Package: sandpaper (via r-universe)

September 4, 2024

Title Create and Curate Carpentries Lessons

Version 0.16.7

Description We provide tools to build a Carpentries-themed lesson repository into an accessible standalone static website. These include local tools and those designed to be used in a continuous integration context so that all the lesson author needs to focus on is writing the content of the actual lesson.

License MIT + file LICENSE

Imports pkgdown (>= 1.6.0), pegboard (>= 0.7.0), cli (>= 3.4.0), commonmark, fs (>= 1.5.0), gh, gert (>= 1.0.1), rstudioapi, rlang (>= 0.4.3), glue, assertthat, yaml, desc, knitr (>= 1.33), rmarkdown (>= 2.4), renv (>= 0.14.0), rprojroot, usethis (>= 2.0.0), withr, whisker, callr, servr, utils, tools

Suggests testthat (>= 3.0.0), covr, markdown, brio, xml2, xslt, jsonlite, sessioninfo, mockr, varnish (>= 0.3.0)

Additional_repositories https://carpentries.r-universe.dev/

Remotes carpentries/pegboard, carpentries/varnish

SystemRequirements pandoc (>= 2.11.4) - https://pandoc.org

Encoding UTF-8

LazyData true

Config/testthat/edition 3

Config/testthat/parallel false

Config/Needs/check rstudio/renv

Config/potools/style explicit

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL https://carpentries.github.io/sandpaper/, https://github.com/carpentries/sandpaper/, https://carpentries.github.io/workbench/

BugReports https://github.com/carpentries/sandpaper/issues/

VignetteBuilder knitr

Repository https://carpentries.r-universe.dev

RemoteUrl https://github.com/carpentries/sandpaper

RemoteRef 0.16.7

RemoteSha d566e2dd87540195e1b4e856f82ace8ad975346c

Contents

`build_handout` *Create a code handout of challenges without solutions*

Description

This function will build a handout and save it to `files/code-handout.R` in your lesson website. This will build with your website if you enable it with `options(sandpaper.handout = TRUE)` or if you want to specify a path, you can use `options(sandpaper.handout = "/path/to/handout.R")` to save the handout to a specific path.

Usage

```
build_handout(path = ".", out = NULL)
```

Arguments

path	the path to the lesson. Defaults to current working directory
out	the path to the handout document. When this is NULL (default) or TRUE, the output will be site/built/files/code-handout.R.

---

build_lesson *Build your lesson site*

Description

This function orchestrates rendering generated lesson content and applying the theme for the HTML site.

Usage

```
build_lesson(
  path = ".",
  rebuild = FALSE,
  quiet = !interactive(),
  preview = TRUE,
  override = list()
)
```

Arguments

path	the path to your repository (defaults to your current working directory)
rebuild	if TRUE, everything will be built from scratch as if there was no cache. Defaults to FALSE, which will only build markdown files that haven't been built before.
quiet	when TRUE, output is supressed
preview	if TRUE, the rendered website is opened in a new window
override	options to override (e.g. building to alternative paths). This is used internally and will likely be changed.

Details

Structure of a Workbench Lesson:

A Carpentries Workbench lesson is comprised of a set of markdown files and folders:

```
+-- config.yaml
+-- index.md
+-- episodes
|   +-- data
|   +-- fig
|   +-- files
|   \-- introduction.Rmd
```

```
+-- instructors
|   \-- instructor-notes.md
+-- learners
|   \-- setup.md
+-- profiles
|   \-- learner-profiles.md
+-- links.md
+-- site
    \-- [...]
+-- renv
|   \-- [...]
+-- CODE_OF_CONDUCT.md
+-- CONTRIBUTING.md
+-- LICENSE.md
\-- README.md
```

Value

TRUE if it was successful, a character vector of issues if it was unsuccessful.

See Also

serve(): an interactive way to build and edit lesson content.

Examples

```
tmp <- tempfile()
create_lesson(tmp, open = FALSE, rmd = FALSE)
create_episode("first-script", path = tmp, open = FALSE)
check_lesson(tmp)
build_lesson(tmp)
```

---

create_episode *Create an Episode from a template*

Description

These functions allow you to create an episode that will be added to the schedule.

