

Package: rJavaEnv (via r-universe)

October 17, 2026

Title 'Java' Environments for R Projects

Version 0.3.0.9000

Description Quickly install 'Java Development Kit (JDK)' without administrative privileges and set environment variables in current R session or project to solve common issues with 'Java' environment management in 'R'. Recommended to users of 'Java'/'rJava'-dependent 'R' packages such as 'r5r', 'opentriplanner', 'xlsx', 'openNLP', 'rWeka', 'RJDBC', 'tabulapdf', and many more. 'rJavaEnv' prevents common problems like 'Java' not found, 'Java' version conflicts, missing 'Java' installations, and the inability to install 'Java' due to lack of administrative privileges. 'rJavaEnv' automates the download, installation, and setup of the 'Java' on a per-project basis by setting the relevant 'JAVA_HOME' in the current 'R' session or the current working directory (via '.Rprofile', with the user's consent). Similar to what 'renv' does for 'R' packages, 'rJavaEnv' allows different 'Java' versions to be used across different projects, but can also be configured to allow multiple versions within the same project (e.g. with the help of 'targets' package). For users who need to install 'rJava' or other 'Java'-dependent packages from source, 'rJavaEnv' will display a message with instructions on how to run 'R CMD javareconf' to make the 'Java' configuration permanent, but also provides a function 'java_build_env_set' that sets the environment variables in the current R session temporarily to allow installation of 'rJava' from source without 'R CMD javareconf'. On 'Linux', in addition to setting environment variables, 'rJavaEnv' also dynamically loads 'libjvm.so' to ensure 'rJava' works correctly. See documentation for more details.

License MIT + file LICENSE

URL <https://github.com/e-kotov/rJavaEnv>,
<https://www.ekotov.pro/rJavaEnv/>

BugReports <https://github.com/e-kotov/rJavaEnv/issues>

Depends R (>= 4.0)

Imports checkmate, cli, curl, digest, memoise, RcppSimdJson, rlang, utils, withr, yaml

Suggests callr, quarto, rJava, testthat (>= 3.0.0)

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libssl-dev

Repository <https://e-kotov.r-universe.dev>

Date/Publication 2026-10-17 07:40:42 UTC

RemoteUrl <https://github.com/e-kotov/rJavaEnv>

RemoteRef HEAD

RemoteSha 989b5cd7e22e48458af2357167f2119fb6f0f8a0

Contents

java_build_env_set	3
java_build_env_unset	4
java_check_compatibility	5
java_check_version_cmd	6
java_check_version_rjava	7
java_clear	8
java_clear_distrib	9
java_clear_installed	9
java_clear_project	10
java_download	11
java_ensure	12
java_env_set	16
java_env_unset	18
java_find_system	18
java_get_home	19
java_install	20
java_list	21
java_list_available	22
java_list_distrib	23
java_list_installed	24
java_list_project	24
java_quick_install	25
java_resolve	26
java_unpack	28

<i>java_build_env_set</i>	3
java_valid_versions	30
local_java_env	31
rje_consent	33
use_java	34
with_java_env	36
with_rjava_env	37
Index	39

<i>java_build_env_set</i>	<i>Set up the environment for building R packages with Java dependencies from source</i>
---------------------------	--

Description

This function configures the current R session with the necessary environment variables to compile Java-dependent packages like 'rJava' from source. **Note: this function is still experimental.**

Usage

```
java_build_env_set(
  java_home = Sys.getenv("JAVA_HOME"),
  where = c("session", "project", "both"),
  project_path = NULL,
  quiet = FALSE
)
```

Arguments

java_home	The path to the desired JAVA_HOME. Defaults to the value of the JAVA_HOME environment variable.
where	Where to set the build environment: "session", "project", or "both". Defaults to "session". When "both" or "project" is selected, the function updates the .Rprofile file in the project directory.
project_path	The path to the project directory, required when where is "project" or "both". Defaults to the current working directory.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.

Value

Invisibly returns NULL after setting the environment variables.

Examples

```
# Download and install Java 17
java_17_distrib <- java_download(version = "17", temp_dir = TRUE)
java_home_path <- java_install(
  java_distrib_path = java_17_distrib,
  project_path = tempdir(),
  autoset_java_env = FALSE # Manually set env
)

# Set up the build environment in the current session
java_build_env_set(java_home = java_home_path)

# Now, install rJava from source
install.packages("rJava", type = "source", repos = "https://cloud.r-project.org")
```

java_build_env_unset *Unset the Java build environment variables in the project .Rprofile*

Description

Unset the Java build environment variables in the project .Rprofile

Usage

```
java_build_env_unset(project_path = NULL, quiet = FALSE)
```

Arguments

project_path	A character vector of length 1 containing the project directory where Java should be installed. If not specified or NULL, defaults to the current working directory.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.

Value

Does not return a value. Invisibly returns NULL.

Examples

```
# Remove Java build environment settings from the current project
java_build_env_unset()
```

`java_check_compatibility`*Verify rJava Compatibility (Guard)*

Description

Checks if the currently initialized rJava session matches the required Java version. Intended for use in the `.onLoad` function of packages that depend on rJava.

Usage

```
java_check_compatibility(  
  version,  
  type = c("min", "exact"),  
  action = c("warn", "stop", "message", "none")  
)
```

Arguments

<code>version</code>	Integer. The required major Java version (e.g., 21).
<code>type</code>	Character. "min" (default) checks for <code>>= version</code> . "exact" checks for <code>== version</code> .
<code>action</code>	Character. What to do if incompatible: • "warn": Issue a warning but continue. • "stop": Throw an error (prevent package loading). • "message": Print a startup message. • "none": Return FALSE invisibly (useful for custom handling).

Details

This function detects if rJava has already locked to a specific Java version (which happens upon package loading) and warns the user if that version does not match the requirement.

Value

Logical TRUE if compatible, FALSE otherwise.

