Package 'livelink'

July 24, 2026

Title Create Shareable Links for 'webR' and 'Shinylive' Environments

Version 0.1.0

Description Creates shareable links for 'R' code in 'WebAssembly' (WASM) Read-Eval-Print Loop (REPL) environments like 'webR' <https://webr.r-wasm.org/> and for 'Shiny' applications using 'Shinylive' <https://shinylive.io/>. Supports single scripts, multi-file projects, exercise and solution pairs, and batch processing. Includes encoding, decoding, and previewing of links for both 'R' and 'Python' environments.

License AGPL (>= 3)

URL https://r-pkg.thecoatlessprofessor.com/livelink/, https://github.com/coatless-rpkg/livelink

BugReports https://github.com/coatless-rpkg/livelink/issues

Depends R (>= 4.1.0)

Imports base64enc, cli, clipr, jsonlite, lzstring, RcppMsgPack, stats, tools, utils

Suggests knitr, quarto, testthat (>= 3.0.0), withr

Encoding UTF-8

SystemRequirements Quarto command line tools (https://github.com/quarto-dev/quarto-cli).

Config/Needs/website pkgdown

Config/testthat/edition 3

Collate 'as-data-frame.R' 'build-webr-repl-link.R' 'classes.R' 'decode-shinylive-link.R' 'utils.R' 'decode-webr-repl-link.R' 'extract-link.R' 'format-methods.R' 'knit-print.R' 'knitr-engine.R' 'options.R' 'process-input.R' 'shinylive-link.R' 'validators.R' 'webr-repl-directory-links.R' 'webr-repl-links.R'

VignetteBuilder quarto

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author James Joseph Balamuta [aut, cre, cph],
reprex authors [cph] (stringify_expression() and
trim_common_leading_ws(), adapted in R/process-input.R; see
inst/COPYRIGHTS)

Maintainer James Joseph Balamuta <james.balamuta@gmail.com>

Repository CRAN

Date/Publication 2026-07-24 10:20:02 UTC

Contents

as.character.webr_link . . . 3
as.data.frame.livelink . . . 4
decode_shinylive_link . . . 5
decode_webr_link . . . 6
format.livelink . . . 8
knit_print.livelink . . . 9
livelink-knitr . . . 10
preview_shinylive_link . . . 12
preview_webr_link . . . 13
print.shinylive_decoded . . . 14
print.shinylive_decoded_batch . . . 15
print.shinylive_directory . . . 15
print.shinylive_link . . . 16
print.shinylive_preview . . . 16
print.shinylive_project . . . 17
print.webr_decoded . . . 17
print.webr_decoded_batch . . . 18
print.webr_directory . . . 18
print.webr_exercise . . . 19
print.webr_link . . . 19
print.webr_preview . . . 20
print.webr_project . . . 20
repl_urls . . . 21
set_webr_base_url . . . 22
shinylive_directory . . . 23
shinylive_project . . . 24
shinylive_py_link . . . 25
shinylive_r_link . . . 26
webr_repl_directory . . . 28
webr_repl_exercise . . . 29
webr_repl_link . . . 31
webr_repl_project . . . 32

Index **35**

`as.character.webr_link` *Extract URLs as character vector*

Description

Extract URLs as character vector

Usage

```
## S3 method for class 'webr_link'
as.character(x, ...)

## S3 method for class 'shinylive_link'
as.character(x, ...)

## S3 method for class 'webr_project'
as.character(x, ...)

## S3 method for class 'webr_exercise'
as.character(x, ...)

## S3 method for class 'webr_directory'
as.character(x, ...)

## S3 method for class 'webr_decoded'
as.character(x, ...)

## S3 method for class 'webr_decoded_batch'
as.character(x, ...)

## S3 method for class 'webr_preview'
as.character(x, ...)

## S3 method for class 'shinylive_project'
as.character(x, ...)

## S3 method for class 'shinylive_directory'
as.character(x, ...)

## S3 method for class 'shinylive_decoded'
as.character(x, ...)

## S3 method for class 'shinylive_decoded_batch'
as.character(x, ...)

## S3 method for class 'shinylive_preview'
```

```
as.character(x, ...)
```

Arguments

x	Link object
...	Additional arguments

Value

A character vector of URLs. Most objects yield a single URL; exercise objects return a length-2 named vector (`exercise`, `solution`); directory and batch objects return one URL per file.

See Also

`repl_urls()` for the same extraction as a named generic you can call explicitly.

---

as.data.frame.livelink

Turn a livelink container into a data frame

Description

The container classes coerce to a tidy data frame, so a folder of links or a batch of decoded results can be tabulated, filtered, joined, or written to CSV with the tools you already use. For a tibble, wrap the result: `tibble::as_tibble(as.data.frame(x))`.

Usage

```
## S3 method for class 'webr_directory'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'shinylive_directory'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'webr_decoded_batch'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'shinylive_decoded_batch'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A `webr_directory`, `shinylive_directory`, `webr_decoded_batch`, or `shinylive_decoded_batch` object.
row.names	A character vector of row names, or `NULL`.
optional	Ignored; present for compatibility with the generic.
...	Ignored.

Value

A data frame. For a directory, one row per generated link with columns `filename` and `url`. For a decoded batch, one row per successfully decoded URL with columns `name`, `url`, `total_files`, `total_size`, and `output_dir` (URLs that failed to decode carry no result and are omitted; the object's `total_urls` and `successful_urls` fields hold the counts).

Examples

```
dir <- tempfile()
dir.create(dir)
writeLines("plot(1:10)",       file.path(dir, "one.R"))
writeLines("hist(rnorm(100))", file.path(dir, "two.R"))

links <- webr_repl_directory(dir)
as.data.frame(links)
```

---

decode_shinylive_link *Decode Shinylive link(s) to extract files to local directory*

Description

Decodes Shinylive sharelinks to extract the embedded files and save them to a local directory. This is the reverse operation of creating Shinylive links. Handles both single URLs and multiple URLs automatically.

Usage

