sphinx-lightbox Documentation

Release 0.5.0

Aputsiak Niels Janussen

Jul 08, 2026

USER GUIDE

Version: 0.5.0

Accessible click-to-enlarge images for Sphinx.

`sphinx-lightbox` is a Sphinx extension that provides click-to-enlarge image viewing in HTML output using a CSS-driven checkbox-toggle mechanism, progressively enhanced with lightweight JavaScript for keyboard activation, focus management, and gallery navigation.

Note

This documentation is *self-referential*: the live examples below are rendered by the extension you are reading about.

CHAPTER

ONE

PROJECT HOMES

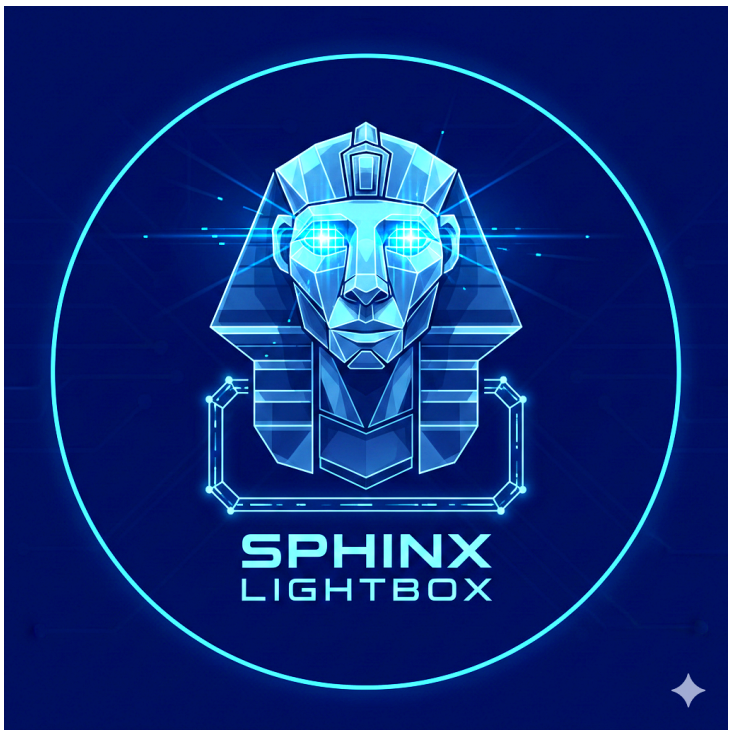
- GitHub repository
- PyPI package

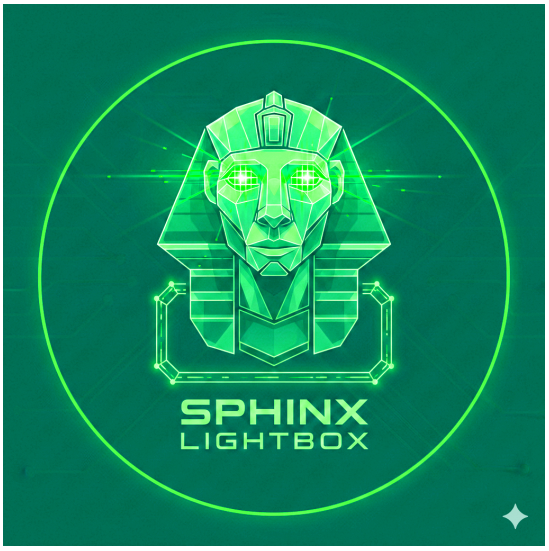
CHAPTER

TWO

IMAGE EXAMPLE

In the HTML documentation, users are told to click these standard Sphinx `image` examples to open them in a lightbox. Because this page contains multiple lightbox images, web users can also move between them with gallery controls or the arrow keys. They can close the lightbox with the close icon, by clicking outside the image, or with `Esc`, `Enter`, or `Space`.




SPHINX
LIGHTBOX

CHAPTER

THREE

FIGURE EXAMPLE

In the HTML documentation, users are told to click this standard Sphinx `figure` example to open it in a lightbox. The web overlay shows the image together with the figure caption and legend text. They can close the lightbox with the close icon, by clicking outside the image, or with `Esc`, `Enter`, or `Space`.

