

Package: privatusanalytics (via r-universe)

October 25, 2026

Title Add the 'Privatus Analytics' Tracking Script to 'shiny' Apps and Documents

Version 1.0.0

Description Adds the cookieless 'Privatus Analytics' [<https://privatusanalytics.com>](https://privatusanalytics.com) tracking script to 'shiny' apps, 'R Markdown' and 'Quarto' documents, built with 'htmltools'. Also adds the HTML data attributes the script reads for custom events, click maps and page properties. The package only prints HTML. It never sends web requests or events from R and sets no cookies.

License MIT + file LICENSE

URL <https://privatusanalytics.com>,
<https://github.com/Privatus-Analytics/r>

BugReports <https://github.com/Privatus-Analytics/r/issues>

Depends R (>= 4.1)

Imports htmltools (>= 0.5.2), utils

Suggests shiny, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository <https://privatus-analytics.r-universe.dev>

Date/Publication 2026-10-25 10:35:47 UTC

RemoteUrl <https://github.com/Privatus-Analytics/r>

RemoteRef v1.0.0

RemoteSha 6d8c007355089f12c126ceee4fc85d48d5224b23

Contents

privatus_analytics	2
privatus_click	4
privatus_enabled	5
privatus_event	5
privatus_shiny	6
use_privatus_header	8

Index	9
--------------	----------

privatus_analytics	<i>The Privatus Analytics script tag</i>
--------------------	--

Description

Returns the tracking script as an `htmltools::tagList()`. Put it in the `<head>` of your page: `tags$head(privatus_analytics())` in a 'shiny' app, or see `use_privatus_header()` for R Markdown and 'Quarto'.

Usage

```
privatus_analytics(
  site_id = getOption("privatus.site_id"),
  props = NULL,
  request = NULL,
  nonce = getOption("privatus.nonce"),
  modules = getOption("privatus.modules"),
  excluded_paths = getOption("privatus.excluded_paths"),
  domains = getOption("privatus.domains"),
  spa = getOption("privatus.spa"),
  manual = getOption("privatus.manual", FALSE),
  enabled = getOption("privatus.enabled"),
  extra_attributes = getOption("privatus.extra_attributes"),
  queue_stub = getOption("privatus.queue_stub", FALSE),
  script_host = getOption("privatus.script_host")
)
```

Arguments

site_id	Your site id. It starts with pa_. Falls back to the PRIVATUS_SITE_ID environment variable. Nothing prints without it.
props	Page props for this page, as a named list. They are sent with every pageview and event on the page. They are merged over the <code>privatus.props</code> option, and a NULL or NA value removes one. The option can be a named list, or a function that gets request and returns one. Never put names, emails, user ids or anything that points to one person in props.

request	Passed to a <code>privatus.props</code> function. In a 'shiny' app whose <code>ui</code> is a function, pass its <code>req</code> .
nonce	A Content Security Policy nonce for the script tags, if your site uses one.
modules	Tracker modules to load, for example <code>c("engage", "auto")</code> . <code>NULL</code> keeps the tracker default. <code>character(0)</code> loads none.
excluded_paths	Path patterns the tracker never counts, for example <code>c("/admin/*", "/preview/*")</code> .
domains	Hostnames to count. <code>NULL</code> counts every host.
spa	Single-page app mode: <code>"auto"</code> , <code>"history"</code> , <code>"hash"</code> or <code>"off"</code> . Only a value other than <code>"auto"</code> is printed.
manual	<code>TRUE</code> turns off the automatic pageview.
enabled	<code>TRUE</code> or <code>FALSE</code> forces the tag on or off. <code>NULL</code> uses <code>privatus_enabled()</code> .
extra_attributes	A named list of any other data- attributes, for example <code>list("data-debug" = "true")</code> .
queue_stub	<code>TRUE</code> prints a small script first, so <code>privatus.track()</code> calls made before the tracker loads are kept and sent later. It prints even when the tag is off, so those calls never fail.
script_host	Where <code>/js/pa.js</code> is served. Falls back to the <code>PRIVATUS_SCRIPT_HOST</code> environment variable, then <code>https://privatusstats.com</code> . Change it only if you host your own copy.

Details

When there is no site id, or when the tag is off (see `privatus_enabled()`), it returns only the queue stub when `queue_stub` is `TRUE`, and otherwise an empty tag list. Every argument defaults to an R option of the same name with a `privatus.` prefix, so you can set them once with `options()` and still change any of them for one page.