```
decode_shinylive_link(
  url,
  output_dir = file.path(tempdir(), "shinylive_files"),
  overwrite = FALSE,
  create_subdir = TRUE,
  name_dirs = TRUE
)
```

Arguments

`url`	Character string or vector containing Shinylive URL(s)
`output_dir`	Character string specifying the output directory path. Defaults to a `shinylive_files` directory inside the session temporary directory; pass an explicit path to extract somewhere permanent.
`overwrite`	Logical. Whether to overwrite existing files (default: FALSE)
`create_subdir`	Logical. If `TRUE` (default), each decoded link is extracted into its own subdirectory under `output_dir` rather than directly into it. For a single URL the subdirectory is named `shinylive_<hash>`, where `<hash>` is a short fingerprint of the URL. For multiple URLs, see `name_dirs`. Set `FALSE` to extract straight

into `output_dir`; with multiple URLs this means identically named files (such as `app.R`) collide, so a later app overwrites an earlier one when `overwrite = TRUE`, or is skipped when `overwrite = FALSE`.

`name_dirs` Logical. For multiple URLs, controls how the per-link subdirectories are named: `TRUE` (default) numbers them `app_01`, `app_02`, ...; `FALSE` names each one `shinylive_<hash>` from the URL fingerprint. Ignored for a single URL, and ignored when `create_subdir = FALSE`.

Value

For a single URL, a `shinylive_decoded` object. For multiple URLs, a `shinylive_decoded_batch` object.

See Also

`preview_shinylive_link()` to inspect a link without writing files; `shinylive_r_link()` and `shinylive_py_link()` to create links.

Examples

```
# Round-trip: build a link, then decode it back to files
url <- as.character(shinylive_r_link("library(shiny)"))

result <- decode_shinylive_link(url)
print(result)

# Extract to a directory of your choosing
out <- file.path(tempdir(), "my_app")
decode_shinylive_link(url, output_dir = out, create_subdir = FALSE, overwrite = TRUE)
list.files(out)

# Both engines at once
urls <- c(url, as.character(shinylive_py_link("from shiny import App")))
decode_shinylive_link(urls, output_dir = file.path(tempdir(), "my_apps"))
```

---

decode_webr_link *Decode webR REPL link(s) to extract files to local directory*

Description

Decodes webR REPL sharelinks to extract the embedded files and save them to a local directory. This is the reverse operation of creating webR links. Handles both single URLs and multiple URLs automatically.

Usage

```
decode_webr_link(
  url,
  output_dir = file.path(tempdir(), "webr_files"),
  overwrite = FALSE,
  create_subdir = TRUE,
  name_dirs = TRUE
)
```

Arguments

url	Character string or vector containing webR REPL URL(s)
output_dir	Character string specifying the output directory path. Defaults to a webr_files directory inside the session temporary directory; pass an explicit path to extract somewhere permanent.
overwrite	Logical. Whether to overwrite existing files (default: FALSE)
create_subdir	Logical. If TRUE (default), each decoded link is extracted into its own subdirectory under output_dir rather than directly into it. For a single URL the subdirectory is named webr_<hash>, where <hash> is a short fingerprint of the URL. For multiple URLs, see name_dirs. Set FALSE to extract straight into output_dir.
name_dirs	Logical. For multiple URLs, controls how the per-link subdirectories are named: TRUE (default) numbers them script_01, script_02, ...; FALSE names each one webr_<hash> from the URL fingerprint. Ignored for a single URL, and ignored when create_subdir = FALSE (all files then extract into output_dir).

Value

For a single URL, a webr_decoded object. For multiple URLs, a webr_decoded_batch object.

See Also

preview_webr_link() to inspect a link without writing files, and webr_repl_link(), webr_repl_project(), and webr_repl_exercise(), which create the links this function decodes.

Examples

```
# Round-trip: build a link, then decode it back to files
url <- as.character(webr_repl_link("plot(1:10)"))

result <- decode_webr_link(url)
print(result)

# Extract to a directory of your choosing
out <- file.path(tempdir(), "my_code")
decode_webr_link(url, output_dir = out, create_subdir = FALSE, overwrite = TRUE)
list.files(out)

# Several links at once
```

```
urls <- c(url, as.character(webr_repl_link("hist(rnorm(100))")))
decode_webr_link(urls, output_dir = file.path(tempdir(), "my_scripts"))
```

`format.livelink` *Format a livelink object as a character vector*

Description

Returns an object's printed representation as a character vector, one element per line, so it can be captured, logged, pasted into a report, or otherwise reused. `print()` renders the very same content to the console; `format()` hands it back to you instead.

Usage

```
## S3 method for class 'webr_link'
format(x, ...)

## S3 method for class 'webr_project'
format(x, ...)

## S3 method for class 'webr_exercise'
format(x, ...)

## S3 method for class 'webr_directory'
format(x, ...)

## S3 method for class 'webr_decoded'
format(x, ...)

## S3 method for class 'webr_decoded_batch'
format(x, ...)

## S3 method for class 'webr_preview'
format(x, ...)

## S3 method for class 'shinylive_link'
format(x, ...)

## S3 method for class 'shinylive_project'
format(x, ...)

## S3 method for class 'shinylive_directory'
format(x, ...)

## S3 method for class 'shinylive_decoded'
format(x, ...)
```

```
## S3 method for class 'shinylive_decoded_batch'
format(x, ...)