Usage

```
create_episode(
  title,
  ext = "Rmd",
  make_prefix = FALSE,
  add = TRUE,
```

```
  path = ".",
  open = rlang::is_interactive()
)

create_episode_md(
  title,
  make_prefix = FALSE,
  add = TRUE,
  path = ".",
  open = rlang::is_interactive()
)

create_episode_rmd(
  title,
  make_prefix = FALSE,
  add = TRUE,
  path = ".",
  open = rlang::is_interactive()
)

draft_episode_md(
  title,
  make_prefix = FALSE,
  path = ".",
  open = rlang::is_interactive()
)

draft_episode_rmd(
  title,
  make_prefix = FALSE,
  path = ".",
  open = rlang::is_interactive()
)
```

Arguments

`title`	the title of the episode
`ext`	a character. If `ext = "Rmd"` (default), then the new episode will be an R Markdown episode. If `ext = "md"`, then the new episode will be a markdown episode, which can not generate dynamic content.
`make_prefix`	a logical. When `TRUE`, the prefix for the file will be automatically determined by the files already present. When `FALSE` (default), it assumes no prefix is needed.
`add`	(logical or numeric) If numeric, it represents the position the episode should be added. If `TRUE`, the episode is added to the end of the schedule. If `FALSE`, the episode is added as a draft episode.
`path`	the path to the sandpaper lesson.
`open`	if interactive, the episode will open in a new editor window.

Examples

```
tmp <- tempfile()
create_lesson(tmp, open = FALSE, rmd = FALSE)
create_episode_md("getting-started", path = tmp)
```

`create_lesson` *Create a carpentries lesson*

Description

This will create a boilerplate directory structure for a Carpentries lesson and initialize a git repository.

Usage

```
create_lesson(
  path,
  name = fs::path_file(path),
  rmd = TRUE,
  rstudio = rstudioapi::isAvailable(),
  open = rlang::is_interactive()
)
```

Arguments

`path`	the path to the new lesson folder
`name`	the name of the lesson. If not provided, the folder name will be used.
`rmd`	logical indicator if the lesson should use R Markdown (`TRUE`, default), or if it should use Markdown (`FALSE`). Note that lessons can be converted to use R Markdown at any time by adding a file with the `.Rmd` file extension in the lesson.
`rstudio`	create an RStudio project (defaults to if RStudio exits)
`open`	if interactive, the lesson will open in a new editor window.

Value

the path to the new lesson

Examples

```
tmp <- tempfile()
on.exit(unlink(tmp))
lsn <- create_lesson(tmp, name = "This Lesson", open = FALSE)
lsn
```

get_config *Get the configuration parameters for the lesson*

Description

Get the configuration parameters for the lesson

Usage

```
get_config(path = ".")
```

Arguments

path	path to the lesson

Value

a yaml list

Examples

```
tmp <- tempfile()
create_lesson(tmp, open = FALSE, rmd = FALSE)
get_config(tmp)
```

get_drafts *Show files in draft form*

Description

By default, sandpaper will use the files in alphabetical order as they are presented in the folders, however, it is **strongly** for authors to specify the order of the files in their lessons, so that it's easy to rearrange or add, split, or rearrange files.

Usage

```
get_drafts(
  path,
  folder = NULL,
  message = getOption("sandpaper.show_draft", TRUE)
)
```

Arguments

path	path to the the sandpaper lesson
folder	the specific folder for which to list the draft files. Defaults to NULL, which indicates all folders listed in config.yaml.
message	if TRUE (default), an informative message about the files that are in draft status are printed to the screen.

Details

This mechanism also allows authors to work on files in a draft form without them being published. This function will list and show the files in draft for automation and audit.

Value

a vector of paths to files in draft and a message (if specified)

---

get_dropdown *Helpers to extract contents of dropdown menus on the site*

Description

This fuction will extract the resources that exist and are listed in the config file.

Usage

```
get_dropdown(path = ".", folder, trim = TRUE)

get_episodes(path = ".", trim = TRUE)

get_learners(path = ".", trim = TRUE)

get_instructors(path = ".", trim = TRUE)

get_profiles(path = ".", trim = TRUE)
```

Arguments

path	the path to the lesson, defaults to the current working directory
folder	the folder to extract fromt he dropdown menues
trim	if TRUE (default), only the file name will be presented. When FALSE, the full path will be prepended.