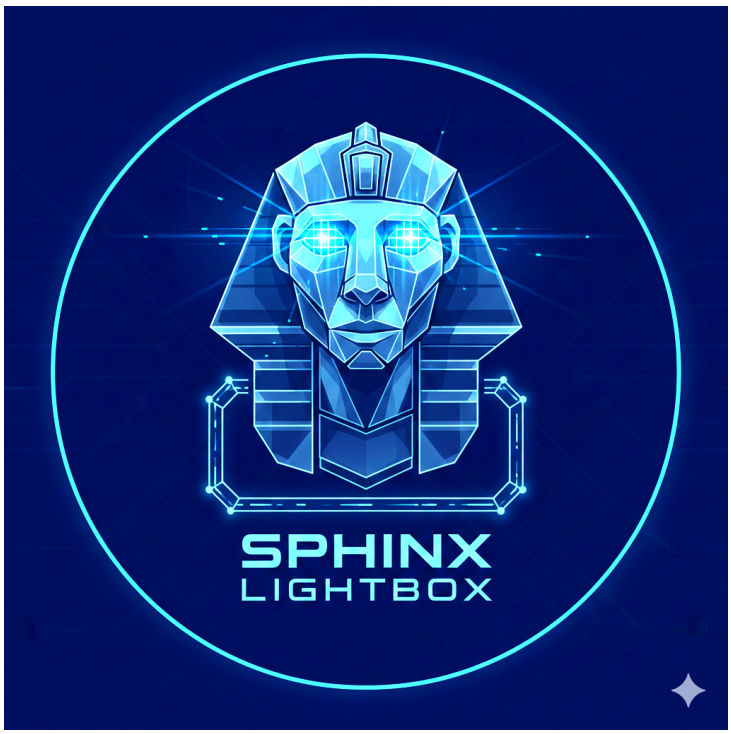


Fig. 1: Figure caption shown on the page and in the lightbox overlay.
This legend is longer explanatory text attached to the figure. It remains on the page and is also shown in the lightbox overlay.

CHAPTER

FOUR

KEY FEATURES

- **CSS-Driven Toggle** — uses a highly robust checkbox-toggle pattern at its core, progressively enhanced with lightweight JavaScript for keyboard accessibility.
- **Sphinx-native image handling** — images are registered with Sphinx's collector, land in `_images/`, and participate in incremental builds.
- **Multi-builder support** — full lightbox in HTML, `\includegraphics` with caption in LaTeX/PDF, plain image fallback in other builders.
- **Accessibility-conscious design** — visible focus indicators, `role="dialog"`, `aria-modal`, accessible control text, `prefers-reduced-motion`, focus management, and `prefers-contrast: more` support.
- **Proper node architecture** — custom docutils nodes with per-builder visitor functions, following Sphinx extension best practices.
- **GPL-3.0-or-later** licensed open source.

CHAPTER

FIVE

CONTENTS

5.1 Installation

5.1.1 Requirements

- Python 3.10 or later
- Sphinx 7.0 through 9.x

5.1.2 Install From PyPI

```
pip install sphinx-lightbox
```

Install the package into the same Python environment that runs Sphinx.

5.1.3 Install From A Checkout

Install the package into the same Python environment that runs Sphinx:

```
pip install -e .
```

This keeps the package importable as `lightbox` while the project is being developed locally.

5.1.4 Enable the extension

Add `lightbox` to your Sphinx `conf.py`:

```
# If the lightbox package is on sys.path:
extensions = [
    "lightbox",
]
```

The extension automatically registers its CSS and JavaScript files. No additional template changes are needed.

5.1.5 Image directory setup

The extension works with Sphinx's standard image pipeline. Place your content images under your source tree, for example:

```
docs/
├── images/
│   ├── topic-a/
│   │   ├── screenshot-1.png
│   │   └── screenshot-2.png
│   └── topic-b/
│       └── detail.png
└── index.rst
```

Reference them with standard Sphinx markup. Absolute paths are resolved from the source root:

```
.. image:: /images/topic-a/screenshot-1.png
   :alt: Topic A screenshot.
   :class: lightbox
```

Or with document-relative paths:

```
.. image:: ../images/topic-a/screenshot-1.png
   :alt: Topic A screenshot.
   :class: lightbox
```

Images are resolved through Sphinx's collector and placed in `_images/` in the HTML build output — not `_static/`.