## S3 method for class 'shinylive_preview'
format(x, ...)
```

Arguments

x	A livelink object (a link, project, exercise, directory, decoded result, batch, or preview).
...	Passed to the object's `print()` method, so preview-specific options such as `show_content` work here too.

Value

A character vector of formatted lines.

Examples

```
link <- webr_repl_link("plot(1:10)")
format(link)
writeLines(format(link))
```

`knit_print.livelink` *Render livelink objects as links in knitted documents*

Description

'knitr' calls `knit_print()` on the last value of a chunk. Without these methods a link object would fall back to `print()`, dumping cli console output (a header, box glyphs, metadata) into the rendered page. These methods instead emit a clickable Markdown link, which is almost always what you want when a link object is the visible result of a chunk.

A single link becomes `[Open in webR](url)` (or Shinylive); a project the same. A directory or an exercise, which carry several named URLs, become a bulleted list, one titled link per entry.

These fire only inside 'knitr'. Call `print()` explicitly for the full cli description, or `as.character()` for the bare URL.

Usage

```
## S3 method for class 'webr_link'
knit_print(x, ...)

## S3 method for class 'webr_project'
knit_print(x, ...)

## S3 method for class 'webr_exercise'
```

```
knit_print(x, ...)

## S3 method for class 'webr_directory'
knit_print(x, ...)

## S3 method for class 'shinylive_link'
knit_print(x, ...)

## S3 method for class 'shinylive_project'
knit_print(x, ...)

## S3 method for class 'shinylive_directory'
knit_print(x, ...)
```

Arguments

x	A livelink object (a link, project, exercise, or directory).
...	Ignored.

Value

A `knit_asis` object (via `knitr::asis_output()`).

See Also

livelink-knitr for the chunk hook and engine, `format.livelink()` and `as.character.webr_link()` for other renderings, and `vignette("links-in-documents", package = "livelink")`.

`livelink-knitr` *Turn document chunks into shareable links*

Description

livelink plugs into 'knitr' two ways, and which you want depends on one question: **should the code also run in your document?**

A chunk hook, on an ordinary `r` chunk. The chunk runs as usual – its output, plots and all, appear in the rendered page – and a link is added underneath. Use this for code you want your reader to *see the result of* and *also* be able to open and play with.

````
```{r}
#| livelink: true
#| autorun: true
# Load the data
data(mtcars)
plot(mtcars$mpg, mtcars$wt)
```
````

An engine, as ```{livelink}. The chunk is displayed but **not** run: only the link is produced. Use this for code your session cannot or should not execute – a Shiny app, something needing a package you have not installed, or anything slow.

````
```{livelink}
#| engine.target: shinylive-r
library(shiny)
shinyApp(fluidPage(), function(input, output) {})
```
````

There is deliberately no {shinylive-r} or {shinylive-py} engine. 'knitr' cannot dispatch a chunk whose engine name contains a hyphen (its chunk syntax forbids it), and in Quarto such a cell is handed to the Shinylive extension rather than to 'knitr'. Name Shinylive through `engine.target` instead.

Usage

```
use_livelink_hook()

use_livelink_engine()
```

Details

Reach for either of these rather than expression input (`webr_repl_link({ ... })`) inside a knitted document. 'knitr' evaluates chunks through `evaluate::evaluate()`, which discards source references, and comments live in those – so comments inside a `{ }` expression are silently dropped from the link when the document renders, and no `keep.source` setting brings them back. Both the hook and the engine are handed the chunk's verbatim source, so nothing is lost.

Both are registered automatically when livelink is loaded, provided 'knitr' is installed. Call these yourself only if you have reset `knitr::knit_hooks` or `knitr::knit_engines`.

Value

Called for their side effect. Invisibly `TRUE` if registration happened, `FALSE` if 'knitr' is not installed.

Chunk options

`livelink` Hook only. `true` for a webR link, or name the target directly: `"webr"`, `"shinylive-r"`, `"shinylive-py"`.

`engine.target` Engine only. `"webr"` (default), `"shinylive-r"`, or `"shinylive-py"`.

`autorun` Logical. Run the code as soon as the link opens. webR only.

`panels` Character vector of webR panels, e.g. `c("editor", "plot")`.

`mode` Shinylive only. Display mode, `"editor"` (default) or `"app"`.

`filename` webR only. Name for the script file in the webR virtual filesystem (default `"script.R"`); must end in `.R` for `autorun` to work.

`link.text` Text for the hyperlink. Defaults to `"Open in webR"` or `"Open in Shinylive"`.

`link.only` Engine only. If `TRUE`, emit the link without the source.

Setting options once

These are ordinary 'knitr' chunk options, so `opts_chunk` sets them for a whole document, and a single chunk opts out with `livelink: false`:

```
knitr::opts_chunk$set(livelink = TRUE, autorun = TRUE)
```

`echo` does not gate the link

It is natural to assume the code must be visible for a link to be made. It need not be. `echo` controls whether the **source is shown in your page**; the link is built from the chunk's source, which 'knitr' hands over either way. So `echo: false` gives a working link whose code the reader simply cannot see.

`eval: false` is the other half: the chunk is displayed but not run, which makes an `r` chunk behave rather like the engine.

See Also

`vignette("links-in-documents", package = "livelink")` for the whole picture, and `webr_repl_link()` for why a braced expression loses comments in a knitted document.

Examples

```
# Both are registered on load; call directly only after resetting knitr's hooks.
use_livelink_hook()
use_livelink_engine()
```

`preview_shinylive_link` *Preview Shinylive link contents without writing files to disk*

Description

Decodes a Shinylive URL and returns information about the embedded files without actually saving them to disk. Use print method options to control display.

Usage