Value

a character vector of episodes in order of presentation

Examples

```
tmp <- tempfile()
create_lesson(tmp, open = FALSE, rmd = FALSE)
get_episodes(tmp)
get_learners(tmp) # information for learners
```

---

known_languages *Show a list of languages known by sandpaper*

Description

Show a list of languages known by sandpaper

Usage

```
known_languages()
```

Details

The known languages are translations of menu and navigational elements that exist in sandpaper. If these elements have not been translated for a given language and you would like to add translations for them, please consult `vignette("translations", package = "sandpaper")` for details of how to do so in the source code for sandpaper.

List of Known Languages::

```
#> - en
#> - es
#> - ja
#> - uk
```

Value

a character vector of language codes known by sandpaper

See Also

`vignette("translations", package = "sandpaper")` for an overview of providing translations.

Examples

```
known_languages()
```

manage_deps *Lesson Runtime Dependency Management*

Description

A customized provisioner for Carpentries Lessons based on **renv** that will install and maintain the requirements for the lesson while *respecting user environments*. This setup leads to several advantages:

- **reliable setup**: the version of the lesson built on the carpentries website will be the same as what you build on your computer because the packages will be identical
- **environmentally friendly**: The lesson dependencies are NOT stored in your default R library and they will not alter your R environment.
- **transparent**: any additions or deletions to the cache will be recorded in the lockfile, which is tracked by git.

The functions that control this cache are the following:

1. `manage_deps()`: Creates and updates the dependencies in your lesson. If no lockfile exists in your lesson, this will create one for you.
2. `update_cache()`: fetches updates for the dependencies and applies them to your cache and lockfile.

This is a wrapper around `renv::record()`, which helps you record a package or set of packages in your lockfile. It can be useful when you want to upgrade or downgrade a specific package.

Usage

```
manage_deps(
  path = ".",
  profile = "lesson-requirements",
  snapshot = TRUE,
  quiet = FALSE
)

update_cache(
  path = ".",
  profile = "lesson-requirements",
  prompt = interactive(),
  quiet = !prompt,
  snapshot = TRUE
)

pin_version(records = NULL, profile = "lesson-requirements", path = ".")
```

Arguments

path	path to your lesson. Defaults to the current working directory.
profile	default to the profile for the lesson. Defaults to `lesson-requirements`. Only use this if you know what you are doing.
snapshot	if `TRUE`, packages from the cache are added to the lockfile (default). Setting this to `FALSE` will add packages to the cache and not snapshot them.
quiet	if `TRUE`, output will be suppressed, defaults to `FALSE`, providing output about different steps in the process of updating the local dependencies.
prompt	if `TRUE`, a message will show you the packages that will be updated in your lockfile and ask for your permission. This is the default if it's running in an interactive session.
records	a character vector or list of packages/resources to include in the lockfile. The most common way to do this is to use the `[package]@[version]` syntax (e.g. `gert@0.1.3`), but there are other specifications where you can specify the remote repository. See `renv::record()` for details.

Details

The **renv** package provides a very useful interface to bring one aspect of reproducibility to R projects. Because people working on Carpentries lessons are also working academics and will likely have projects on their computer where the package versions are necessary for their work, it's important that those environments are respected.

Our flavor of renv applies a package cache explicitly to the content of the lesson, but does not impose itself as the default renv environment.

This provisioner will do the following steps:

1. check for consent to use the package cache via `use_package_cache()` and prompt for it if needed
2. check if the profile has been created and create it if needed via `renv::init()`
3. populate the cache with packages needed from the user's system and download any that are missing via `renv::hydrate()`. This includes all new packages that have been added to the lesson.
4. If there is a lockfile already present, make sure the packages in the cache are aligned with the lockfile (downloading sources if needed) via `renv::restore()`.
5. Record the state of the cache in a lockfile tracked by git. This will include adding new packages and removing old packages. `renv::snapshot()`

When the lockfile changes, you will see it in git and have the power to either commit or restore those changes.

Value

if `snapshot = TRUE`, a nested list representing the lockfile will be returned.

the contents of the lockfile, invisibly

See Also

use_package_cache() and no_package_cache() for turning on and off the package cache, respectively.

`move_episode` *Move an episode in the schedule*

Description

If you need to move a single episode, this function gives you a programmatic or interactive interface to accomplishing this task, whether you need to add and episode, draft, or remove an episode from the schedule.