Examples

```
## Not run:  
# In a package .onLoad:  
.onLoad <- function(libname, pkgname) {  
  rJavaEnv::java_check_compatibility(version = 21, action = "warn")  
}  
  
## End(Not run)
```

`java_check_version_cmd`*Check installed Java version using terminal commands*

Description

Check installed Java version using terminal commands

Usage

```
java_check_version_cmd(java_home = NULL, quiet = FALSE, .use_cache = FALSE)
```

Arguments

<code>java_home</code>	Path to Java home directory. If <code>NULL</code> , the function uses the <code>JAVA_HOME</code> environment variable.
<code>quiet</code>	A logical value indicating whether to suppress messages. Can be <code>TRUE</code> or <code>FALSE</code> .
<code>.use_cache</code>	Logical. If <code>TRUE</code> , uses cached results for repeated calls with the same <code>JAVA_HOME</code> . If <code>FALSE</code> (default), forces a fresh check. Set to <code>TRUE</code> for performance in loops or repeated checks within the same session.

Value

A character vector of length 1 containing the major Java version, or `FALSE` if `JAVA_HOME` is not set or the Java executable cannot be found.

Performance

This function is memoised (cached) within the R session using the effective `JAVA_HOME` as cache key. First call for a given `JAVA_HOME`: ~37ms. Subsequent calls (with `.use_cache = TRUE`): <1ms. When you switch Java versions via `use_java()`, `JAVA_HOME` changes, creating a new cache entry. Cache is session-scoped.

Examples

```
java_check_version_cmd()
```

`java_check_version_rjava`*Check Java Version with a Specified JAVA_HOME Using a Separate R Session*

Description

This function sets the JAVA_HOME environment variable, initializes the JVM using rJava, and prints the Java version that would be used if the user sets the given JAVA_HOME in the current R session. This check is performed in a separate R session to avoid having to reload the current R session. The reason for this is that once Java is initialized in an R session, it cannot be uninitialized unless the current R session is restarted.

Usage

```
java_check_version_rjava(java_home = NULL, quiet = FALSE, .use_cache = FALSE)
```

Arguments

<code>java_home</code>	Path to Java home directory. If NULL, the function uses the JAVA_HOME environment variable.
<code>quiet</code>	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
<code>.use_cache</code>	Logical. If TRUE, uses cached results for repeated calls with the same JAVA_HOME. If FALSE (default), forces a fresh check. Set to TRUE for performance in loops or repeated checks within the same session.

Value

A character vector of length 1 containing the major Java version, or FALSE if rJava is not installed or the version check fails.

Examples

```
java_check_version_rjava()
```

java_clear

Manage Java installations and distributions caches

Description

Wrapper function to clear the Java symlinked in the current project, installed, or distributions caches.

Usage

```
java_clear(  
  type = c("project", "installed", "distrib"),  
  target_dir = NULL,  
  check = TRUE,  
  delete_all = FALSE  
)
```

Arguments

type	What to clear: "project" - remove symlinks to install cache in the current project, "installed" - remove installed Java versions, "distrib" - remove downloaded Java distributions.
target_dir	The directory to clear. Defaults to current working directory for "project" and user-specific data directory for "installed" and "distrib". Not recommended to change.
check	Whether to list the contents of the cache directory before clearing it. Defaults to TRUE.
delete_all	Whether to delete all items without prompting. Defaults to FALSE.

Value

A message indicating whether the cache was cleared or not.

Examples

```
if (interactive()) {  
  java_clear("project")  
  java_clear("installed")  
  java_clear("distrib")  
}
```

java_clear_distrib	<i>Clear the Java distributions cache folder</i>
--------------------	--

Description

Clear the Java distributions cache folder

Usage

```
java_clear_distrib(  
  cache_path = getOption("rJavaEnv.cache_path"),  
  check = TRUE,  
  delete_all = FALSE  
)
```

Arguments

cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.
check	Whether to list the contents of the cache directory before clearing it. Defaults to TRUE.
delete_all	Whether to delete all items without prompting. Defaults to FALSE.

Value

A message indicating whether the cache was cleared or not.

Examples

```
if (interactive()) {  
  java_clear_distrib()  
}
```

java_clear_installed	<i>Clear the Java installations cache folder</i>
----------------------	--

Description

Clear the Java installations cache folder

Usage

```
java_clear_installed(
  check = TRUE,
  delete_all = FALSE,
  cache_path = getOption("rJavaEnv.cache_path")
)
```

Arguments

check	Whether to list the contents of the cache directory before clearing it. Defaults to TRUE.
delete_all	Whether to delete all items without prompting. Defaults to FALSE.
cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.

Value

A message indicating whether the cache was cleared or not.

Examples

```
if (interactive()) {
  java_clear_installed()
}
```

java_clear_project	<i>Clear the Java versions symlinked in the current project</i>
--------------------	---

Description

Clear the Java versions symlinked in the current project

Usage

```
java_clear_project(project_path = NULL, check = TRUE, delete_all = FALSE)
```

Arguments

project_path	The project directory to clear. Defaults to the current working directory.
check	Whether to list the contents of the cache directory before clearing it. Defaults to TRUE.
delete_all	Whether to delete all items without prompting. Defaults to FALSE.

Value

A message indicating whether the symlinks were cleared or not.