```
preview_shinylive_link(url)
```

Arguments

`url`	Character string containing the Shinylive URL

Value

shinylive_preview object with file information and metadata

See Also

decode_shinylive_link() to extract files to disk; shinylive_r_link() and shinylive_py_link() to create links.

Examples

```
url <- as.character(shinylive_r_link("library(shiny)"))

# Inspect a link without writing anything to disk
preview <- preview_shinylive_link(url)

# Default print (no content)
print(preview)

# Show file contents
print(preview, show_content = TRUE)

# Show file contents with custom length limit
print(preview, show_content = TRUE, max_content_length = 200)

# Access the preview data
preview$files_data
preview$total_files
```

---

preview_webr_link *Preview webR REPL link contents without writing files to disk*

Description

Decodes a webR URL and returns information about the embedded files without actually saving them to disk. Use print method options to control display.

Usage

```
preview_webr_link(url)
```

Arguments

url	Character string containing the webR URL

Value

A webr_preview object (a list) with elements including files_data (the embedded files), total_files, total_size, mode, version, flags, and autorun_files. Its print() method accepts show_content and max_content_length to display file bodies.

See Also

decode_webr_link() to extract the embedded files to disk once you are happy with the preview.

Examples

```
url <- as.character(webr_repl_link("plot(1:10)"))

# Inspect a link without writing anything to disk
preview <- preview_webr_link(url)

# Default print (no content)
print(preview)

# Show file contents
print(preview, show_content = TRUE)

# Show file contents with custom length limit
print(preview, show_content = TRUE, max_content_length = 200)

# Access the preview data
preview$files_data
preview$total_files
```

---

print.shinylive_decoded

Print method for shinylive_decoded objects

Description

Print method for shinylive_decoded objects

Usage

```
## S3 method for class 'shinylive_decoded'
print(x, ...)
```

Arguments

x	shinylive_decoded object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.shinylive_decoded_batch *Print method for shinylive_decoded_batch objects*

Description

Print method for shinylive_decoded_batch objects

Usage

```
## S3 method for class 'shinylive_decoded_batch'
print(x, ...)
```

Arguments

x	shinylive_decoded_batch object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.shinylive_directory *Print method for shinylive_directory objects*

Description

Print method for shinylive_directory objects

Usage

```
## S3 method for class 'shinylive_directory'
print(x, ...)
```

Arguments

x	shinylive_directory object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.shinylive_link *Print method for shinylive_link objects*

Description

Print method for shinylive_link objects

Usage

```
## S3 method for class 'shinylive_link'
print(x, ...)
```

Arguments

x	shinylive_link object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.shinylive_preview *Print method for shinylive_preview objects*

Description

Print method for shinylive_preview objects

Usage

```
## S3 method for class 'shinylive_preview'
print(x, show_content = FALSE, max_content_length = 500, ...)
```

Arguments

x	shinylive_preview object
show_content	Logical. Whether to print the contents of each file (default: FALSE)
max_content_length	Maximum number of characters of content to show per file (default: 500)
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.shinylive_project *Print method for shinylive_project objects*

Description

Print method for shinylive_project objects

Usage

```
## S3 method for class 'shinylive_project'
print(x, ...)
```

Arguments

x	shinylive_project object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_decoded *Print method for webr_decoded objects*

Description

Print method for webr_decoded objects

Usage

```
## S3 method for class 'webr_decoded'
print(x, ...)
```

Arguments

x	webr_decoded object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_decoded_batch *Print method for webr_decoded_batch objects*

Description

Print method for webr_decoded_batch objects

Usage

```
## S3 method for class 'webr_decoded_batch'
print(x, ...)
```

Arguments

x	webr_decoded_batch object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_directory *Print method for webr_directory objects*

Description

Print method for webr_directory objects

Usage

```
## S3 method for class 'webr_directory'
print(x, ...)
```

Arguments

x	webr_directory object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_exercise *Print method for webr_exercise objects*

Description

Print method for webr_exercise objects

Usage

```
## S3 method for class 'webr_exercise'
print(x, ...)
```

Arguments

x	webr_exercise object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_link *Print method for webr_link objects*

Description

Print method for webr_link objects

Usage

```
## S3 method for class 'webr_link'
print(x, ...)
```

Arguments

x	webr_link object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_preview *Print method for webr_preview objects*

Description

Print method for webr_preview objects

Usage

```
## S3 method for class 'webr_preview'
print(x, show_content = FALSE, max_content_length = 500, ...)
```

Arguments

x	webr_preview object
show_content	Logical. Whether to print the contents of each file (default: FALSE)
max_content_length	Maximum number of characters of content to show per file (default: 500)
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.webr_project *Print method for webr_project objects*

Description

Print method for webr_project objects

Usage

```
## S3 method for class 'webr_project'
print(x, ...)
```

Arguments

x	webr_project object
...	Additional arguments (ignored)

Value

Invisibly returns the object

repl_urls *Extract shareable URLs from livelink objects*

Description

Generic function to extract the shareable URL(s) from any livelink object, covering both webR REPL and Shinylive results. Provides a clear way to get just the URLs for sharing or further processing.

Usage

```
repl_urls(x, ...)

