Package: shinyseo (via r-universe)

June 8, 2026

Type Package

Title Search Engine Optimization, Social Metadata, and Site Verification Helpers for 'Shiny' Apps

Version 1.2.0

Author Rolf Lindgren [aut, cre]

Maintainer Rolf Lindgren <rolf@grendel.no>

URL https://CRAN.R-project.org/package=shinyseo, https://github.com/rolfmblindgren/grendelMeta

BugReports https://github.com/rolfmblindgren/grendelMeta/issues

Description Utilities for injecting search engine optimization (SEO), Open Graph, Twitter, site verification, and schema.org metadata into 'Shiny' applications from YAML files or named lists.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports shiny, yaml, jsonlite

Suggests htmltools, httr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

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

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

Date/Publication 2026-06-08 21:35:51 UTC

RemoteUrl https://github.com/rolfmblindgren/shinyseo

RemoteRef HEAD

RemoteSha 4610b47c10c44cdb534d23636ad07d11d5215927

Contents

`generate_assets` *Generate missing visual assets with a user-supplied LLM or image API*

Description

Looks for missing favicon, Apple touch icon, and share image fields in `meta` and asks a caller-supplied `generator` function to create them. shinyseo does not call any image-generation service itself – `generator` is the caller's own function, written against whatever LLM or image-generation API they already have access to (OpenAI, Adobe Firefly, a local model, and so on). This keeps the package free of any particular vendor's dependencies, API keys, formats, or running costs – the caller's function is the only thing that needs to know which service it is talking to.

Usage

```
generate_assets(
  meta,
  generator,
  path = "www",
  assets = c("favicon", "apple_touch_icon", "image")
)
```

Arguments

`meta`	Either a path to a YAML file or a named list, as for `social_meta()`.
`generator`	A function taking `(prompt, kind)` and returning either a raw vector of image bytes or a path to an image file on disk. `kind` is one of `"favicon"`, `"apple_touch_icon"`, or `"image"`, so the same generator can vary size, aspect ratio, or style depending on what is being made.
`path`	Directory to write generated files into. Defaults to `"www"`, matching `write_manifest()`.
`assets`	Character vector naming which fields to fill in if missing. Defaults to all three: `c("favicon", "apple_touch_icon", "image")`.

Value

The updated `meta` list, with newly generated fields set to paths under `path` (e.g. `"/favicon.png"`). Fields already present in `meta` are left untouched. Write the result back with `yaml::write_yaml()` to persist it.

init_meta *Interactively create a metadata YAML file*

Description

Walks through the fields used by `social_meta()` and `write_manifest()` at the console and writes the answers to a YAML file. Meant as a quick on-ramp so new users can get a working `meta.yml` without first reading the field reference.

Usage

```
init_meta(path = "meta.yml")
```

Arguments

`path`	File to write the metadata to. Defaults to `"meta.yml"`. If the file already exists, its current values are shown as defaults — press Enter on any field to keep what's already there.

Value

The path to the written file, invisibly.

openai_image_generator *Build a generator that calls OpenAI's image API*

Description

Returns a function suitable for the `generator` argument of `generate_assets()`. The returned function calls OpenAI's image generation endpoint and hands back the raw image bytes, so `generate_assets()` can write them straight to disk.

Usage

```
openai_image_generator(
  api_key = Sys.getenv("OPENAI_API_KEY"),
  model = "gpt-image-1"
)
```

Arguments

`api_key`	OpenAI API key. Defaults to the `OPENAI_API_KEY` environment variable.
`model`	OpenAI image model to call. Defaults to `"gpt-image-1"`.

Details

This is the only built-in generator shinyseo ships. It exists because OpenAI's image API is a straightforward fit for the job – one endpoint, one request shape, raster images back. Other providers are welcome as community contributions (see `LLM.md`); shinyseo does not bundle a generator for every LLM vendor, only ones that can actually generate images and pull their weight as a built-in.