Examples

```
if (interactive()) {
  java_clear_project()
}
```

java_download	<i>Download a Java distribution</i>
---------------	-------------------------------------

Description

Download a Java distribution

Usage

```
java_download(
  version = 21,
  distribution = "Corretto",
  backend = getOption("rJavaEnv.backend", "native"),
  cache_path = getOption("rJavaEnv.cache_path"),
  platform = platform_detect()$os,
  arch = platform_detect()$arch,
  quiet = FALSE,
  force = FALSE,
  temp_dir = FALSE
)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of java_list_available(backend = "sdkman").
distribution	The Java distribution to download. One of "Corretto", "Temurin", or "Zulu". Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
backend	Download backend to use. One of "native" (vendor APIs) or "sdkman". Defaults to "native". Can also be set globally via options(rJavaEnv.backend = "sdkman").
cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.

platform	The platform for which to download the Java distribution. Defaults to the current platform.
arch	The architecture for which to download the Java distribution. Defaults to the current architecture.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
force	A logical. Whether the distribution file should be overwritten or not. Defaults to FALSE.
temp_dir	A logical. Whether the file should be saved in a temporary directory. Defaults to FALSE.

Details

SDKMAN catalogues can contain older releases or identifiers that its download broker cannot serve. When resolving a new SDKMAN download, the selected identifier is printed unless quiet = TRUE. Cached SDKMAN archives are reused without querying the catalogue or broker; their resolved identifier is not stored or reported. Use backend = "native" for the current release from a supported vendor, or request a specific version or SDKMAN identifier.

Cached archive names include the backend to prevent native and SDKMAN downloads from sharing an archive. Archives cached by earlier versions are left in place and downloaded again under the new name when needed. Existing installed Java environments are unchanged.

Value

The path to the downloaded Java distribution file.

Examples

```
# download distribution of Java version 17
java_download(version = "17", temp_dir = TRUE)

# download default Java distribution (version 21)
java_download(temp_dir = TRUE)

# download using SDKMAN backend
try(java_download(version = "21", backend = "sdkman", temp_dir = TRUE))
```

Description

Checks for a specific Java version in the following order:

1. Checks if the currently active session has the required version.
2. (Optional) Scans system for installed Java (via `java_find_system`).
3. Checks if the required version is already cached in `rJavaEnv`'s installed cache.
4. If none found, downloads and installs the version (if `install = TRUE`).

This function is designed to be "lazy": it will do nothing if a valid Java version is already detected, making it safe to use in scripts or package startup code (provided `install = FALSE`).

Usage

```
java_ensure(
  version = NULL,
  type = c("exact", "min"),
  accept_system_java = TRUE,
  install = TRUE,
  distribution = "Corretto",
  backend = getOption("rJavaEnv.backend", "native"),
  check_against = c("rJava", "cmd"),
  quiet = FALSE,
  cache_path = getOption("rJavaEnv.cache_path"),
  platform = platform_detect()$os,
  arch = platform_detect()$arch,
  .use_cache = FALSE,
  .check_rjava_fun = check_rjava_initialized,
  .rjava_ver_fun = java_check_current_rjava_version
)
```

Arguments

<code>version</code>	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the <code>distribution</code> and <code>backend</code> arguments are ignored and derived from the identifier. Find available identifiers in the <code>identifier</code> column of <code>java_list_available(backend = "sdkman")</code>.
<code>type</code>	Character. "exact" (default) checks for exact version match. "min" checks for version $\geq$ version.
<code>accept_system_java</code>	Logical. If TRUE (default), the function will scan the system for existing Java installations (using <code>JAVA_HOME</code> , <code>PATH</code> , and OS-specific locations). If a system

	Java matching the version and type requirements is found, it will be used. Set to FALSE to ignore system installations and strictly use an rJavaEnv managed version.
install	Logical. If TRUE (default), attempts to download/install if missing. If FALSE, returns FALSE if the version is not found.
distribution	Character. The Java distribution to download. Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
backend	Download backend to use. One of "native" (vendor APIs) or "sdkman". Defaults to "native". Can also be set globally via <code>options(rJavaEnv.backend = "sdkman")</code> .
check_against	Character. Controls which context validity the function checks against. • "rJava" (default): Checks if the requested version can be enforced for rJava. If rJava is already initialized and locked to a different version, this will error, as the requested version cannot be enforced for the active rJava session. • "cmd": Checks if the requested version can be enforced for command-line use. This ignores the state of rJava and allows setting the environment variables even if rJava is locked to a different version.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.
platform	The platform for which to download the Java distribution. Defaults to the current platform.
arch	The architecture for which to download the Java distribution. Defaults to the current architecture.
.use_cache	A logical value controlling caching behavior. If FALSE (default), performs a fresh check each time (safe, reflects current state). If TRUE, uses session-scoped cached results for performance in loops or repeated calls.

Caching Behavior:

- Session-scoped: Cache is cleared when R restarts
- Key-based for version checks: Changes to JAVA_HOME create new cache entries
- System-wide for scanning: Always recalculates current default Java

Performance Benefits:

- First call: ~37-209ms (depending on operation)
- Cached calls: <1ms
- Prevents 30-100ms delays on every call in performance-critical code

When to Enable:

- Package initialization code (.onLoad or similar)
- Loops calling the same function multiple times
- Performance-critical paths with frequent version checks

When to Keep Default (FALSE):

- Interactive use (one-off checks)
- When you need current data reflecting recent Java installations
- General-purpose function calls that aren't time-critical

`.check_rjava_fun`

Internal. Function to check if rJava is initialized.

`.rjava_ver_fun` Internal. Function to get the current rJava version.

Value

Logical. TRUE if the requirement is met (active or set successfully), FALSE otherwise.

Additional Notes

If rJava is already loaded, this function will warn you but will not prevent the environment variable change (which won't help rJava at that point).

rJava Path-Locking

Important for *rJava* Users: This function sets environment variables (JAVA_HOME, PATH) that affect both command-line Java tools and *rJava* initialization. However, due to *rJava*'s path-locking behavior when `.jinit` is called (see <https://github.com/s-u/rJava/issues/25>, <https://github.com/s-u/rJava/issues/249>, and <https://github.com/s-u/rJava/issues/334>), this function must be called **BEFORE** `.jinit` is invoked. Once `.jinit` initializes, the Java version is locked for that R session and cannot be changed without restarting R.

`.jinit` is invoked (and Java locked) when you:

- Explicitly call `library(rJava)`
- Load any package that imports *rJava* (which auto-loads it as a dependency)
- Even just use IDE autocomplete with `rJava:` (this triggers initialization!)
- Call any *rJava*-dependent function

Once any of these happen, the Java version used by *rJava* for that session is locked in. For command-line Java tools that don't use *rJava*, this function can be called at any time to switch Java versions for subsequent system calls.

See Also

`vignette("for-developers")` for comprehensive guidance on integrating rJavaEnv into your package, including how to use `java_ensure()` in different scenarios and detailed use cases with `type` and `accept_system_java` parameters.

Examples