## Default S3 method:
repl_urls(x, ...)
```

Arguments

x	A livelink object. Supported classes are webr_link, webr_project, webr_exercise, webr_directory, webr_decoded, webr_decoded_batch, webr_preview, shinylive_link, shinylive_project, shinylive_directory, shinylive_decoded, shinylive_decoded_batch, and shinylive_preview.
...	Additional arguments passed to methods

Value

A character vector of URLs. Most objects yield a single URL; exercise objects return a length-2 named vector (exercise, solution); directory and batch objects return one URL per file.

See Also

The as.character() methods (for example as.character.webr_link()), which repl_urls() delegates to.

Examples

```
# Single link
link <- webr_repl_link("plot(1:10)")
repl_urls(link)

# Exercise (returns named vector)
exercise <- webr_repl_exercise("# TODO", "plot(1:10)", "test")
repl_urls(exercise)

# Shinylive links work the same way
repl_urls(shinylive_r_link("library(shiny)"))

# Decoded files (returns the original URL)
```

```
decoded <- decode_webr_link(as.character(link))
repl_urls(decoded)
```

`set_webr_base_url` *Set global base URL for webR links*

Description

Overrides the base URL used when building webR REPL links, for instance to point at a self-hosted or pinned webR deployment. The value is stored in the `livelink.base_url` option and applies to webR links only; Shinylive links are unaffected. Once set, a custom base URL takes precedence over the `version` argument passed to the link builders.

Usage

```
set_webr_base_url(base_url = NULL)
```

Arguments

`base_url`	Custom base URL to use for all webR links

Value

Invisibly returns the `base_url` value (or `NULL` if resetting to default).

Examples

```
# Remember the current setting so it can be restored afterwards
old <- getOption("livelink.base_url")

# Set custom base URL
set_webr_base_url("https://my-custom-webr.com/")

# Reset to default (removes custom setting)
set_webr_base_url(NULL)

# Restore the previous setting
set_webr_base_url(old)
```

shinylive_directory *Create Shinylive sharelinks from a directory of Shiny apps*

Description

Batch processes directories containing Shiny applications to create individual Shinylive links. Each subdirectory is treated as a separate Shiny app project.

Only text files with extensions .R, .py, .txt, .md, .csv, .json, .yaml, or .yml are embedded in a link. Other files (for example images or binary data) are skipped with a warning.

Usage

```
shinylive_directory(
  directory_path,
  engine,
  mode = "editor",
  header = TRUE,
  app_file = NULL,
  base_url = NULL
)
```

Arguments

directory_path	Character string specifying the path to the directory containing Shiny app directories
engine	Engine to use: "r" for R Shiny or "python" for Python Shiny
mode	Shinylive display mode (default `"editor"`). `"editor"` shows an editable code panel beside the running app; `"app"` shows only the running app.
header	Logical, whether to show the Shinylive header bar. It applies only when `mode = "app"` and is ignored in the default `"editor"` mode. Defaults to `TRUE`.
app_file	Main app filename to look for (default: "app.R" for R, "app.py" for Python)
base_url	Custom Shinylive base URL. If NULL (default), links point at https://shinylive.io.

Value

shinylive_directory object containing URLs and metadata for all found apps

Examples

```
# Each app lives in its own subdirectory:
#   shiny_apps/
#     app1/app.R
#     app2/app.R
shiny_apps <- tempfile()
dir.create(file.path(shiny_apps, "app1"), recursive = TRUE)
dir.create(file.path(shiny_apps, "app2"), recursive = TRUE)
```

```
writeLines("library(shiny)", file.path(shiny_apps, "app1", "app.R"))
writeLines("library(shiny)", file.path(shiny_apps, "app2", "app.R"))

links <- shinylive_directory(shiny_apps, engine = "r", mode = "editor")
print(links)

# Extract just the URLs
repl_urls(links)
```

`shinylive_project` *Create a Shinylive sharelink for multi-file projects*

Description

Creates Shinylive projects for either R or Python from named lists or file path vectors.

Usage

```
shinylive_project(
  input,
  engine,
  mode = "editor",
  header = TRUE,
  base_url = NULL
)
```

Arguments

`input`	Input for multiple files. Can be: • Named list: `list("app.R" = code1, "utils.R" = code2)` • Vector of file paths: `c("app.R", "utils.R", "data.csv")`
`engine`	Engine to use: "r" for R Shiny or "python" for Python Shiny
`mode`	Shinylive display mode (default `"editor"`). `"editor"` shows an editable code panel beside the running app; `"app"` shows only the running app.
`header`	Logical, whether to show the Shinylive header bar. It applies only when `mode = "app"` and is ignored in the default `"editor"` mode. Defaults to `TRUE`.
`base_url`	Custom Shinylive base URL. If NULL (default), links point at https://shinylive.io.

Value

shinylive_project object containing the Shinylive URL and metadata

See Also

`shinylive_r_link()` and `shinylive_py_link()` for single apps; `decode_shinylive_link()` and `preview_shinylive_link()` to read a link back.

Examples

```
# Named list input
files <- list(
  "app.R" = "library(shiny)\nshinyApp(fluidPage(), function(i, o) {})",
  "utils.R" = "# Utility functions"
)
shinylive_project(files, engine = "r", mode = "editor")

# File paths input
project_dir <- tempfile()
dir.create(project_dir)
app <- file.path(project_dir, "app.R")
utils <- file.path(project_dir, "utils.R")
writeLines("library(shiny)", app)
writeLines("# utils", utils)
shinylive_project(c(app, utils), engine = "r")
```

---

shinylive_py_link *Create a Shinylive sharelink for Python Shiny apps*

Description

Generates a shareable URL for Python Shiny applications that can run in the browser using Shinylive. Supports character strings, file paths, named lists, and clipboard input.

Usage

```
shinylive_py_link(
  input = NULL,
  mode = "editor",
  header = TRUE,
  base_url = NULL
)
```

Arguments

input	App input. Can be: • Character string: Python code for the app • File path: Path to app.py file • Vector of file paths: Multiple files for the app • Named list: `list("app.py" = code1, "utils.py" = code2)` • NULL: Read from clipboard
mode	Shinylive display mode (default `"editor"`). `"editor"` shows an editable code panel beside the running app; `"app"` shows only the running app.
header	Logical, whether to show the Shinylive header bar. It applies only when `mode = "app"` and is ignored in the default `"editor"` mode. Defaults to `TRUE`.
base_url	Custom Shinylive base URL. If NULL (default), links point at https://shinylive.io.

Value

shinylive_link object containing the Shinylive URL and metadata

See Also

shinylive_project() for multi-file apps; decode_shinylive_link() and preview_shinylive_link() to read a link back.

Examples