Usage

```
move_episode(ep = NULL, position = NULL, write = FALSE, path = ".")
```

Arguments

`ep`	the name of a draft episode or the name/number of a published episode to move.
`position`	the position in the schedule to move the episode. Valid positions are from 0 to the number of episodes (+1 for drafts). A value of 0 indicates that the episode should be removed from the schedule.
`write`	defaults to `FALSE`, which will show the potential changes. If `TRUE`, the schedule will be modified and written to `config.yaml`
`path`	the path to the lesson (defaults to the current working directory)

See Also

create_episode(), set_episodes(), get_drafts(), get_episodes()

Examples

```
if (interactive() || Sys.getenv("CI") != "") {
  tmp <- tempfile()
  create_lesson(tmp)
  create_episode_md("getting-started", path = tmp, open = FALSE)
  create_episode_rmd("plotting", path = tmp, open = FALSE)
  create_episode_md("experimental", path = tmp, add = FALSE, open = FALSE)
  set_episodes(tmp, c("getting-started.md", "introduction.Rmd", "plotting.Rmd"),
    write = TRUE)

  # Default episode order is alphabetical, we can use this to nudge episodes
  get_episodes(tmp)
  move_episode("introduction.Rmd", 1L, path = tmp) # by default, it shows you the change
  move_episode("introduction.Rmd", 1L, write = TRUE, path = tmp) # write the results
  get_episodes(tmp)
```

```
  # Add episodes from the drafts
  get_drafts(tmp)
  move_episode("experimental.md", 2L, path = tmp) # view where it will live
  move_episode("experimental.md", 2L, write = TRUE, path = tmp)
  get_episodes(tmp)

  # Unpublish episodes by setting position to zero
  move_episode("experimental.md", 0L, path = tmp) # view the results
  move_episode("experimental.md", 0L, write = TRUE, path = tmp)
  get_episodes(tmp)

  # Interactively select the position where the episode should go by omitting
  # the position argument
  if (interactive()) {
    move_episode("experimental.md", path = tmp)
  }
}
```

`reset_episodes` *Clear the schedule in the lesson*

Description

Clear the schedule in the lesson

Usage

```
reset_episodes(path = ".")
```

Arguments

`path`	path to the lesson

Value

NULL, invisibly

Examples

```
tmp <- tempfile()
create_lesson(tmp, open = FALSE, rmd = FALSE)
get_episodes(tmp) # produces warning
set_episodes(tmp, get_episodes(tmp), write = TRUE)
get_episodes(tmp) # no warning
reset_episodes(tmp)
get_episodes(tmp) # produces warning again because there is no schedule
```

reset_site *Remove all files associated with the site*

Description

Use this if you want to rebuild your site from scratch.

Usage

```
reset_site(path = ".")
```

Arguments

path	the path to the site

Examples

```
tmp <- tempfile()
create_lesson(tmp, open = FALSE, rmd = FALSE)
build_lesson(tmp, preview = FALSE)
dir(file.path(tmp, "site"))
reset_site(tmp)
dir(file.path(tmp, "site"))
```

sandpaper.options *Global Options*

Description

this is some documentation about options

Details

```
option("sandpaper.show_draft" = TRUE)
option("sandpaper.links" = NULL)
option("sandpaper.use_renv" = FALSE)
option("sandpaper.package_cache_trigger" = FALSE)
option("sandpaper.test_fixture" = NULL)
```

As of 2022-02-22, there are several options that are used in sandpaper that may be manipulated by the user. This set may change in the future, but here are the description of these options and how they are set on startup:

sandpaper.show_draft:

Default: `TRUE` This is for user messages. If `TRUE`, a message about episodes in draft status (i.e. episodes that are in the folder, but not in the schedule) will be printed with `get_drafts()`. Setting this option to `FALSE` will turn off this feature.

sandpaper.links:

Default: NULL This option provides a way to override the default place for links in your sandpaper lesson. If it is NULL and there is a file called `links.md` at the top of the repository, this will be appended to the bottom of each page before it is rendered to HTML.

sandpaper.use_renv:

Default: variable This option should not be modified by the user. It determines if renv should be used locally for R-based lessons. It is set by `use_package_cache()` and unset by `no_package_cache()`. If a local user has never consented to using renv previously, then it defaults to FALSE, but if renv has previously been used, it will be TRUE.

sandpaper.package_cache_trigger:

Default: FALSE locally/TRUE on GitHub this tells R Markdown lessons to rebuild everything if the renv lockfile changes.

sandpaper.test_fixture:

Default: NULL This is ONLY for internal use for testing interactive components non-interactively and for setting renv to behave correctly while testing.