When there are page props, a small script is printed before the tracker. It sets `window.privatusPageProps`, and the tracker tag gets `data-props="privatusPageProps"`. With `queue_stub = TRUE` the official queue stub is printed first.

Value

An `htmltools::tagList()`.

Examples

```
privatus_analytics(site_id = "pa_YOURSITEID", enabled = TRUE)

privatus_analytics(
  site_id = "pa_YOURSITEID", enabled = TRUE,
  modules = c("engage", "auto"), props = list(plan = "pro", role = "admin")
)

# In a Shiny app:
# ui <- fluidPage(tags$head(privatus_analytics()), ...)
```

privatus_click	<i>Click map names</i>
----------------	------------------------

Description

A click map shows where people click on a page. Give the important parts a name so they stay easy to find, even when the page changes.

Usage

```
privatus_click(tag, name)
```

```
privatus_click_data(name)
```

```
privatus_click_attr(name)
```

Arguments

tag	An 'htmltools' tag, for example <code>htmltools::tags\$a("See plans")</code> .
name	The click map name, for example <code>"pricing.cta"</code> .

Details

A name may use letters, digits and `_ . : -`, up to 100 characters. Anything else gives nothing. Outside production (see [privatus_enabled\(\)](#)) you also get a warning, so you can fix the name.

- `privatus_click()` adds `data-privatus-click` to an 'htmltools' tag.
- `privatus_click_data()` returns it as a named list.
- `privatus_click_attr()` returns it as an HTML string.

Value

`privatus_click()` returns the tag, with the attribute added when the name is valid. `privatus_click_data()` returns a named list (empty for an invalid name). `privatus_click_attr()` returns an `htmltools::HTML()` string (empty for an invalid name).

Examples

```
library(htmltools)
privatus_click(tags$a(href = "/pricing", "See plans"), "pricing.cta")
privatus_click_attr("pricing.cta")
```

privatus_enabled	<i>Whether the tag is on</i>
------------------	------------------------------

Description

The tag is on when enabled is TRUE and off when it is FALSE. When enabled is NULL (the default), the tag is on only when the PRIVATUS_ENV or R_CONFIG_ACTIVE environment variable is "production". So your own clicking around while you build is never counted.

Usage

```
privatus_enabled(enabled = getOption("privatus.enabled"))
```

Arguments

enabled	TRUE, FALSE or NULL. Defaults to the <code>privatus.enabled</code> option.
---------	--

Details

R has no single setting for "this is the live site". If you deploy to shinyapps.io or 'Posit Connect', set PRIVATUS_ENV=production in the app's environment variables (or R_CONFIG_ACTIVE=production if you use the 'config' package), or set `options(privatus.enabled = TRUE)`.

Value

TRUE or FALSE.

Examples

```
privatus_enabled()
privatus_enabled(TRUE)
```

privatus_event	<i>Event attributes for buttons, links and forms</i>
----------------	--

Description

The tracker reads these data- attributes in the browser and sends a custom event when the element is clicked. On a `<form>` the event is sent when the form is submitted. Your R code sends nothing.

Usage

```
privatus_event(tag, name, ..., .props = list())
privatus_event_data(name, ..., .props = list())
privatus_event_attrs(name, ..., .props = list())
```

Arguments

tag	An 'htmltools' tag, for example <code>htmltools::tags\$button("Export")</code> .
name	The event name. Use a short "Noun verb" name like "Report exported".
...	Event props as key = value pairs, for example <code>format = "csv"</code> .
.props	A named list of more props. Props in ... win when a key is in both.

Details

- `privatus_event()` adds the attributes to an 'htmltools' tag, such as `tags$button()` or `shiny::ActionButton()`.
- `privatus_event_data()` returns them as a named list, so you can splice them into a tag with `!!!` or `do.call()`.
- `privatus_event_attrs()` returns them as an HTML string to put inside a tag you write by hand, for example in an R Markdown document.

The name is trimmed and cut to 120 characters. An empty name gives nothing. Prop keys are made lowercase, and anything that is not a letter, a digit or `_` becomes `_`, so "Report Type" becomes `report_type`. Keys are cut to 64 characters. At most 30 props are kept. Values become text: TRUE is "true", numbers are written out, and vectors are joined with commas. NULL and NA values are left out. Values are cut to 500 characters. Everything is escaped for HTML.