Calling this function does not make any network request – it only builds and returns the generator. The 'httr' package is required, but only loaded (via `requireNamespace()`) when you call this constructor, so it costs nothing if you never use OpenAI generation.

Value

A function `function(prompt, kind)` that POSTs to `https://api.openai.com/v1/images/generations` and returns the generated image as a raw vector of bytes.

---

social_meta *Inject social metadata into Shiny UI*

Description

Inject social metadata into Shiny UI

Usage

```
social_meta(meta)
```

Arguments

`meta`	Either a path to a YAML file or a named list. The final metadata must include `title`, `description`, `url`, and `image`.

Details

If `meta` is a character string, it is treated as a YAML file path and read with `yaml::read_yaml()`. Set `schema = FALSE` to suppress JSON-LD output. `bing_site_verification` falls back to `SHINYSEO_BING_SITE_VERIFICATIO` when that environment variable is set. `twitter_site` and `twitter_creator` fall back to `SHINYSEO_TWITTER_SITE` and `SHINYSEO_TWITTER_CREATOR` when those environment variables are set. The helper does not emit a `<title>` tag; set the document title in the app UI so it does not clash with an existing Shiny title. If `favicon` points to an SVG, also set `favicon_png` to a PNG fallback (e.g. 32x32) – Chromium-based browsers' address bar does not render SVG favicons and shows a generic globe icon without one. `favicon_png_sizes` overrides the `sizes` attribute (defaults to `"32x32"`). Set `apple_mobile_web_app_capable = TRUE` to let the app run in standalone mode when added to a phone's home screen (emits `apple-mobile-web-app-capable` and `mobile-web-app-capable`). `apple_mobile_web_app_title` sets the name shown under the home screen icon, and `apple_mobile_web_app_status_bar` controls the iOS status bar appearance. Optional verification fields include `bing_site_verification`, `google_site_verification`, `yandex_site_verification`, `baidu_site_verification`, `naver_site_verification`, `facebook_domain_verification`, and `pinterest_domain_verification`.

Value

A `shiny::tags$head()` fragment containing canonical, Open Graph, Twitter Card, and optional schema.org metadata.

`update_meta` *Reactively update social metadata from the server*

Description

Call this inside a Shiny `server` function to update one or more metadata fields after the page has loaded — for example when the user navigates between tabs or routes. Only the fields you supply are changed; all others stay at whatever value `social_meta()` set at startup.

Usage

```
update_meta(
  session,
  title = NULL,
  description = NULL,
  url = NULL,
  image = NULL
)
```

Arguments

`session`	The Shiny `session` object passed to `server`.
`title`	New page title. Also updates `og:title` and `twitter:title`.
`description`	New description. Also updates `og:description` and `twitter:description`.
`url`	New canonical URL. Also updates `og:url` and `twitter:url`.
`image`	New share image URL. Also updates `og:image` and `twitter:image`.

Value

Called for its side-effect; returns `invisible(NULL)`.

`write_manifest` *Write a web app manifest to the Shiny www directory*

Description

Generates a Web App Manifest (`manifest.json`) and writes it to `www/` so Shiny can serve it at `/manifest.json`. Call this once in `global.R` before the app starts.

Usage

```
write_manifest(
  meta,
  path = "www",
  display = "standalone",
  start_url = "/",
  background_color = NULL
)
```

Arguments

`meta`	Either a path to a YAML file or a named list. The same object you pass to `social_meta()` works here.
`path`	Directory to write `manifest.json` into. Defaults to `"www"` relative to the working directory.
`display`	Browser display mode. One of `"standalone"`, `"fullscreen"`, `"minimal-ui"`, or `"browser"`. Defaults to `"standalone"`.
`start_url`	Start URL passed to the manifest. Defaults to `"/"`.
`background_color`	Background colour shown on the splash screen while the app loads. Defaults to `theme_color` when set, otherwise `"#ffffff"`.

Details

Reference `manifest.json` from the UI by passing `manifest = "/manifest.json"` to `social_meta()`.

Value

The path to the written file, invisibly.

Index