```
# For end users: Ensure Java 21 is ready BEFORE loading rJava packages
library(rJavaEnv)
java_ensure(version = 21, type = "min")
# Now safe to load packages that depend on rJava
```

```
# library(myRJavaPackage)

# For packages using command-line Java (not rJava):
# Can use java_ensure() within functions to set Java before calling system tools
my_java_tool <- function() {
  java_ensure(version = 17)
  system2("java", c("-jar", "tool.jar"))
}
```

java_env_set

Set the JAVA_HOME and PATH environment variables to a given path

Description

Sets the JAVA_HOME and PATH environment variables for command-line Java tools and rJava initialization. See details for important information about rJava timing.

Usage

```
java_env_set(
  where = c("session", "both", "project"),
  java_home,
  project_path = NULL,
  quiet = FALSE
)
```

Arguments

where	Where to set the JAVA_HOME: "session", "project", or "both". Defaults to "session" and only updates the paths in the current R session. When "both" or "project" is selected, the function updates the .Rprofile file in the project directory to set the JAVA_HOME and PATH environment variables at the start of the R session.
java_home	The path to the desired JAVA_HOME.
project_path	A character vector of length 1 containing the project directory where Java should be installed. If not specified or NULL, defaults to the current working directory.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.

Value

Nothing. Sets the JAVA_HOME and PATH environment variables.

Additional Details

To use a different Java version with *rJava*-dependent packages, you must:

1. Set `JAVA_HOME` using this function **BEFORE** loading *rJava* or any package that imports it
2. Restart your R session if you already loaded *rJava* with the wrong Java version

rJava Path-Locking

Important for *rJava* Users: This function sets environment variables (`JAVA_HOME`, `PATH`) that affect both command-line Java tools and *rJava* initialization. However, due to *rJava*'s path-locking behavior when `.jinit` is called (see <https://github.com/s-u/rJava/issues/25>, <https://github.com/s-u/rJava/issues/249>, and <https://github.com/s-u/rJava/issues/334>), this function must be called **BEFORE** `.jinit` is invoked. Once `.jinit` initializes, the Java version is locked for that R session and cannot be changed without restarting R.

`.jinit` is invoked (and Java locked) when you:

- Explicitly call `library(rJava)`
- Load any package that imports *rJava* (which auto-loads it as a dependency)
- Even just use IDE autocomple with `rJava::` (this triggers initialization!)
- Call any *rJava*-dependent function

Once any of these happen, the Java version used by *rJava* for that session is locked in. For command-line Java tools that don't use *rJava*, this function can be called at any time to switch Java versions for subsequent system calls.

Examples

```
# download, install Java 17
java_17_distrib <- java_download(version = "17", temp_dir = TRUE)
java_home <- java_install(
  java_distrib_path = java_17_distrib,
  project_path = tempdir(),
  autoset_java_env = FALSE
)

# now manually set the JAVA_HOME and PATH environment variables in current session
java_env_set(
  where = "session",
  java_home = java_home
)

# or set JAVA_HOME and PATH in the specific projects' .Rprofile
java_env_set(
  where = "project",
  java_home = java_home,
  project_path = tempdir()
)
```

java_env_unset	<i>Unset the JAVA_HOME and PATH environment variables in the project .Rprofile</i>
----------------	--

Description

Unset the JAVA_HOME and PATH environment variables in the project .Rprofile

Usage

```
java_env_unset(project_path = NULL, quiet = FALSE)
```

Arguments

project_path	A character vector of length 1 containing the project directory where Java should be installed. If not specified or NULL, defaults to the current working directory.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.

Value

Nothing. Removes the JAVA_HOME and PATH environment variables settings from the project .Rprofile.

Examples

```
# clear the JAVA_HOME and PATH environment variables in the specified project .Rprofile
java_env_unset(project_path = tempdir())
```

java_find_system	<i>Discover system-wide Java installations</i>
------------------	--

Description

Scans the system for installed Java Development Kits (JDKs). Implementation based on CMake FindJava.cmake logic, rJava detection, and standard OS checks.

Note: This function only returns true system installations. Java installations managed by rJavaEnv (stored in the package cache) are explicitly excluded, even if JAVA_HOME or PATH currently point to them.

Usage

```
java_find_system(quiet = TRUE, .use_cache = FALSE)
```

Arguments

<code>quiet</code>	Logical. If TRUE, suppresses informational messages.
<code>.use_cache</code>	Logical. If TRUE, uses memoisation cache for the expensive system scanning part. The <code>is_default</code> flag is always calculated dynamically based on the current <code>JAVA_HOME</code> . Default: FALSE (bypass cache for safety). Set to TRUE for performance in loops or repeated calls.

Value

A data frame with columns:

- `java_home`: Character. Path to the Java home directory.
- `version`: Character. Major Java version (e.g., "17", "21").
- `is_default`: Logical. TRUE if this matches the current system default Java (determined by `JAVA_HOME` or `PATH`). Rows are ordered with the default Java first (if detected), then by version descending. Returns an empty data frame if no Java installations are found.

Performance

The system scan is memoised (cached) for the session duration. First scan: ~209ms. Subsequent scans with `.use_cache = TRUE`: <1ms. The `is_default` flag is always calculated fresh to reflect current `JAVA_HOME`.

Examples

```
# List all system Java installations
java_find_system()