5.2 Usage

5.2.1 Standard Images And Figures

The recommended API is ordinary Sphinx `image` and `figure` markup. The extension transforms eligible images only for HTML output.

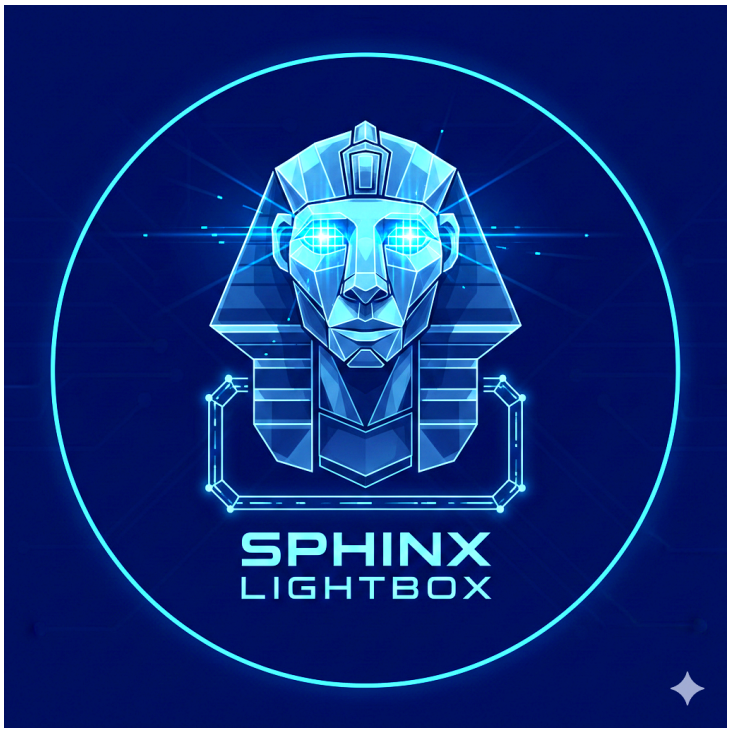
Configure the transform policy in `conf.py`:

```
lightbox_images = "explicit"   # "explicit", "all", or "none"
lightbox_figures = "all"       # "explicit", "all", or "none"
lightbox_default_class = "with-shadow"
```

With `lightbox_images = "explicit"`, opt a normal image into lightbox handling with `:class: lightbox`:

```
.. image:: /images/example-screenshot.png
   :alt: Standard image with lightbox behavior.
   :class: lightbox
```

Plain images do not have built-in Sphinx captions, so the overlay shows only the image:



With `lightbox_figures = "all"`, figures are transformed by default. The normal figure caption and legend remain in the page and are copied into the lightbox overlay:

```
.. figure:: /images/example-screenshot.png
   :alt: Figure with lightbox behavior.

   This caption appears in the page and in the lightbox overlay.

   This legend is longer explanatory text attached to the figure.
```

Add `:class: no-lightbox` to an individual image or figure to opt out when its policy is `"all"`.

5.2.2 Gallery Mode

Gallery navigation is optional. It adds previous and next controls between lightboxes in the same document without changing the authoring markup:

```
lightbox_gallery = "document"  # "document" or "none"
lightbox_gallery_wrap = False
```

When gallery mode is `"document"`, transformed images and figures are ordered by source order. If a document has more than one lightbox, overlays show previous and next controls. ArrowLeft and ArrowRight navigate the same gallery. A single lightbox does not render gallery controls.

Use `lightbox_gallery = "none"` to keep each lightbox independent.

