Package 'packer'

October 20, 2020

Title An Opinionated Framework for Using 'JavaScript'

Date 2020-10-11

Version 0.0.6

Description
Enforces good practice and provides convenience functions to make work with 'JavaScript' not just easier but also scalable. It is a robust wrapper to 'NPM' and 'webpack' that enables to compartmentalize 'JavaScript' code, leverage 'NPM' packages, and much more.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1.9000

Imports fs, usethis, jsonlite, htmlwidgets, cli, assertthat, rprojroot, rstudioapi

URL https://github.com/JohnCoene/packer, https://packer.john-coene.com

BugReports https://github.com/JohnCoene/packer/issues

Suggests testthat, covr, golem

NeedsCompilation no

Author John Coene [aut, cre] (<https://orcid.org/0000-0002-6637-4107>)

Maintainer John Coene <jcoenep@gmail.com>

Repository CRAN

Date/Publication 2020-10-20 14:30:02 UTC

R topics documented:

`add_plugin_clean` *Clean Plugin*

Description

Add the clean-webpack-plugin to clean the bundled files.

Usage

```
add_plugin_clean(dry = FALSE, verbose = FALSE, clean = TRUE, protect = TRUE)
```

Arguments

`dry`	Whether to simulate the removal of files.
`verbose`	Write Logs to the console.
`clean`	Whether to automatically remove all unused webpack assets on rebuild.
`protect`	Whether to not allow removal of current webpack assets.

add_plugin_html *HTML Plugin*

Description

Add the html-webpack-plugin to the configuration to generate HTML with webpack, used in packer to generate the UI of a golem app with webpack.

Usage

```
add_plugin_html(use_pug = FALSE, output_path = "../index.html")
```

Arguments

use_pug	Set to TRUE to use the pug engine.
output_path	Path to the generated html file, defaults to `../index.html` as is ideal for golem. Note that this path is relative to your output directory specified in your `webpack.common.js` file.

apply_react *Apply React*

Description

Apply React to a project, adds the relevant (babel) loader, installs dependencies, and creates, updates, or replaces the `srcjs/index.js` file.

Usage

```
apply_react(use_cdn = TRUE)
```

Arguments

use_cdn	Whether to use the CDN for `react` and `react-dom` (recommended). This means later importing the dependencies in the shiny UI using `reactCDN()`, this function will be created in a `R/react_cdn.R`. The correct instructions to do so are printed to the console by the function.

Details

After running this function and bundling the JavaScript remember to place the code printed by the function in shiny UI. By default `apply_react()` does not bundle `react` and `react-dom` and thus requires using `reactCDN()` to import the dependencies in the shiny application: this function is created in a `R/react_cdn.R`.

apply_vue *Apply Vue*

Description

Apply Vue to a project, adds the relevant (babel) loader, installs dependencies, and creates, or updates, or replaces the `srcjs/index.js` file.

Usage

```
apply_vue(use_cdn = TRUE)
```

Arguments

use_cdn	Whether to use the CDN for vue (recommended). This means later importing the dependencies in the shiny UI using `vueCDN()`, this function will be created in a `R/vue_cdn.R`. The correct instructions are printed to the console by the application.

Details

After running this function and bundling the JavaScript remember to place div(id = "app"), tags$script(src = "www/index.js") at the bottom of your shiny UI.

bundle *bundle & Watch*

Description

Bundle and watch the JavaScript.

Usage

```
bundle(mode = c("production", "development", "none"))

bundle_prod()

bundle_dev()

watch()
```

Arguments

mode	The configuration mode tells webpack to use its built-in optimisations accordingly.

Functions

- `bundle()` - bundle the project.
- `bundle_prod()` - bundle the project optimising production, equivalent to `bundle("production")` and npm run production.
- `bundle_dev()` - bundle the project including debugging developer tools, equivalent to `bundle("development")` and npm run development.
- `watch()` - watches for changes in the `srcjs` and rebuilds if necessary, equivalent to npm run watch.

---

mockup *Mock up*

Description

Functions to mock up packages for tests

Usage

```
tmp_package()

tmp_golem()

tmp_project()

tmp_ambiorix()

tmp_delete(tmp)
```

Arguments

tmp	A temp mock up project.

---

npm_console *Npm Output*

Description

Prints the output of the last npm command run, useful for debugging.

Usage

```
npm_console()
```

npm_fix *Audit Fix*

Description

Scan your project for vulnerabilities and automatically install any compatible updates to vulnerable dependencies.