```
# String input
app_code <- "
from shiny import App, render, ui
app_ui = ui.page_fluid(ui.h2('Hello World'))
def server(input, output, session): pass
app = App(app_ui, server)
"
shinylive_py_link(app_code)

# Multiple files as a named list
shinylive_py_link(list(
  "app.py" = app_code,
  "utils.py" = "def helper(): return 42"
))

# File path input
app_dir <- tempfile()
dir.create(app_dir)
app_path <- file.path(app_dir, "app.py")
writeLines("from shiny import App, ui", app_path)
shinylive_py_link(app_path)

# Read the app from the clipboard
if (interactive()) {
  shinylive_py_link()
}
```

---

shinylive_r_link *Create a Shinylive sharelink for R Shiny apps*

Description

Generates a shareable URL for R Shiny applications that can run in the browser using Shinylive. Supports expressions, character strings, file paths, named lists, and clipboard input.

Usage

```
shinylive_r_link(input = NULL, mode = "editor", header = TRUE, base_url = NULL)
```

Arguments

input	App input. Can be: • R expression (no quotes needed): `shinylive_r_link({ shinyApp(ui, server) })` • Character string: R code for the app • File path: Path to app.R file • Vector of file paths: Multiple files for the app • Named list: `list("app.R" = code1, "utils.R" = code2)` • NULL: Read from clipboard
mode	Shinylive display mode (default `"editor"`). `"editor"` shows an editable code panel beside the running app; `"app"` shows only the running app.
header	Logical, whether to show the Shinylive header bar. It applies only when `mode = "app"` and is ignored in the default `"editor"` mode. Defaults to `TRUE`.
base_url	Custom Shinylive base URL. If NULL (default), links point at https://shinylive.io.

Value

shinylive_link object containing the Shinylive URL and metadata

Comments in expression input

Comments inside a `{ }` expression are recovered from R's source references, so they survive interactively but are dropped inside a knitted 'Quarto' or 'R Markdown' document. Pass a string or a file path, or use the `livelink` chunk engine, if you need them preserved. See `webr_repl_link()` for the details.

See Also

`shinylive_project()` for multi-file apps; `decode_shinylive_link()` and `preview_shinylive_link()` to read a link back; livelink-knitr to give a document chunk its own link; `vignette("webr-and-shinylive", package = "livelink")` for the guide.

Examples

```
# Expression input (no quotes needed!)
shinylive_r_link({
  ui <- fluidPage(titlePanel("Hello World"))
  server <- function(input, output) {}
  shinyApp(ui, server)
})

# Multiple files as a named list
shinylive_r_link(list(
  "app.R" = "library(shiny)\nshinyApp(fluidPage(), function(i, o) {})",
  "utils.R" = "helper <- function() 42"
))

# File path input
```

```
app_dir <- tempfile()
dir.create(app_dir)
app_path <- file.path(app_dir, "app.R")
writeLines("library(shiny)", app_path)
shinylive_r_link(app_path)

# Read the app from the clipboard
if (interactive()) {
  shinylive_r_link()
}
```

`webr_repl_directory` *Create webR REPL sharelinks from a directory of R files*

Description

Batch processes all R files in a directory. By default each file becomes its own webR sharelink; with `single_link = TRUE` the whole directory is bundled into one link instead, exactly as `webr_repl_project()` would. Useful for converting collections of scripts, examples, or course materials.

Usage

```
webr_repl_directory(
  directory_path,
  autorun = FALSE,
  single_link = FALSE,
  pattern = "\\.R$",
  base_path = "/home/web_user/",
  panels = NULL,
  version = "latest",
  base_url = NULL
)
```

Arguments

`directory_path`	Character string specifying the path to the directory containing R files
`autorun`	Logical. Whether to enable autorun for all generated links (default: FALSE). With `single_link = TRUE`, this runs every R file in the bundle on arrival.
`single_link`	Logical. If `FALSE` (default), each matched file becomes its own link and the result is a `webr_directory`. If `TRUE`, all matched files are packed into one link and the result is a single `webr_project`.
`pattern`	Regular expression matched against file names in `directory_path`. Defaults to `"\\.R$"`, i.e. files ending in `.R`.
`base_path`	Base directory path for files in webR (default: "/home/web_user/")

panels	Character vector or string specifying which webR interface panels to show. Valid panels: "plot", "files", "terminal", "editor". Can be c("plot", "files") or "plot-files". If NULL (default), shows all panels.
version	webR version to use ("latest" or specific version >= "v0.5.4")
base_url	webR application URL. If NULL, uses global option or builds from version

Value

By default a webr_directory object, whose urls element is a named character vector mapping each filename to its webR sharelink. With single_link = TRUE, a single webr_project object bundling every matched file into one link.

See Also

webr_repl_project(), which bundles a named list or a vector of file paths into one link.

Examples

```
# A directory of R scripts
examples <- tempfile()
dir.create(examples)
writeLines("plot(1:10)", file.path(examples, "plot.R"))
writeLines("hist(rnorm(100))", file.path(examples, "hist.R"))

links <- webr_repl_directory(examples, autorun = TRUE)
print(links)

# Bundle the whole directory into one link instead
webr_repl_directory(examples, single_link = TRUE, panels = c("editor", "plot"))