# Use cache for faster repeated calls if you are using it in your R package
java_find_system(.use_cache = TRUE)
```

<code>java_get_home</code>	<i>Get JAVA_HOME</i>
----------------------------	----------------------

Description

Get the current `JAVA_HOME` environment variable.

Usage

```
java_get_home()
```

Value

The value of the `JAVA_HOME` environment variable.

Examples

```
java_get_home()
```

```
java_install
```

```
Install Java from a distribution file
```

Description

Unpack Java distribution file into cache directory and link the installation into a project directory, optionally setting the JAVA_HOME and PATH environment variables to the Java version that was just installed.

Usage

```
java_install(  
  java_distrib_path,  
  project_path = NULL,  
  distribution = NULL,  
  backend = NULL,  
  autoset_java_env = TRUE,  
  quiet = FALSE,  
  force = FALSE  
)
```

Arguments

java_distrib_path	A character vector of length 1 containing the path to the Java distribution file.
project_path	A character vector of length 1 containing the project directory where Java should be installed. If not specified or NULL, defaults to the current working directory.
distribution	The Java distribution name (e.g., "Corretto", "Temurin"). If NULL, uses attributes from java_distrib_path or defaults to "unknown".
backend	The download backend used (e.g., "native", "sdkman"). If NULL, uses attributes from java_distrib_path or defaults to "unknown".
autoset_java_env	A logical indicating whether to set the JAVA_HOME and PATH environment variables to the installed Java directory. Defaults to TRUE.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
force	A logical. Whether to overwrite an existing installation. Defaults to FALSE.

Value

The path to the installed Java directory.

Examples

```
## Not run:

# set cache dir to temporary directory
options(rJavaEnv.cache_path = tempdir())
# download, install and autoset environmnet variables for Java 17
java_17_distrib <- java_download(version = "17")
java_install(java_distrib_path = java_17_distrib, project_path = tempdir())

## End(Not run)
```

java_list

List the contents of the Java versions installed or cached

Description

This function lists one of the following:

- **project** - list the contents of the Java symlinked/copied in the current project or directory specified by `target_dir`
- **distrib** - list the contents of the downloaded Java distributions cache in default location or specified by `target_dir`
- **installed** - list the contents of the Java installations cache (unpacked distributions) in default location or specified by `target_dir`

Usage

```
java_list(
  type = c("project", "installed", "distrib"),
  output = c("data.frame", "vector"),
  quiet = TRUE,
  target_dir = NULL
)
```

Arguments

<code>type</code>	The type of cache to list: "distrib", "installed", or "project". Defaults to "project".
<code>output</code>	The format of the output: <code>data.frame</code> or <code>vector</code> . Defaults to <code>data.frame</code> .
<code>quiet</code>	A logical value indicating whether to suppress messages. Can be <code>TRUE</code> or <code>FALSE</code> .
<code>target_dir</code>	The cache directory to list. Defaults to the user-specific data directory for "distrib" and "installed", and the current working directory for "project".

Value

A dataframe or character vector with the contents of the specified cache or project directory.

Examples

```
java_list("project")
java_list("installed")
java_list("distrib")
```

java_list_available	<i>List Available Java Versions</i>
---------------------	-------------------------------------

Description

This function retrieves a list of all available installable Java versions from the specified backend(s). It returns a unified data frame with version details, vendors, and checksum availability.

Usage

```
java_list_available(
  backend = c("both", "native", "sdkman"),
  platform = platform_detect()$os,
  arch = platform_detect()$arch,
  force = FALSE,
  quiet = FALSE
)
```

Arguments

backend	Character vector. Backends to query: "native", "sdkman", or "both" (default).
platform	Platform OS. Defaults to current platform. Use "all" to list for all supported platforms.
arch	Architecture. Defaults to current architecture. Use "all" to list for all supported architectures.
force	Logical. If TRUE, bypasses and refreshes the internal cache.
quiet	Logical. If TRUE, suppresses progress messages.

Value

A data.frame with columns:

- backend: "native" or "sdkman"
- vendor: Java distribution name
- major: Major version number
- version: Full version string
- platform: Platform OS
- arch: Architecture
- identifier: Internal identifier (mainly for SDKMAN)
- checksum_available: Whether checksum verification is available

Examples

```
## Not run:
# List all available versions for current platform
java_list_available()

# List all versions for all platforms (Pro users)
java_list_available(platform = "all", arch = "all")

## End(Not run)
```

java_list_distrib	<i>List the contents of the Java distributions cache folder</i>
-------------------	---

Description

List the contents of the Java distributions cache folder

Usage

```
java_list_distrib(
  cache_path = getOption("rJavaEnv.cache_path"),
  output = c("data.frame", "vector"),
  quiet = TRUE
)
```

Arguments

cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.
output	The format of the output: <code>data.frame</code> or <code>vector</code> . Defaults to <code>data.frame</code> .
quiet	A logical value indicating whether to suppress messages. Can be <code>TRUE</code> or <code>FALSE</code> .

Value

A data frame or character vector with the contents of the cache directory.

Examples

```
java_list_distrib()
```

java_list_installed	<i>List the contents of the Java installations cache folder</i>
---------------------	---

Description

List the contents of the Java installations cache folder

Usage

```
java_list_installed(
  output = c("data.frame", "vector"),
  quiet = TRUE,
  cache_path = getOption("rJavaEnv.cache_path")
)
```

Arguments

output	The format of the output: <code>data.frame</code> or <code>vector</code> . Defaults to <code>data.frame</code> .
quiet	A logical value indicating whether to suppress messages. Can be <code>TRUE</code> or <code>FALSE</code> .
cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.

Value

A data frame or character vector with the contents of the cache directory.

Examples

```
# List the contents
java_list_installed()
```

java_list_project	<i>List the Java versions symlinked in the current project</i>
-------------------	--

Description

List the Java versions symlinked in the current project

Usage