Usage

```
npm_fix()
```

Details

Runs npm audit fix

npm_install *Install Npm Packges*

Description

Install npm packges.

Usage

```
npm_install(..., scope = c("dev", "prod", "global"))
```

Arguments

`...`	Packages to install.
`scope`	Scope of installation, see scopes.

Scopes

- `prod` - Installs packages for project with `--save`
- `dev` - Installs dev packages for project with `--save-dev`
- `global` - Instals packages globally with `-g`

scaffold_ambiorix *Ambiorix*

Description

Creates the basic structure for an ambiorix application.

Usage

```
scaffold_ambiorix(vue = FALSE, use_cdn = TRUE, edit = interactive())
```

Arguments

vue	Whether to include Vue, internally runs apply_vue() and adapts the srcjs/index.js template for Vue.
use_cdn	Whether to use the CDN for react or vue dependencies, this is passed to apply_react() or apply_vue() if react or vue arguments are set to TRUE and ignored otherwise.
edit	Automatically open pertinent files.

Details

Only one of react or vue can be set to TRUE.

Value

TRUE (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_ambiorix()

# move to package
setwd(tmp)

# scaffold ambiorix
scaffold_ambiorix()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

scaffold_extension *Shiny Extension*

Description

Creates the basic structure for a shiny extension.

Usage

```
scaffold_extension(name, edit = interactive())
```

Arguments

name	Name of extension used to define file names and functions.
edit	Automatically open pertinent files.

Value

TRUE (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_package()

# move to package
setwd(tmp)

# scaffold ambiorix
scaffold_extension()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

scaffold_golem *Golem*

Description

Creates the basic structure for golem app with JavaScript.

Usage

```
scaffold_golem(
  react = FALSE,
  vue = FALSE,
  use_cdn = TRUE,
  edit = interactive()
)
```

Arguments

react	Whether to include React, internally runs apply_react() and adapts the srcjs/index.js template for React.
vue	Whether to include Vue, internally runs apply_vue() and adapts the srcjs/index.js template for Vue.
use_cdn	Whether to use the CDN for react or vue dependencies, this is passed to apply_react() or apply_vue() if react or vue arguments are set to TRUE and ignored otherwise.
edit	Automatically open pertinent files.

Details

Only one of react or vue can be set to TRUE.

Value

TRUE (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_golem()

# move to package
setwd(tmp)
```

```
# scaffold ambiorix
scaffold_golem()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

`scaffold_input` *Scaffold a Custom Input*

Description

Sets basic structure for a shiny input.

Usage

```
scaffold_input(name, edit = interactive())
```

Arguments

`name`	Name of input, will define internal name binding and CSS class.
`edit`	Automatically open pertinent files.

Value

`TRUE` (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_package()

# move to package
setwd(tmp)

# scaffold ambiorix
scaffold_input()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

scaffold_output *Scaffold Shiny Output*

Description

Sets basic structure for a shiny input.

Usage

```
scaffold_output(name, edit = interactive())
```

Arguments

name	Name of output, will define internal name binding and CSS class.
edit	Automatically open pertinent files.

Value

TRUE (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_package()

# move to package
setwd(tmp)

# scaffold ambiorix
scaffold_output()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

scaffold_rmd *Golem*

Description

Creates the basic structure for golem app with JavaScript.

Usage

```
scaffold_rmd(react = FALSE, vue = FALSE, edit = interactive())
```

Arguments

react	Whether to include React, internally runs apply_react() and adapts the srcjs/index.js template for React.
vue	Whether to include Vue, internally runs apply_vue() and adapts the srcjs/index.js template for Vue.
edit	Automatically open pertinent files.

Details

Only one of react or vue can be set to TRUE.

Value

TRUE (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_project()

# move to package
setwd(tmp)

# scaffold ambiorix
scaffold_rmd()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

scaffold_widget *Scaffold Widget*

Description

Creates basic structure for a widget.

Usage

```
scaffold_widget(name, edit = interactive())
```

Arguments

name	Name of widget, also passed to `htmlwidgets::scaffoldWidget()`.
edit	Automatically open pertinent files.

Details

Internally runs `htmlwidgets::scaffoldWidget()` do not run it prior to this function.

Value

TRUE (invisibly) if successfully run.

