

# Package: shinyStep (via r-universe)

May 30, 2026

**Type** Package

**Title** User-Editable R Functions in 'Shiny' Apps with a Step Debugger

**Version** 0.5.1

**Description** A pair of 'Shiny' modules that let end users of a 'Shiny' application author their own R functions directly in the browser. Host apps can expose these modules as extension points where user-supplied code augments or replaces built-in logic, without requiring users to modify the app's source. Each module embeds an 'Ace' editor with a structured argument table, an in-frame R console rooted in the paused function's local environment, and a step debugger that handles for, while, repeat, and if/else blocks at any nesting depth. Two module flavours are provided: solo editors for testing a function in isolation with literal argument values, and embedded editors for pausing a function mid-execution inside a larger host program.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** false

**URL** <https://github.com/zhangh12/shinyStep>

**BugReports** <https://github.com/zhangh12/shinyStep/issues>

**Imports** shiny (>= 1.7.0), shinyAce (>= 0.4.0)

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**RoxygenNote** 7.3.3

**Config/pak/sysreqs** cmake make libuv1-dev zlib1g-dev

**Repository** <https://zhangh12.r-universe.dev>

**Date/Publication** 2026-05-23 04:32:57 UTC

**RemoteUrl** <https://github.com/zhangh12/shinystep>

**RemoteRef** HEAD

**RemoteSha** 1acfe8afc5e5088ad0654e0f820472866341bfcf

## Contents

continue_to_next_pause . . . . .	2
embeddedStepServer . . . . .	3
embeddedStepUI . . . . .	4
make_runner . . . . .	5
run_demo . . . . .	5
run_program . . . . .	6
soloStepServer . . . . .	6
soloStepUI . . . . .	8
step_fn . . . . .	9
step_out_frame . . . . .	9
stop_runner . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

continue\_to\_next\_pause

*Continue execution to the next registered pause point (Continue button)*

---

### Description

Drains all remaining expressions in the currently paused function, then resumes the main program until the next registered function is entered or the program finishes.

### Usage

```
continue_to_next_pause(runner, run_log)
```

### Arguments

runner	Runner from make_runner().
run_log	A reactiveVal for log output.

### Value

No return value. Called for its side effect of resuming execution and updating run\_log.

---

embeddedStepServer      *Server for an embedded-mode step debugger*

---

## Description

Registers the function with the shared runner and manages the editor, argument table, and Debug checkbox. The host app calls `run_program()` with its own `main_code`; this module contributes a pause point automatically whenever Debug is ticked.

## Usage

```
embeddedStepServer(
  id,
  runner,
  run_log,
  initial_fn_name = NULL,
  initial_body = NULL,
  initial_args = NULL,
  reserved_args = NULL
)
```

## Arguments

<code>id</code>	Module namespace ID. Must match <code>embeddedStepUI()</code> .
<code>runner</code>	Runner object from <code>make_runner()</code> .
<code>run_log</code>	A <code>reactiveVal(character(1))</code> shared across all modules and the host app.
<code>initial_fn_name</code>	Initial function name — static string or reactive.
<code>initial_body</code>	Initial function body — static string or reactive.
<code>initial_args</code>	Initial argument specs — a list of <code>list(name, default)</code> entries, or a reactive returning such a list. (No <code>test_value</code> column — embedded functions are called from <code>main_code</code> , not via a Test button.)
<code>reserved_args</code>	Optional list of reserved-argument specs. Each entry pins one row at the top of the argument table: the name field is readonly, the delete button is hidden, and the user cannot rename, remove, or reorder the row. Entries may be a bare character name ("trial") or a list with fields: <ul style="list-style-type: none"> <li><code>name</code> The reserved argument name (required).</li> <li><code>allow_default</code> If FALSE, the default-value field is hidden for this row. Defaults to TRUE.</li> <li><code>allow_test_value</code> If FALSE, the test-value field is hidden for this row. Defaults to TRUE. (Embedded mode never shows the test-value column, so this flag is usually moot.)</li> </ul> Most callers only need to pin names (e.g. <code>list("n")</code> ); disallowing both value fields is unusual but useful when the host supplies the argument at call time (e.g. <code>list(list(name = "trial", allow_default = FALSE, allow_test_value = FALSE))</code> ).

**Value**

A named list:

save\_clicked, back\_clicked Reactives that fire on Save / Back clicks.

fn\_name Reactive returning the current function name.

enabled Reactive — TRUE when the Debug checkbox is ticked.

get\_fn\_name(), get\_body(), get\_args() Functions returning the current editor state (isolated reads).