```
java_list_project(
  project_path = NULL,
  output = c("data.frame", "vector"),
  quiet = TRUE
)
```

Arguments

project_path	The project directory to list. Defaults to the current working directory.
output	The format of the output: <code>data.frame`</code> or <code>vector`</code> . Defaults to <code>data.frame</code> .
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.

Value

A data frame or character vector with the symlinked Java versions in the project directory.

Examples

```
java_list_project()
```

java_quick_install	<i>Download and install and set Java in current working/project directory</i>
--------------------	---

Description

Download and install and set Java in current working/project directory

Usage

```
java_quick_install(
  version = 21,
  distribution = "Corretto",
  backend = getOption("rJavaEnv.backend", "native"),
  project_path = NULL,
  platform = platform_detect()$os,
  arch = platform_detect()$arch,
  quiet = FALSE,
  temp_dir = FALSE
)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version.
---------	---

	• SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of <code>java_list_available</code>(backend = "sdkman").
distribution	The Java distribution to download. One of "Corretto", "Temurin", or "Zulu". Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
backend	Download backend to use. One of "native" (vendor APIs) or "sdkman". Defaults to "native". Can also be set globally via <code>options(rJavaEnv.backend = "sdkman")</code> .
project_path	A character vector of length 1 containing the project directory where Java should be installed. If not specified or NULL, defaults to the current working directory.
platform	The platform for which to download the Java distribution. Defaults to the current platform.
arch	The architecture for which to download the Java distribution. Defaults to the current architecture.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
temp_dir	A logical. Whether the file should be saved in a temporary directory. Defaults to FALSE.

Value

Invisibly returns the path to the Java home directory. If quiet is set to FALSE, also prints a message indicating that Java was installed and set in the current working/project directory.

Examples

```
# quick download, unpack, install and set in current working directory default Java version (21)
java_quick_install(17, temp_dir = TRUE)
```

java_resolve

Resolve path to specific Java version

Description

Finds or installs Java and returns the path to JAVA_HOME. This function does not modify any environment variables.

Usage

```
java_resolve(
  version = NULL,
  type = c("exact", "min"),
  distribution = "Corretto",
  backend = getOption("rJavaEnv.backend", "native"),
  install = TRUE,
  accept_system_java = TRUE,
  quiet = FALSE,
  cache_path = getOption("rJavaEnv.cache_path"),
  platform = platform_detect()$os,
  arch = platform_detect()$arch,
  .use_cache = FALSE
)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of java_list_available(backend = "sdkman").
type	Character. "exact" (default) checks for exact version match. "min" checks for version >= version.
distribution	Character. The Java distribution to download. Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
backend	Download backend to use. One of "native" (vendor APIs) or "sdkman". Defaults to "native". Can also be set globally via <code>options(rJavaEnv.backend = "sdkman")</code> .
install	Logical. If TRUE (default), attempts to download/install if missing. If FALSE, returns FALSE if the version is not found.
accept_system_java	Logical. If TRUE (default), the function will scan the system for existing Java installations (using JAVA_HOME, PATH, and OS-specific locations). If a system Java matching the version and type requirements is found, it will be used. Set to FALSE to ignore system installations and strictly use an rJavaEnv managed version.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.

platform	The platform for which to download the Java distribution. Defaults to the current platform.
arch	The architecture for which to download the Java distribution. Defaults to the current architecture.
.use_cache	A logical value controlling caching behavior. If FALSE (default), performs a fresh check each time (safe, reflects current state). If TRUE, uses session-scoped cached results for performance in loops or repeated calls.

Caching Behavior:

- Session-scoped: Cache is cleared when R restarts
- Key-based for version checks: Changes to JAVA_HOME create new cache entries
- System-wide for scanning: Always recalculates current default Java

Performance Benefits:

- First call: ~37-209ms (depending on operation)
- Cached calls: <1ms
- Prevents 30-100ms delays on every call in performance-critical code

When to Enable:

- Package initialization code (.onLoad or similar)
- Loops calling the same function multiple times
- Performance-critical paths with frequent version checks

When to Keep Default (FALSE):

- Interactive use (one-off checks)
- When you need current data reflecting recent Java installations
- General-purpose function calls that aren't time-critical

Value

Character string (Path to JAVA_HOME)

Examples

```
# Get path to Java 21 (installing if necessary)
path <- java_resolve(version = 21, install = TRUE)
```

java_unpack

Unpack a Java distribution file into cache directory

Description

Unpack the Java distribution file into cache directory and return the path to the unpacked Java directory with Java binaries.

Usage

```
java_unpack(
  java_distrib_path,
  distribution = NULL,
  backend = NULL,
  quiet = FALSE,
  force = FALSE
)
```

Arguments

java_distrib_path	A character vector of length 1 containing the path to the Java distribution file.
distribution	The Java distribution name (e.g., "Corretto", "Temurin"). If NULL, attempts to read from attributes on java_distrib_path, otherwise defaults to "unknown".
backend	The download backend used (e.g., "native", "sdkman"). If NULL, attempts to read from attributes on java_distrib_path, otherwise defaults to "unknown".
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
force	A logical. Whether to overwrite an existing installation. Defaults to FALSE.

Value

A character vector containing of length 1 containing the path to the unpacked Java directory.

Examples

```
## Not run:

# set cache dir to temporary directory
options(rJavaEnv.cache_path = tempdir())

# download Java 17 distrib and unpack it into cache dir
java_17_distrib <- java_download(version = "17")
java_home <- java_unpack(java_distrib_path = java_17_distrib)

# set the JAVA_HOME environment variable in the current session
# to the cache dir without touching any files in the current project directory
java_env_set(where = "session", java_home = java_home)

## End(Not run)
```

java_valid_versions	<i>Retrieve Valid Java Versions</i>
---------------------	-------------------------------------

Description

This function retrieves a list of valid Java versions by querying an appropriate API endpoint based on the chosen distribution. The result is cached across sessions via file cache (24 hours) and within a session in memory (8 hours) to avoid repeated API calls. If the API call fails (for example, due to a lack of internet connectivity), the function falls back to a pre-defined list of Java versions.

Usage