`serve` *Build your lesson and work on it at the same time*

Description

This function will serve your lesson and it will auto-update whenever you save a file.

Usage

```
serve(path = ".", quiet = !interactive(), ...)
```

Arguments

`path`	the path to your lesson. Defaults to the current path.
`quiet`	if TRUE, then no messages are printed to the output. Defaults to FALSE in non-interactive sessions, which allows messages to be printed.
`...`	options passed on to `servr::server_config()` by way of `servr::httw()`. These can include **port** and **host** configuration.

Details

`sandpaper::serve()` is an entry point to working on any lesson using The Carpentries Workbench. When you run this function interactively, a preview window will open either in RStudio or your browser with an address like `localhost:4321` (note the number will likely be different). When you make changes to files in your lesson, this preview will update automatically.

When you are done with the preview, you can run `servr::daemon_stop()`.

Command line usage:

You can use this on the command line if you do not use RStudio or another IDE that acts as a web browser. To run this on the command line, use:

```
R -e 'sandpaper::serve()'
```

Note that unlike an interactive session, progress messages are not printed (except for the accessibility checks) and the browser window will not automatically launch. You can have these messages print to screen with the `quiet = FALSE` argument. In addition, If you want to specify a port and host for this function, you can do so using the port and host arguments:

```
R -e 'sandpaper::serve(quiet = FALSE, host = "127.0.0.1", port = "3435")'
```

Value

the output of `servr::httw()`, invisibly. This is mainly used for its side-effect

See Also

`build_lesson()`, render the lesson once, locally.

Examples

```
if (FALSE) {
  # create an example lesson
  tmp <- tempfile()
  create_lesson(tmp, open = FALSE)

  # open the episode for editing
  file.edit(fs::path(tmp, "episodes", "01-introduction.Rmd"))

  # serve the lesson and begin editing the file. Watch how the file will
  # auto-update whenever you save it.
  sandpaper::serve()
  #
  # to stop the server, run
  servr::daemon_stop()
  #
  # If you want to use a different port, you can specify it directly
  sandpaper::serve(host = "127.0.0.1", port = "3435")
}
```

---

set_config *Set individual keys in a configuration file*

Description

Set individual keys in a configuration file

Usage

```
set_config(pairs = NULL, create = FALSE, path = ".", write = FALSE)
```

Arguments

pairs	a named list or character vector with keys as the names and the new values as the contents
create	if TRUE, any new values in pairs will be created and appended; defaults to FALSE, which prevents typos from sneaking in. single key-pair values currently supported.
path	path to the lesson. Defaults to the current directory.
write	if TRUE, the schedule will overwrite the schedule in the current file.

Details

This function deals strictly with keypairs in the yaml. For lists, see `set_dropdown()`.

Default Keypairs Known by Sandpaper:

When you create a new lesson in sandpaper, there are a set of default keypairs that are pre-filled. To make sure contact information and links in the footer are accurate, please modify these values.

- **carpentry** [character] one of cp, dc, swc, lab, incubator
- **title** [character] the lesson title (e.g. 'Introduction to R for Plant Pathologists'
- **created** [character] Date in ISO 8601 format (e.g. '2021-02-09')
- **keywords** [character] comma-separated list (e.g 'static site, R, tidyverse')
- **life_cycle** [character] one of pre-alpha, alpha, beta, stable
- **license** [character] a license for the lesson (e.g. 'CC-BY 4.0')
- **source** [character] the source repository URL
- **branch** [character] the default branch (e.g. 'main')
- **contact** [character] an email address of who to contact for more information about the lesson

Optional Keypairs Known by Sandpaper:

The following keypairs are known by sandpaper, but are optional:

- **lang** [character] the language code that matches the language of the lesson content. This defaults to "en", but can be any language code (e.g. "ja" specifying Japanese) or combination language code and country code (e.g. "pt_BR" specifies Pourtugese used in Brazil). For more information on how this is used, see the Locale Names section of the gettext manual
- **url** [character] custom URL if you are deploying to a URL that is not the default github pages io domain.
- **fail_on_error** [boolean] for R Markdown lessons; fail the build if any chunks produce an error. Use `#| error: true` in chunk options to allow the error to be displayed
- **workbench-beta** [boolean] if truthy, this displays a banner on the site that indicates the site is in the workbench beta phase.
- **overview** [boolean] All lessons must have episodes with the exception of overview lessons. To indicate that your lesson serves as an overview for other lessons, use `overview: true`

- **handout** [boolean] or [character] This option instructs sandpaper to create a handout of all RMarkdown files via pegboard, which uses knitr::purl() in the background after removing everything but the challenges (without solutions) and any code blocks where purl = TRUE. The default path for the handout is files/code-handout.R

As the workbench becomes more developed, some of these optional keys may disappear.