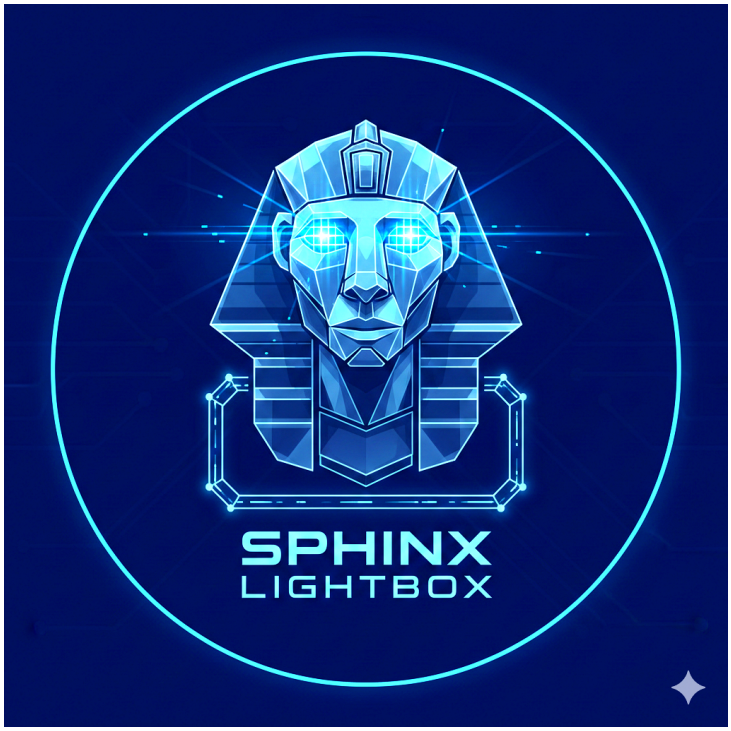


Fig. 1: This caption appears in the page and in the lightbox overlay.
This legend is longer explanatory text attached to the figure.

5.2.3 Captions And Legends

Plain `image` directives do not have Sphinx-native captions, so their overlays show only the image. Use `figure` when an image needs caption or legend text:

```
.. figure:: /images/example-screenshot.png
   :alt: Detail screenshot.
   :width: 55%

   The caption is copied into the lightbox overlay.

   The legend is copied too.
```

5.2.4 Sizing And Styling

Use standard Sphinx image options for thumbnails:

```
.. image:: /images/example-screenshot.png
   :alt: Smaller thumbnail.
   :width: 40%
   :align: center
   :class: lightbox with-border
```

Classes other than `lightbox` and `no-lightbox` are preserved on the thumbnail and overlay image. `lightbox_default_class` adds a default CSS class to transformed images; set it to an empty string to disable the default styling.

5.2.5 Lightbox Directive

The `.. lightbox::` directive creates a lightbox directly and provides directive-specific sizing options.

The directive argument is the image path:

```
.. lightbox:: /images/example-screenshot.png
   :alt: Lightbox directive example.
   :caption: Caption text shown in the overlay.
   :class: with-border
   :percentage: 60 90
```

The `:percentage:` option accepts one or two integers:

- **First value** — thumbnail width as a percentage of the container.
- **Second value** — lightbox display size in HTML and, unless `:latex-width:` is set, the `\linewidth` fraction in LaTeX.

To control PDF sizing independently, add `:latex-width:`:

```
.. lightbox:: /images/example-screenshot.png
   :alt: Lightbox directive with independent PDF sizing.
   :caption: 40% thumbnail, 95% HTML overlay, 60% PDF width.
   :percentage: 40 95
   :latex-width: 0.60
```

Rendered example:

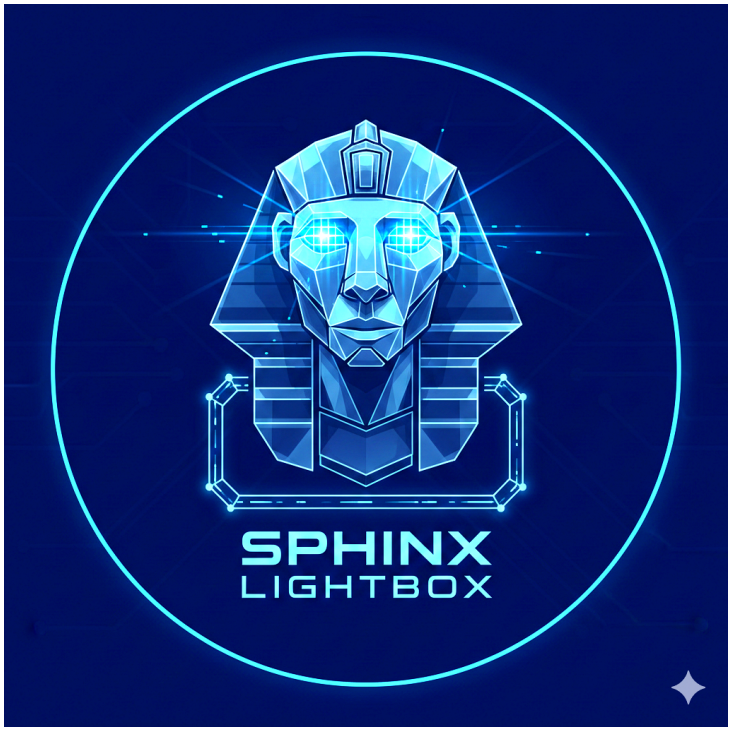


Fig. 2: 40% thumbnail, 95% HTML overlay, 60% PDF width.