```
java_valid_versions(  
  distribution = "Corretto",  
  platform = platform_detect()$os,  
  arch = platform_detect()$arch,  
  force = FALSE  
)
```

Arguments

distribution	The Java distribution to download. One of "Corretto", "Temurin", or "Zulu". Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
platform	The platform for which to download the Java distribution. Defaults to the current platform.
arch	The architecture for which to download the Java distribution. Defaults to the current architecture.
force	Logical. If TRUE, forces a fresh API call even if a cached value exists. Defaults to FALSE.

Value

A character vector of valid Java versions.

Examples

```
# Retrieve valid Java versions (cached if available) using Amazon Corretto endpoint  
versions <- java_valid_versions()  
  
# Force refresh the list of Java versions for Amazon Corretto  
versions <- java_valid_versions(distribution = "Corretto", force = TRUE)
```

local_java_env	<i>Set Java environment for the current scope</i>
----------------	---

Description

Sets JAVA_HOME and PATH to the specified Java version for the remainder of the current function or scope. The environment is automatically restored when the scope exits.

Usage

```
local_java_env(
  version,
  type = c("exact", "min"),
  distribution = "Corretto",
  install = TRUE,
  accept_system_java = TRUE,
  quiet = TRUE,
  .local_envir = parent.frame(),
  .use_cache = FALSE
)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of java_list_available(backend = "sdkman").
type	Character. "exact" (default) checks for exact version match. "min" checks for version >= version.
distribution	Character. The Java distribution to download. Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
install	Logical. If TRUE (default), attempts to download/install if missing. If FALSE, returns FALSE if the version is not found.
accept_system_java	Logical. If TRUE (default), the function will scan the system for existing Java installations (using JAVA_HOME, PATH, and OS-specific locations). If a system Java matching the version and type requirements is found, it will be used. Set to FALSE to ignore system installations and strictly use an rJavaEnv managed version.

<code>quiet</code>	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
<code>.local_envir</code>	The environment to apply the scope to. Defaults to the calling frame.
<code>.use_cache</code>	Logical. If TRUE, uses cached results for Java version checks, improving performance in repeated calls (e.g., inside loops).

Details

This is the recommended way for package developers to use Java executables (via `system2` or `processx`) without permanently altering the user's global environment.

Value

Invisibly returns the path to the selected JAVA_HOME.

Warning - Not for rJava

Do not use this function if your package depends on rJava. `rJava` locks the JVM at initialization and cannot switch versions within a session. If you load an `rJava`-dependent package inside a `local_java_env()` scope, `rJava` will lock to the scoped version, but after the scope exits, `JAVA_HOME` will revert while `rJava` remains locked to the old version. For `rJava` packages, use `java_ensure()` at the start of the R session instead.

Examples

```
## Not run:
# Using system2
my_tool_wrapper <- function() {
  rJavaEnv::local_java_env(version = 21)
  system2("java", "-version")
} # Environment restored automatically here

# Using processx
# run_java_jar <- function(jar_path, args) {
#   rJavaEnv::local_java_env(version = 21)
#   processx::run("java", c("-jar", jar_path, args))
# }

# With caching for repeated calls
# process_files <- function(files) {
#   for (f in files) {
#     rJavaEnv::local_java_env(version = 21, .use_cache = TRUE)
#     processx::run("java", c("-jar", "processor.jar", f))
#   }
# }

## End(Not run)
```

`rje_consent`*Obtain User Consent for rJavaEnv*

Description

Get user consent for `rJavaEnv` to write and update files on the file system. `rJavaEnv` needs permission to manage files in your project and cache directories to function correctly.

Usage

```
rje_consent(provided = FALSE)
```

Arguments

<code>provided</code>	Logical indicating if consent is already provided. To provide consent in non-interactive R sessions use <code>rJavaEnv::rje_consent(provided = TRUE)</code> . Default is <code>FALSE</code> .
-----------------------	---

Details

In line with **CRAN policies**, explicit user consent is required before making these changes. Please call `rJavaEnv::consent()` to provide consent.

Alternatively, you can set the following R option (especially useful for non-interactive R sessions):

```
options(rJavaEnv.consent = TRUE)
```

The function is based on the code of the `renv` package. Copyright 2023 Posit Software, PBC
License: <https://github.com/rstudio/renv/blob/main/LICENSE>

Value

`TRUE` if consent is given, otherwise an error is raised.

Examples

```
# to provide consent and prevent other functions from interrupting to get the consent  
rje_consent(provided = TRUE)
```

use_java	<i>Install specified Java version and set the JAVA_HOME and PATH environment variables in current R session</i>
----------	---

Description

Using specified Java version, set the JAVA_HOME and PATH environment variables in the current R session. If Java distribution has not been downloaded yet, download it. If it was not installed into cache directory yet, install it there and then set the environment variables. This function first checks if the requested version is already present in the rJavaEnv cache. If found, it sets the environment immediately without downloading. If not found, it downloads and unpacks the distribution.

This is intended as a quick and easy way to use different Java versions in R scripts that are in the same project, but require different Java versions. For example, one could use this in scripts that are called by targets package or callr package.