Custom Engines:
To use a specific version of sandpaper or varnish locally, you would install them using remotes::install_github("carp
syntax, but to provision these versions on GitHub, you can provision these in the config.yaml file:

- **sandpaper** [character] github string or version number of sandpaper version to use
- **varnish** [character] github string or version number of varnish version to use
- **pegboard** [character] github string or version number of pegboard version to use

For example, if you had forked your own version of varnish to modify the colourscheme, you could use:

```
varnish: MYACCOUNT/varnish
```

If there is a specific branch of sandpaper or varnish that is being tested, and you want to test it on your lesson temporarily, you could use the @ symbol to refer to the specific branch or commit to use:

```
sandpaper: carpentries/sandpaper@BRANCH-NAME
varnish: carpentries/varnish@BRANCH-name
```

Examples

```
if (FALSE) {
tmp <- tempfile()
create_lesson(tmp, "test lesson", open = FALSE, rmd = FALSE)
# Change the title and License (default vars)
set_config(c(title = "Absolutely Free Lesson", license = "CC0"),
  path = tmp,
  write = TRUE
)

# add the URL and workbench-beta indicator
set_config(list("workbench-beta" = TRUE, url = "https://example.com/"),
  path = tmp,
  create = TRUE,
  write = TRUE
)
}
```

---

set_dropdown *Set the order of items in a dropdown menu*

Description

Set the order of items in a dropdown menu

Usage

```
set_dropdown(path = ".", order = NULL, write = FALSE, folder)

set_episodes(path = ".", order = NULL, write = FALSE)

set_learners(path = ".", order = NULL, write = FALSE)

set_instructors(path = ".", order = NULL, write = FALSE)

set_profiles(path = ".", order = NULL, write = FALSE)
```

Arguments

path	path to the lesson. Defaults to the current directory.
order	the files in the order presented (with extension)
write	if TRUE, the schedule will overwrite the schedule in the current file.
folder	one of four folders that sandpaper recognises where the files listed in order are located: episodes, learners, instructors, profiles.

Examples

```
tmp <- tempfile()
create_lesson(tmp, "test lesson", open = FALSE, rmd = FALSE)
# Change the title and License
set_config(c(title = "Absolutely Free Lesson", license = "CC0"),
  path = tmp,
  write = TRUE
)
create_episode("using-R", path = tmp, open = FALSE)
print(sched <- get_episodes(tmp))

# reverse the schedule
set_episodes(tmp, order = rev(sched))
# write it
set_episodes(tmp, order = rev(sched), write = TRUE)

# see it
get_episodes(tmp)
```

---

strip_prefix *This will strip existing episode prefixes and set the schedule*

Description

Episode order for Carpentries lessons originally used a strategy of prefixing files by a two-digit number to force a specific order by filename. This function will strip these numbers from the filename and set the schedule according to the original order.

Usage

```
strip_prefix(path = ".", write = FALSE)
```

Arguments

path	the path to the lesson (defaults to the current working directory)
write	defaults to FALSE, which will show the potential changes. If TRUE, the schedule will be modified and written to config.yaml

Value

when write = TRUE, the modified list of episodes. When write = FALSE, the modified call is returned.

Note

git will recognise this as deleting a file and then adding a new file in the stage. If you run git add, it should recognise that it is a rename.

See Also

create_episode() for creating new episodes, move_episode() for moving individual episodes around.

Examples

```
if (FALSE) {
  strip_prefix() # test if the function is doing what you want it to do
  strip_prefix(write = TRUE) # rewrite the episode names
}
```

---

update_github_workflows

Update github workflows

Description

This function copies and updates the workflows to run sandpaper.