---

 embeddedStepUI

*UI for an embedded-mode step debugger*


---

**Description**

Renders the function editor with a **Debug** checkbox. Ticking Debug makes this function a pause point the next time the host app calls `run_program()`. The editor holds the function body only — do not type `fn_name <- function(...){}` wrappers; they are stripped defensively if present. The function name and argument list live in structured inputs above the editor.

**Usage**

```
embeddedStepUI(
  id,
  label = id,
  height = "500px",
  theme = "textmate",
  default_body = "",
  default_fn_name = ""
)
```

**Arguments**

id	Module namespace ID.
label	Toolbar label. Defaults to id.
height	Ace editor height as a CSS string (e.g. "500px").
theme	Ace editor theme name (passed to <b>shinyAce</b> ).
default_body	Initial body text pre-filled in the editor.
default_fn_name	Initial function name pre-filled in the name input.

**Details**

Pair with [embeddedStepServer\(\)](#) using the same id.

**Value**

A tagList suitable for inclusion anywhere in a Shiny UI.

---

make_runner	<i>Create a shared debugger runner</i>
-------------	--

---

**Description**

Call once inside your Shiny server() function. Pass the returned object to every `soloStepServer()` / `embeddedStepServer()` call and to `run_program()`.

**Usage**

```
make_runner()
```

**Value**

A list with three named fields:

**state** A `reactiveValues` holding all execution state (running, paused, step stack, environments, etc.).

**registry** Environment mapping module ids to their accessor functions. Populated automatically by each step-server module.

**pd** Environment of `getParseData()` results keyed by module id. Populated at run time and used for editor line-number tracking.

---

run_demo	<i>Launch the bundled demo Shiny app</i>
----------	--

---

**Description**

Runs the demo application shipped under `inst/test_app/`, which shows both solo and embedded modules with a shared packages prelude and a host Main program panel.

**Usage**

```
run_demo(...)
```

**Arguments**

... Further arguments passed to `runApp`, e.g. `port`, `host`, `launch.browser`.

**Value**

This function is invoked for its side effect and does not return.

**Examples**

```
if (interactive()) {
  shinyStep::run_demo()
}
```

---

run_program	<i>Launch the main program</i>
-------------	--------------------------------

---

### Description

Initialises the runner, installs proxy closures for every registered module whose body is non-blank, and starts executing `main_code`. Execution pauses automatically when a function listed in `debug_targets` is called.

### Usage

```
run_program(runner, main_code, debug_targets = NULL, run_log)
```

### Arguments

runner	Runner from <code>make_runner()</code> .
main_code	Character string – the R program to execute.
debug_targets	Character vector of function names to pause at, or NULL (default) to auto-collect from every embedded module whose Debug checkbox is ticked. Solo modules are never auto-included.
run_log	A <code>reactiveVal</code> used as the shared log.

### Value

No return value. Called for its side effect of installing proxy closures, executing `main_code`, and updating the runner state (which Shiny observers react to).

---

soloStepServer	<i>Server for a solo-mode step debugger</i>
----------------	---

---

### Description

Registers the function with the shared runner, manages the editor and argument table, and wires the **Test** button. Clicking Test assembles `fn_name(arg = test_value, ...)` and calls `run_program()` with `debug_targets = fn_name`, pausing at the first body expression.

### Usage

```
soloStepServer(
  id,
  runner,
  run_log,
  initial_fn_name = NULL,
  initial_body = NULL,
  initial_args = NULL,
```

```

    prelude = NULL,
    reserved_args = NULL
)

```

### Arguments

<code>id</code>	Module namespace ID. Must match <code>soloStepUI()</code> .
<code>runner</code>	Runner object from <code>make_runner()</code> .
<code>run_log</code>	A <code>reactiveVal(character(1))</code> shared across all modules and the host app.
<code>initial_fn_name</code>	Initial function name – static string or reactive. Leave <code>NULL</code> or <code>""</code> to start unnamed.
<code>initial_body</code>	Initial function body – static string or reactive.
<code>initial_args</code>	Initial argument specs – a list of <code>list(name, default, test_value)</code> entries, or a reactive returning such a list. <code>test_value</code> is used by the Test button; <code>default</code> is the function signature default.
<code>prelude</code>	Optional character string or reactive prepended to the generated call on every Test click. Use it to load packages or define helpers the function needs, e.g. <code>"library(dplyr)"</code> .
<code>reserved_args</code>	Optional list of reserved-argument specs. Each entry pins one row at the top of the argument table: the name field is readonly, the delete button is hidden, and the user cannot rename, remove, or reorder the row. Entries may be a bare character name ( <code>"n"</code> ) or a list with fields: <ul style="list-style-type: none"> <li><code>name</code> The reserved argument name (required).</li> <li><code>allow_default</code> If <code>FALSE</code>, the default-value field is hidden for this row. Defaults to <code>TRUE</code>.</li> <li><code>allow_test_value</code> If <code>FALSE</code>, the test-value field is hidden for this row. Defaults to <code>TRUE</code>.</li> </ul> Most callers only need to pin names (e.g. <code>list("n")</code> for a simulator that always passes <code>n</code> ); disallowing both value fields is unusual but useful when the host supplies the argument at call time (e.g. <code>list(list(name = "trial", allow_default = FALSE, allow_test_value = FALSE))</code> ).