Examples

```
if(interactive()){
# current directory
wd <- getwd()

# create a mock up ambiorix project
tmp <- tmp_package()

# move to package
setwd(tmp)

# scaffold ambiorix
scaffold_widget()

# clean up
setwd(wd)
tmp_delete(tmp)
}
```

set_npm *Use npm*

Description

By default packer looks for the npm installation using the `which` command. This function lets you override that behaviour and force a specific npm installation.

Usage

```
set_npm(path = NULL)
```

Arguments

`path`	Path to npm installation to use.

style_loaders *Use Styles*

Description

Installs loaders and adds relevant configuration rules to `srcjs/config/loaders.json`.

Usage

```
use_loader_css(test = "\\.css$")

use_loader_sass(test = "\\.s[ac]ss$/i")
```

Arguments

`test`	Test regular expression test which files should be transformed by the loader.

Details

This will let you import styles much like any other modules, e.g.: import './styles.css'.

Packages

- `use_loader_css()` - installs and imports `style-loader` and `css-loader` packages as dev.
- `use_loader_sass()` - installs and imports `style-loader`, `css-loader`, and `sass-loader` as dev.

tests *Add Tests*

Description

Adds tests to a project.

Usage

```
include_tests(esm = TRUE)

add_test_file(name)

run_tests()
```

Arguments

esm	Whether to install esm and require it for tests (recommended).
name	Name of the test file to add, without extension.

Details

Uses mocha and mocha-webpack and creates a directory called testjs where tests should be placed. The function run_tests() will then uses mocha on all the files in the testjs directory. All tests should end with .test.js. Internally include_tests() also runs use_loader_mocha().

Requiring esm (esm = TRUE) is recommended as it will allow using the latest ESM, e.g.: import in tests.

use_loader_babel *Use babel Loader*

Description

Adds the loader for babel comiler to the loader configuration file.

Usage

```
use_loader_babel(test = "\\.(js|jsx)$", use_eslint = FALSE)
```

Arguments

test	Test regular expression test which files should be transformed by the loader.
use_eslint	Whether to also add the ESlint loader.

Details

The `use_elsint` argument is useful here as loaders have to be defined in the correct order or files might be checked after being processed by babel.

Excludes `node_modules` by default.

`use_loader_coffee` *Use Coffee Loader*

Description

Adds the `coffee-loader` to use cofeescript.

Usage

```
use_loader_coffee(test = "\\.coffee$")
```

Arguments

`test`	Test regular expression test which files should be transformed by the loader.

Details

Excludes `node_modules` by default.

`use_loader_eslint` *Use ESlint*

Description

Adds the `eslint-loader` to resolve files: `png`, `jpg`, `jpeg`, and `gif`.

Usage

```
use_loader_eslint(test = "\\.(js|jsx)$")
```

Arguments

`test`	Test regular expression test which files should be transformed by the loader.

use_loader_file *Use File Loader*

Description

Adds the file-loader to resolve files: png, jpg, jpeg, and gif.

Usage

```
use_loader_file(test = "\\.(png|jpe?g|gif)$/i")
```

Arguments

test	Test regular expression test which files should be transformed by the loader.

use_loader_mocha *Use Mocha Loader*

Description

Adds the mocha-loader for tests.

Usage

```
use_loader_mocha(test = "\\.test\\.js$")
```

Arguments

test	Test regular expression test which files should be transformed by the loader.

Details

Excludes node_modules by default.

use_loader_pug *Use Pug Loader*

Description

Adds the loader for the pug templating engine.

Usage

```
use_loader_pug(test = "\\.pug$")
```

Arguments

test	Test regular expression test which files should be transformed by the loader.

use_loader_rule *Add a Loader Ruée*

Description

Adds a loader rule that is not yet implemened in packer.

Usage

```
use_loader_rule(packages, test, ..., use = as.list(packages))
```

Arguments

packages	NPM packages (loaders) to install.
test	Test regular expression test which files should be transformed by the loader.
...	Any other options to pass to the rule.
use	Name of the loaders to use for test.

Details

Reads the srcsjs/config/loaders.json and appends the rule.

use_loader_vue *Use Vue Loader*

Description

Adds the Vue loader to the loader configuration file.

Usage

```
use_loader_vue(test = "\\.vue$")
```

Arguments

test	Test regular expression test which files should be transformed by the loader.

Details

Every time a new version of Vue is released, a corresponding version of vue-template-compiler is released together. The compiler's version must be in sync with the base Vue package so that vue-loader produces code that is compatible with the runtime. This means every time you upgrade Vue in your project, you should upgrade vue-template-compiler to match it as well.

Index