Usage

```
use_java(
  version = NULL,
  distribution = "Corretto",
  backend = getOption("rJavaEnv.backend", "native"),
  cache_path = getOption("rJavaEnv.cache_path"),
  platform = platform_detect()$os,
  arch = platform_detect()$arch,
  quiet = TRUE,
  .skip_rjava_check = FALSE
)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of java_list_available(backend = "sdkman").
distribution	The Java distribution to download. Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
backend	The download backend used (e.g., "native", "sdkman"). If NULL, uses attributes from java_distrib_path or defaults to "unknown".
cache_path	The destination directory to download the Java distribution to. Defaults to a user-specific data directory.

platform	The platform for which to download the Java distribution. Defaults to the current platform.
arch	The architecture for which to download the Java distribution. Defaults to the current architecture.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
._skip_rjava_check	Internal. If TRUE, skip the rJava initialization check.

Value

Logical. Returns TRUE invisibly on success. Prints status messages if quiet is set to FALSE.

rJava Path-Locking

Important for *rJava* Users: This function sets environment variables (JAVA_HOME, PATH) that affect both command-line Java tools and *rJava* initialization. However, due to *rJava*'s path-locking behavior when `.jinit` is called (see <https://github.com/s-u/rJava/issues/25>, <https://github.com/s-u/rJava/issues/249>, and <https://github.com/s-u/rJava/issues/334>), this function must be called **BEFORE** `.jinit` is invoked. Once `.jinit` initializes, the Java version is locked for that R session and cannot be changed without restarting R.

`.jinit` is invoked (and Java locked) when you:

- Explicitly call `library(rJava)`
- Load any package that imports *rJava* (which auto-loads it as a dependency)
- Even just use IDE autocomplete with `rJava::` (this triggers initialization!)
- Call any *rJava*-dependent function

Once any of these happen, the Java version used by *rJava* for that session is locked in. For command-line Java tools that don't use *rJava*, this function can be called at any time to switch Java versions for subsequent system calls.

Examples

```
## Not run:

# set cache directory for Java to be in temporary directory
options(rJavaEnv.cache_path = tempdir())

# For end users: Install and set Java BEFORE loading rJava packages
use_java(21)
# library(myRJavaPackage) # Now uses Java 21

# For command-line Java tools (no rJava involved):
# Can switch versions between system calls
use_java(8)
system2("java", "--version") # Shows Java 8
use_java(17)
system2("java", "--version") # Shows Java 17
```

```
# WARNING: Do NOT do this with rJava packages:
# library(rJava)           # Initializes with system Java
# use_java(21)             # Too late! rJava is already locked
# rJava still uses the Java version from before use_java() call
```

```
## End(Not run)
```

with_java_env

Execute code with a specific Java environment

Description

Temporarily sets JAVA_HOME and PATH for the duration of the provided code block, then restores the previous environment.

Usage

```
with_java_env(version, code, ...)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of java_list_available(backend = "sdkman").
code	The code to execute with the temporary Java environment.
...	Additional arguments passed to <code>local_java_env()</code> .

Value

The result of executing code.

Warning - Not for rJava

Do not use this function if your package depends on rJava. See `local_java_env()` for details on why rJava is incompatible with scoped Java switching.

Examples

```
# Using system2
rJavaEnv::with_java_env(version = 21, {
  system2("java", "-version")
})

# Using processx
# rJavaEnv::with_java_env(version = 21, {
#   processx::run("java", c("-jar", "tool.jar", "--help"))
# })
```

with_rjava_env

Execute rJava code in a separate process with specific Java version

Description

Runs the provided function in a fresh R subprocess where rJava has not yet been loaded. This allows you to enforce a specific Java version for rJava operations without affecting the main R session or requiring a restart.

Usage

```
with_rjava_env(
  version,
  func,
  args = list(),
  distribution = "Corretto",
  install = TRUE,
  accept_system_java = TRUE,
  quiet = TRUE,
  libpath = .libPaths()
)
```

Arguments

version	Java version specification. Accepts: • Major version (e.g., 21, 17): Downloads the latest release for that major version. • Specific version (e.g., "21.0.9", "11.0.29"): Downloads the exact version. • SDKMAN identifier (e.g., "25.0.1-amzn", "24.0.2-open"): Uses the SDKMAN backend automatically. When an identifier is detected, the distribution and backend arguments are ignored and derived from the identifier. Find available identifiers in the identifier column of java_list_available(backend = "sdkman").
func	The function to execute in the subprocess.

args	A list of arguments to pass to func.
distribution	Character. The Java distribution to download. Defaults to "Corretto". Ignored if version is a SDKMAN identifier.
install	Logical. If TRUE (default), attempts to download/install if missing. If FALSE, returns FALSE if the version is not found.
accept_system_java	Logical. If TRUE (default), the function will scan the system for existing Java installations (using JAVA_HOME, PATH, and OS-specific locations). If a system Java matching the version and type requirements is found, it will be used. Set to FALSE to ignore system installations and strictly use an rJavaEnv managed version.
quiet	A logical value indicating whether to suppress messages. Can be TRUE or FALSE.
libpath	Optional character vector of library paths to use in the subprocess. Defaults to .libPaths().

Details

This function requires the **callr** package.

Value

The result of func.

Examples

```
# Run a function using Java 21 in a subprocess
result <- with_rjava_env(
  version = 21,
  func = function(x) {
    library(rJava)
    .jinit()
    .jcall("java.lang.System", "S", "getProperty", "java.version")
  },
  args = list(x = 1)
)
print(result)
```

Index

.jinit, [15](#), [17](#), [35](#)

java_build_env_set, [3](#)
java_build_env_unset, [4](#)
java_check_compatibility, [5](#)
java_check_version_cmd, [6](#)
java_check_version_rjava, [7](#)
java_clear, [8](#)
java_clear_distrib, [9](#)
java_clear_installed, [9](#)
java_clear_project, [10](#)
java_download, [11](#)
java_ensure, [12](#)
java_env_set, [16](#)
java_env_unset, [18](#)
java_find_system, [18](#)
java_get_home, [19](#)
java_install, [20](#)
java_list, [21](#)
java_list_available, [11](#), [13](#), [22](#), [26](#), [27](#), [31](#),
[34](#), [36](#), [37](#)
java_list_distrib, [23](#)
java_list_installed, [24](#)
java_list_project, [24](#)
java_quick_install, [25](#)
java_resolve, [26](#)
java_unpack, [28](#)
java_valid_versions, [30](#)

local_java_env, [31](#)

rje_consent, [33](#)

use_java, [34](#)

with_java_env, [36](#)
with_rjava_env, [37](#)