### Value

A named list:

`save_clicked`, `back_clicked` Reactives that fire on Save / Back clicks. Wire these in the host app to persist or discard the editor state.

`fn_name` Reactive returning the current function name.

`get_fn_name()`, `get_body()`, `get_args()` Functions returning the current editor state (isolated reads).

---

`soloStepUI`*UI for a solo-mode step debugger*

---

## Description

Renders the function editor with a **Test** button. The editor holds the function body only – do not type `fn_name <- function(...)` wrappers; they are stripped defensively if present. The function name and argument list (including test values) live in structured inputs above the editor.

## Usage

```
soloStepUI(  
  id,  
  label = id,  
  height = "500px",  
  theme = "textmate",  
  default_body = "",  
  default_fn_name = ""  
)
```

## Arguments

<code>id</code>	Module namespace ID.
<code>label</code>	Toolbar label. Defaults to <code>id</code> .
<code>height</code>	Ace editor height as a CSS string (e.g. "500px").
<code>theme</code>	Ace editor theme name (passed to <b>shinyAce</b> ).
<code>default_body</code>	Initial body text pre-filled in the editor.
<code>default_fn_name</code>	Initial function name pre-filled in the name input. Use this when re-mounting the editor for an already-named function so the field is populated on first render (the reactive <code>initial_fn_name</code> alone cannot seed the DOM that is created later).

## Details

Pair with `soloStepServer()` using the same `id`.

## Value

A `tagList` suitable for inclusion anywhere in a Shiny UI.

---

step_fn	<i>Execute the next expression, auto-expanding compound blocks (Next button)</i>
---------	--

---

**Description**

For simple expressions, evaluates atomically. For compound expressions (for/while/repeat/if), pushes their body onto the step stack so sub-expressions can be stepped individually. Matches RStudio debugger behaviour – no separate Step Into needed.

**Usage**

```
step_fn(runner, run_log)
```

**Arguments**

runner	Runner from <code>make_runner()</code> .
run_log	A <code>reactiveVal</code> for log output.

**Value**

No return value. Called for its side effect of advancing the runner state and appending to `run_log`.

---

step_out_frame	<i>Step out of the current loop or if/else block (Step Out button)</i>
----------------	--

---

**Description**

Pops the innermost non-function frame, landing at the next statement in the enclosing scope. For example, when paused inside an inner for loop, clicking Step Out exits that loop and resumes at the next statement of the outer loop body. No-op when only the function frame remains on the stack.

**Usage**

```
step_out_frame(runner, run_log)
```

**Arguments**

runner	Runner from <code>make_runner()</code> .
run_log	A <code>reactiveVal</code> for log output.

**Value**

No return value. Called for its side effect of popping the innermost non-function frame and updating `run_log`.

---

stop_runner	<i>Stop execution and reset the runner (Stop button)</i>
-------------	--

---

**Description**

Stops any running or paused execution, clears the step stack, and marks the runner as ended. Logs a "Stopped by user." message when the runner was active.

**Usage**

```
stop_runner(runner, run_log)
```

**Arguments**

runner	Runner from <code>make_runner()</code> .
run_log	A <code>reactiveVal</code> for log output.

**Value**

No return value. Called for its side effect of resetting the runner state.

# Index

`continue_to_next_pause`, 2

`embeddedStepServer`, 3, 4, 5

`embeddedStepUI`, 3, 4

`make_runner`, 3, 5, 7

`run_demo`, 5

`run_program`, 3–5, 6, 6

`runApp`, 5

`soloStepServer`, 5, 6, 8

`soloStepUI`, 7, 8

`step_fn`, 9

`step_out_frame`, 9

`stop_runner`, 10