# Show only the editor and terminal panels
webr_repl_directory(examples, panels = c("editor", "terminal"))

# Match a subset of files
webr_repl_directory(examples, pattern = "^plot")

# The URLs, named by file
repl_urls(links)
```

---

webr_repl_exercise *Create paired exercise and solution webR REPL links*

Description

Generates a pair of webR links for educational purposes: one for student exercises (without autorun) and one for solutions (with autorun enabled).

Usage

```
webr_repl_exercise(
  exercise_text,
  solution_text,
  exercise_name,
  base_path = "/home/web_user/",
  version = "latest",
  base_url = NULL
)
```

Arguments

exercise_text	Character string containing the exercise code with placeholders or TODOs
solution_text	Character string containing the complete solution code
exercise_name	Base name for the exercise (will create "name_exercise.R" and "name_solution.R")
base_path	Base directory path for files (default: "/home/web_user/")
version	webR version to use ("latest" or specific version >= "v0.5.4")
base_url	webR application URL. If NULL, uses global option or builds from version

Value

webr_exercise object holding the paired exercise and solution links

See Also

webr_repl_link(), which this builds on; vignette("teaching", package = "livelink") for using links in a course.

Examples

```
exercise_code <- "
# Exercise: Calculate mean of mtcars$mpg
# TODO: Complete the line below
mean_mpg <- 0
print(mean_mpg)
"

solution_code <- "
# Solution: Calculate mean of mtcars$mpg
mean_mpg <- mean(mtcars$mpg)
print(mean_mpg)
"

links <- webr_repl_exercise(exercise_code, solution_code, "basic_stats")
links$exercise
links$solution

# Custom path and version
webr_repl_exercise(exercise_code, solution_code, "stats",
```

```
base_path = "/exercises/", version = "v0.5.4")
```

`webr_repl_link` *Create a webR REPL sharelink from R code*

Description

Generates a shareable URL for R code that can be executed in the webR environment. Supports expressions, file paths, character strings, and clipboard input.

Usage

```
webr_repl_link(
  input = NULL,
  filename = "script.R",
  path = NULL,
  autorun = FALSE,
  panels = NULL,
  version = "latest",
  base_url = NULL
)
```

Arguments

`input`	Code input. Can be: • R expression (no quotes needed): `webr_repl_link({ plot(1:10) })` • Character string: R code to execute • File path: Path to R file to read • NULL: Read from clipboard (requires clipr package)
`filename`	Name for the file (default: `"script.R"`)
`path`	Full path where the file will be placed in webR. If NULL (default), the file is placed at `"/home/web_user/{filename}"`.
`autorun`	Logical. Whether to auto-execute the code when link is opened (default: `FALSE`). Only R files (`.R`) can be auto-executed.
`panels`	Character vector or string specifying which webR interface panels to show. Valid panels: `"plot"`, `"files"`, `"terminal"`, `"editor"`. Can be `c("plot", "files")` or `"plot-files"`. If NULL (default), shows all panels.
`version`	webR version to use (`"latest"` or specific version >= "v0.5.4")
`base_url`	webR application URL. If NULL, uses global option or builds from version

Value

webr_link object containing the webR sharelink and metadata

Comments in expression input

Expression input recovers your source from R's *source references*, which R only attaches when `keep.source` is enabled. Comments therefore survive in an interactive session, but are dropped when the calling code is parsed without source references – notably inside a knitted 'Quarto' or 'R Markdown' document, because 'knitr' evaluates chunks through `evaluate::evaluate()`, which discards them. No `keep.source` setting recovers them there.

If you need comments preserved, pass the code as a string or a file path, or write it as a chunk in the document – see livelink-knitr and `vignette("links-in-documents", package = "livelink")`.

See Also

`webr_repl_project()` for multi-file projects and `webr_repl_exercise()` for exercise and solution pairs; livelink-knitr to give a document chunk its own link; `vignette("getting-started", package = "livelink")` for an introduction.

Examples

```
# Expression input (no quotes needed!)
webr_repl_link({
  plot(1:10)
  summary(mtcars)
})

# Traditional string input
webr_repl_link("plot(1:10)")

# Choose which panels the REPL shows
webr_repl_link({ hist(rnorm(100)) }, panels = c("plot", "editor"))

# Run the code as soon as the link opens
webr_repl_link("plot(1:10)", autorun = TRUE)

# File path input
script <- tempfile(fileext = ".R")
writeLines("plot(1:10)", script)
webr_repl_link(script)

# Read the code from the clipboard
if (interactive()) {
  webr_repl_link()
}
```

---

webr_repl_project *Create webR REPL sharelink for multiple files*

Description

Creates a webR sharelink for projects with multiple R files, data files, or other resources. Supports named lists and file path vectors as input.

Usage

```
webr_repl_project(
  input,
  autorun_files = character(0),
  base_path = "/home/web_user/",
  panels = NULL,
  version = "latest",
  base_url = NULL
)
```

Arguments

input	Input for multiple files. Can be: • Named list of braced expressions, so each file is written as R rather than as a string full of escaped newlines: `list("main.R" = { plot(1:10) }, "utils.R" = { f <- function() 42 })` • Named list of strings: `list("main.R" = code1, "utils.R" = code2)` • Vector of file paths: `c("main.R", "utils.R", "data.csv")` The two list forms mix freely, which is what you want for a project holding both code and, say, a `README.md`.
autorun_files	Character vector of filenames to auto-execute when project loads, or "all" to autorun all R files (default: none)
base_path	Base directory path for all files (default: `"/home/web_user/"`)
panels	Character vector or string specifying which webR interface panels to show. Valid panels: `"plot"`, `"files"`, `"terminal"`, `"editor"`. Can be `c("plot", "files")` or `"plot-files"`. If NULL (default), shows all panels.
version	webR version to use (`"latest"` or specific version >= "v0.5.4")
base_url	webR application URL. If NULL, uses global option or builds from version

Value

webr_project object containing the webR sharelink for the multi-file project

Writing a project as code

A file's contents can be given as a `{ ... }` block instead of a string, which spares you escaping every newline and quote:

```
webr_repl_project(list(
  "main.R"    = { source("utils.R"); summarise(mtcars) },
  "utils.R"   = { summarise <- function(d) summary(d) },
  "README.md" = "# Analysis"
))
```

The blocks are **never evaluated** – they are source to ship, not code to run – so an assignment inside one leaves nothing behind in your session.

Two things to know. Comments inside `{ }` survive in an interactive session but not in a knitted document (see `webr_repl_link()`). And a `library()` call inside a block is visible to `R CMD check`, which will report the package as an undeclared dependency of *yours*; in a vignette or an example, use a string for code that loads packages.

See Also

`webr_repl_link()` for the single-file case.

Examples

```
# Each file written as R, rather than as an escaped string
webr_repl_project(list(
  "main.R"  = { source("utils.R"); summarise(mtcars) },
  "utils.R" = { summarise <- function(d) summary(d) }
))

# Strings still work, and the two forms mix
files <- list(
  "main.R" = "source('utils.R')\nresult <- analyze_data(mtcars)",
  "utils.R" = "analyze_data <- function(data) { summary(data) }",
  "README.md" = "# My Analysis\nThis project analyzes the mtcars dataset."
)
webr_repl_project(files, autorun_files = "main.R")

# Autorun every R file in the project
webr_repl_project(files, autorun_files = "all")

# File paths input
project_dir <- tempfile()
dir.create(project_dir)
main <- file.path(project_dir, "main.R")
utils <- file.path(project_dir, "utils.R")
writeLines("source('utils.R')", main)
writeLines("# utils", utils)
webr_repl_project(c(main, utils))
```

Index

as.character(), *9*
as.character.shinylive_decoded
(as.character.webr_link), 3
as.character.shinylive_decoded_batch
(as.character.webr_link), 3
as.character.shinylive_directory
(as.character.webr_link), 3
as.character.shinylive_link
(as.character.webr_link), 3
as.character.shinylive_preview
(as.character.webr_link), 3
as.character.shinylive_project
(as.character.webr_link), 3
as.character.webr_decoded
(as.character.webr_link), 3
as.character.webr_decoded_batch
(as.character.webr_link), 3
as.character.webr_directory
(as.character.webr_link), 3
as.character.webr_exercise
(as.character.webr_link), 3
as.character.webr_link, 3
as.character.webr_link(), *10*, *21*
as.character.webr_preview
(as.character.webr_link), 3
as.character.webr_project
(as.character.webr_link), 3
as.data.frame.livelink, 4
as.data.frame.shinylive_decoded_batch
(as.data.frame.livelink), 4
as.data.frame.shinylive_directory
(as.data.frame.livelink), 4
as.data.frame.webr_decoded_batch
(as.data.frame.livelink), 4
as.data.frame.webr_directory
(as.data.frame.livelink), 4

decode_shinylive_link, 5
decode_shinylive_link(), *13*, *24*, *26, 27*
decode_webr_link, 6
decode_webr_link(), *14*

format.livelink, 8
format.livelink(), *10*
format.shinylive_decoded
(format.livelink), 8
format.shinylive_decoded_batch
(format.livelink), 8
format.shinylive_directory
(format.livelink), 8
format.shinylive_link
(format.livelink), 8
format.shinylive_preview
(format.livelink), 8
format.shinylive_project
(format.livelink), 8
format.webr_decoded (format.livelink), 8
format.webr_decoded_batch
(format.livelink), 8
format.webr_directory
(format.livelink), 8
format.webr_exercise (format.livelink),
8
format.webr_link (format.livelink), 8
format.webr_preview (format.livelink), 8
format.webr_project (format.livelink), 8

knit_print.livelink, 9
knit_print.shinylive_directory
(knit_print.livelink), 9
knit_print.shinylive_link
(knit_print.livelink), 9
knit_print.shinylive_project
(knit_print.livelink), 9
knit_print.webr_directory
(knit_print.livelink), 9
knit_print.webr_exercise
(knit_print.livelink), 9
knit_print.webr_link
(knit_print.livelink), 9

knit_print.webr_project
(knit_print.livelink), 9
knitr::asis_output(), *10*

livelink-knitr, *10*, 10, *27*, *32*

preview_shinylive_link, 12
preview_shinylive_link(), *6*, *24*, *26, 27*
preview_webr_link, 13
preview_webr_link(), *7*
print(), *8, 9*
print.shinylive_decoded, 14
print.shinylive_decoded_batch, 15
print.shinylive_directory, 15
print.shinylive_link, 16
print.shinylive_preview, 16
print.shinylive_project, 17
print.webr_decoded, 17
print.webr_decoded_batch, 18
print.webr_directory, 18
print.webr_exercise, 19
print.webr_link, 19
print.webr_preview, 20
print.webr_project, 20

repl_urls, 21
repl_urls(), *4*

set_webr_base_url, 22
shinylive_directory, 23
shinylive_project, 24
shinylive_project(), *26, 27*
shinylive_py_link, 25
shinylive_py_link(), *6*, *13*, *24*
shinylive_r_link, 26
shinylive_r_link(), *6*, *13*, *24*

use_livelink_engine (livelink-knitr), 10
use_livelink_hook (livelink-knitr), 10

webr_repl_directory, 28
webr_repl_exercise, 29
webr_repl_exercise(), *7*, *32*
webr_repl_link, 31
webr_repl_link(), *7*, *12*, *27*, *30*, *34*
webr_repl_project, 32
webr_repl_project(), *7*, *28, 29*, *32*