Usage

```
update_github_workflows(
  path = ".",
  files = "",
  overwrite = TRUE,
  clean = "*.yaml",
  quiet = FALSE
)
```

Arguments

path	path to the current lesson.
files	the files to include in the update. Defaults to an empty string, which will update all files
overwrite	if TRUE (default), the file(s) will be overwritten.
clean	glob of files to be cleaned before writing. Defaults to "*.yaml". to remove all files with the four-letter "yaml" extension (but it will not remove the ".yml" extension). You can also specify a whole file name like "workflow.yaml" to remove one specific file. If you do not want to clean, set this to NULL.
quiet	if TRUE, the process will not output any messages, default is FALSE, which will report on the progress of each step.

Value

the paths to the new files.

---

update_varnish *Update the local version of the carpentries style*

Description

Update the local version of the carpentries style

Usage

```
update_varnish(version = NULL, ...)
```

Arguments

version	if NULL, update the latest version, otherwise, this can be a version string to identify the specific version to use.
...	arguments passed on to utils::install.packages()

Value

NULL, invisibly

Note

this requires an internet connection

use_package_cache *Give Consent to Use Package Cache*

Description

These functions explicitly gives **sandpaper** permission to use **renv** to create a package cache for this and future lessons. There are two states that you can use:

1. `use_package_cache()`: Gives explicit permission to set up and use the package cache with your lesson.
2. `no_package_cache()`: Temporarily suspends permission to use the package cache with your lesson, regardless if it was previously given.

Once you have a package cache defined, you can use changes in the lockfile to trigger rebuilds of the lesson. To do this, you can use:

- `package_cache_trigger(TRUE)`

The above function is best used in conjunction with `update_cache()`

Usage

```
use_package_cache(prompt = interactive(), quiet = !prompt)

no_package_cache()

package_cache_trigger(rebuild = NULL)
```

Arguments

`prompt`	if `TRUE` (default when interactive), a prompt for consent giving information about the proposed modifications will appear on the screen asking for the user to choose to apply the changes or not.
`quiet`	if `TRUE`, messages will not be issued unless `prompt = TRUE`. This defaults to the opposite of `prompt`.
`rebuild`	The new value of the `sandpaper.package_cache_trigger` global option. Setting this to `TRUE` will result in *all materials* being rebuilt when new records enter the package cache lockfile even if no source files have changed. Setting this to `FALSE` will return this to the default state, which is to rebuld only if the source files have changed. The default is `NULL`, which does nothing.

Details

Background:

By default, **sandpaper** will happily build your lesson using the packages available in your default R library, but this can be undesirable for a couple of reasons:

1. You may have a different version of a lesson package that is used on the lesson website, which may result in strange errors, warnings, or incorrect output.
2. You might be very cautious about updating any components of your current R infrastructure because your work depends on you having the correct package versions installed.

To alleviate these concerns, **sandpaper** uses the **renv** package to generate a lesson-specific library that has package versions pinned until the lesson authors choose to update them. This is designed to be minimally-invasive, using the packages you already have and downloading from external repositories only when necessary.

What if I have used renv before?:

If you have used **renv** in the past, then there is no need to give consent to use the cache.

How do I turn off the feature temporarily?:

To turn off the feature you can use `no_package_cache()`. **sandpaper** will respect this option when building your lesson and will use your global library instead.

I have used renv before; how do I turn it off before sandpaper loads?:

You can set `options(sandpaper.use_renv = FALSE)` before loading sandpaper.

Value

nothing. this is used for its side-effect

the value of `getOption("sandpaper.package_cache_trigger")` or `FALSE`, if it is unset.

See Also

`manage_deps()` and `update_cache()` for managing the requirements inside the package cache.

Examples

```
if (!getOption("sandpaper.use_renv") && interactive()) {
  # The first time you set up {renv}, you will need permission
  use_package_cache(prompt = TRUE)
  # The package cache trigger is FALSE, by default
  default <- package_cache_trigger()
  # You can set this to `TRUE` when you update packages with `update_cache()`
  package_cache_trigger(TRUE)
  # set the trigger back to its former state
  package_cache_trigger(default)
}

if (getOption("sandpaper.use_renv") && interactive()) {
  # If you have previously used {renv}, permission is implied
  use_package_cache(prompt = TRUE)

  # You can temporarily turn this off
  no_package_cache()
  getOption("sandpaper.use_renv") # should be FALSE
  use_package_cache(prompt = TRUE)
}
```

Index