

```
dt2_update_libs(pkg_dir = ".", download = TRUE, dry_run = FALSE)
```

Arguments

pkg_dir	Path to the DT2 source root (the directory containing DESCRIPTION). Defaults to the current working directory.
download	Logical. If TRUE, runs the shell script after patching version numbers. Default TRUE.
dry_run	Logical. If TRUE, shows what would change without modifying any files. Default FALSE.

Details

This function only works from the **package source tree** (i.e. during development). It will refuse to run from an installed package.

The workflow is:

1. Query npm for the latest compatible version of every library.

2. Patch tools/get-dt2-libs.sh (version variables).
3. Patch R/dt2_extensions.R (extension registry).
4. Patch R/dt2_check_updates.R (core lib versions).
5. Patch R/dt2_deps.R (DataTables core version).
6. Run bash tools/get-dt2-libs.sh to download the files.

Version constraints prevent incompatible upgrades:

- jQuery is pinned to 3.x (DataTables 2 requires jQuery 3).
- pdfmake is pinned to 0.2.x (0.3.x has breaking changes and is not available on cdnjs).
- Bootstrap is pinned to 5.x.

Libraries marked as "PINNED" are skipped. Only "UPDATE" items are applied.

Value

Invisibly, a data.frame with the update results.

Examples

```
## Not run:
# Developer-only tool: requires the DT2 package source tree
# (DESCRIPTION, tools/get-dt2-libs.sh, R/dt2_extensions.R, ...).
# It cannot run from an installed package, so it is not executable
# in CRAN check or from a regular user session.

# from the DT2 source root:
dt2_update_libs()

# preview changes without modifying anything:
dt2_update_libs(dry_run = TRUE)

## End(Not run)
```

dt2_use_buttons

Enable Buttons (extension) and define buttons

Description

Uses the modern DataTables 2.x layout API (not the deprecated dom).

Usage

```
dt2_use_buttons(
  options = list(),
  buttons = c("copy", "csv", "excel", "print"),
  position = "topEnd",
  button_class = NULL
)
```

Arguments

options	Options list.
buttons	Vector of button ids (e.g., c("copy","csv","excel","print","colvis")).
position	Where to place buttons in the layout. One of "topEnd" (default), "topStart", "bottomEnd", "bottomStart".
button_class	CSS class for buttons (e.g., "btn btn-sm btn-primary"). If NULL, uses the theme default ("btn btn-sm btn-outline-secondary"). Applied per-button via className.

Value

Updated options.

See Also

[dt2_buttons\(\)](#) for a lower-level variant that takes full button objects and can relocate the buttons container to a custom CSS selector.

Examples

```
opts <- dt2_use_buttons(list(), buttons = c("copy", "csv", "excel"))
dt2(mtcars, options = opts)
```

dt2_use_renderer	<i>Use a named JS renderer on columns</i>
------------------	---

Description

Use a named JS renderer on columns

Usage

```
dt2_use_renderer(options = list(), col_specs, name)
```

Arguments

options	Options list (returned modified).
col_specs	Column names or indices.
name	Name used in dt2_register_renderer().

Value

Modified options.

observe_dt2_events	<i>Observe DataTables events published by dt2.js</i>
--------------------	--

Description

Listen for table events (init, draw, order, search, page, select, deselect).

Usage

```
observe_dt2_events(input, id, handler)
```

Arguments

input	Shiny input object.
id	Widget ID.
handler	Function with signature (event, type, indexes, rowData).

Value

No return value, called for side effects. Sets up a Shiny observer that calls handler whenever the table emits an event.

render_dt2	<i>Shiny render function for DT2</i>
------------	--------------------------------------

Description

Render a DT2 table in a Shiny server function.

Usage

```
render_dt2(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	Expression returning a <code>dt2()</code> widget.
env, quoted	Standard shinyRenderWidget arguments.

Value

A Shiny render function (closure produced by `htmlwidgets::shinyRenderWidget()`) that emits a DT2 widget.

Index

dt2, [3](#)
dt2(), [26](#), [30](#)
dt2_bind_server, [5](#)
dt2_bind_server(), [25](#)
dt2_buttons, [6](#)
dt2_buttons(), [29](#)
dt2_check_updates, [7](#)
dt2_col_button, [12](#)
dt2_col_checkbox, [12](#)
dt2_col_template, [13](#)
dt2_cols_align, [8](#)
dt2_cols_escape, [8](#)
dt2_cols_hide, [9](#)
dt2_cols_html, [9](#)
dt2_cols_render_js, [10](#)
dt2_cols_render_orthogonal, [10](#)
dt2_cols_width, [11](#)
dt2_draw, [13](#)
dt2_extensions, [14](#)
dt2_format_datetime, [14](#)
dt2_format_number, [15](#)
dt2_format_number_abbrev, [16](#)
dt2_format_time_format, [17](#)
dt2_format_time_relative, [17](#)
dt2_language, [18](#)
dt2_length_menu, [19](#)
dt2_order, [19](#)
dt2_output, [20](#)
dt2_output(), [20](#)
dt2_proxy, [20](#)
dt2_proxy(), [14](#), [21–24](#)
dt2_proxy_order, [21](#)
dt2_proxy_page, [21](#)
dt2_proxy_search, [22](#)
dt2_register_renderer, [22](#)
dt2_replace_data, [23](#)
dt2_search_global, [23](#)
dt2_select_rows, [24](#)
dt2_ssp_handler, [24](#)
dt2_ssp_handler(), [5](#), [25](#)
dt2_state, [25](#)
dt2_theme, [26](#)
dt2_theme(), [3](#)
dt2_update_libs, [27](#)
dt2_use_buttons, [28](#)
dt2_use_buttons(), [6](#)
dt2_use_renderer, [29](#)

htmlwidgets::JS, [9](#)
htmlwidgets::shinyRenderWidget(), [30](#)

observe_dt2_events, [30](#)

render_dt2, [30](#)