The attributes print even when the tracker tag is off. They do nothing without the tracker, so your pages look the same in development and production.

Never put names, emails, user ids or anything that points to one person in event props.

Value

`privatus_event()` returns the tag with the attributes added. `privatus_event_data()` returns a named list (empty for an empty name). `privatus_event_attrs()` returns an `htmltools::HTML()` string (empty for an empty name).

Examples

```
library(htmltools)
privatus_event(tags$button("Export"), "Report exported", format = "csv")
tags$button("Export", !!!privatus_event_data("Report exported", format = "csv"))
privatus_event_attrs("Report exported", format = "csv", source = "toolbar")
```

privatus_shiny

Track events from a Shiny server function

Description

`privatus_shiny()` adds a tiny script to your 'shiny' app. It listens for messages from your server function and passes them to the tracker in the browser with `privatus.track()`. `privatus_track()` sends such a message.

Usage

```
privatus_shiny()

privatus_track(name, ..., .props = list(), session = NULL)
```

Arguments

name	The event name. Use a short "Noun verb" name like "Report exported".
...	Event props as key = value pairs, for example format = "csv".
.props	A named list of more props.
session	The 'shiny' session. Defaults to the current one.

Details

Your R server never talks to Privatus. The message goes to the visitor's own browser over the 'shiny' connection, and the browser's tracker sends the event, just like a click would. When the tracker is not on the page (for example in development) nothing happens.

Props follow the page props rules: keys are cleaned, numbers and TRUE/FALSE keep their type, and at most 30 props are kept. Never put names, emails, user ids or anything that points to one person in props.

Value

`privatus_shiny()` returns an `htmltools::tagList()` holding the script as an `htmltools::htmlDependency()`.
`privatus_track()` returns TRUE invisibly when a message was sent, and FALSE for an empty name.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  library(shiny)
  ui <- fluidPage(
    tags$head(privatus_analytics(), privatus_shiny()),
    downloadButton("export", "Export")
  )
  server <- function(input, output, session) {
    output$export <- downloadHandler("data.csv", function(file) {
      privatus_track("Report exported", format = "csv")
      utils::write.csv(mtcars, file)
    })
  }
  shinyApp(ui, server)
}
```

use_privatus_header	<i>Write the tag to a file for R Markdown and Quarto</i>
---------------------	--

Description

R Markdown and 'Quarto' can add an HTML file to the <head> of every page they build. This writes the tag (from `privatus_analytics()`) to such a file. Point your document at it:

Usage

```
use_privatus_header(path = "_privatus.html", ..., quiet = FALSE)
```

Arguments

path	Where to write the file.
...	Passed to <code>privatus_analytics()</code> , for example <code>site_id</code> or <code>props</code> .
quiet	TRUE hides the message that says what was written.

Details

R Markdown (html_document and most HTML formats):

```
output:
  html_document:
    includes:
      in_header: _privatus.html
```

'Quarto' (_quarto.yml or the document header):

```
format:
  html:
    include-in-header: _privatus.html
```

Write the file before you build, for example in a build script or as a 'Quarto' pre-render script, so it follows your settings each time. When the tag is off (see `privatus_enabled()`) or there is no site id, the file is written empty (or with only the queue stub when `queue_stub = TRUE`), so the build still works.

Value

The path, invisibly.

Examples

```
path <- tempfile(fileext = ".html")
use_privatus_header(path, site_id = "pa_YOURSITEID", enabled = TRUE)
readLines(path)
```

Index

`do.call()`, 6

`htmltools::HTML()`, 4, 6

`htmltools::htmlDependency()`, 7

`htmltools::tagList()`, 2, 3, 7

`options()`, 3

`privatus_analytics`, 2

`privatus_analytics()`, 8

`privatus_click`, 4

`privatus_click_attr(privatus_click)`, 4

`privatus_click_data(privatus_click)`, 4

`privatus_enabled`, 5

`privatus_enabled()`, 3, 4, 8

`privatus_event`, 5

`privatus_event_attrs(privatus_event)`, 5

`privatus_event_data(privatus_event)`, 5

`privatus_shiny`, 6

`privatus_track(privatus_shiny)`, 6

`use_privatus_header`, 8

`use_privatus_header()`, 2