Directive lightboxes participate in gallery mode alongside transformed standard images and figures.

5.2.6 Image Paths

The extension supports two path styles:

Absolute paths (from the source root):

```
.. image:: /images/topic/screenshot.png
   :alt: Topic screenshot.
   :class: lightbox
```

Document-relative paths:

```
.. image:: ../images/topic/screenshot.png
   :alt: Topic screenshot.
   :class: lightbox
```

Both styles are resolved through Sphinx's standard image pipeline. If the image file does not exist, a clear warning is emitted at build time with the resolved absolute path.

5.2.7 LaTeX / PDF Output

In LaTeX builds, each lightbox renders as a `figure` environment with `\includegraphics` and `\caption`. By default, the second `:percentage:` value controls the width as a fraction of `\linewidth` (e.g. `95` becomes `0.95\linewidth`). The thumbnail is skipped — only the full-size image appears.

To control PDF sizing independently of HTML, use the optional `:latex-width:` option:

```
.. lightbox:: /images/diagram.png
   :alt: Architecture diagram.
   :percentage: 60 90
   :latex-width: 0.8
```

This sets the HTML overlay to 90% of the viewport while the PDF figure uses 80% of `\linewidth`. When `:latex-width:` is omitted, the second `:percentage:` value is used as before.

5.2.8 Other Builders

For the class-based `image` and `figure` transform, non-HTML builders keep the original Sphinx nodes. For the `.. lightbox::` directive, builders that are neither HTML nor LaTeX (epub, man, texinfo, text) receive a plain `image` node with the alt text, ensuring content is never silently dropped.

5.2.9 JavaScript Disabled

The CSS checkbox-toggle mechanism still supports pointer-based opening and closing when JavaScript is disabled. Keyboard activation, Escape-to-close, focus movement, focus trapping, and arrow-key gallery navigation require the external `lightbox.js` enhancement.

5.2.10 Content Security Policy (CSP)

The lightbox extension uses an external JavaScript file for keyboard activation, Escape-to-close, focus management, and gallery navigation. It does not inject inline JavaScript.

The generated HTML does use inline styles for thumbnail width and overlay sizing. If your documentation is hosted with a strict Content Security Policy, ensure your policy permits inline styles:

```
Content-Security-Policy: style-src 'self' 'unsafe-inline';
```

5.3 Accessibility

The lightbox extension is built with an **accessibility-conscious design**. The core open/close behavior is HTML/CSS-first, and a small JavaScript file adds the keyboard and focus behavior that CSS alone cannot provide.

5.3.1 CSS-Only Toggle Mechanism

The lightbox uses a hidden `<input type="checkbox">` and CSS `:checked` selector to toggle the overlay. This approach works without JavaScript, ensuring the lightbox functions even when scripts are disabled.

5.3.2 ARIA Attributes

The generated HTML includes:

- `role="dialog"` and `aria-modal="true"` on the overlay container, so screen readers announce it as a modal dialog.
- Visually hidden text on the trigger and close controls, describing the action that will occur.
- `aria-hidden="true"` on the hidden checkbox, so screen readers skip the implementation detail.

5.3.3 Keyboard Navigation

All interactive elements have `tabindex="0"` and are reachable via the Tab key:

1. **Tab** to the thumbnail image — it receives a visible focus outline.
2. **Enter** or **Space** to open the lightbox (activates the label).
3. Focus automatically moves to the close button inside the overlay.
4. **Tab** cycles within the overlay (focus is trapped inside the dialog).
5. **ArrowLeft** and **ArrowRight** move between gallery items when gallery controls are present.
6. **Enter**, **Space**, or **Esc** to close.
7. Focus returns to the thumbnail that opened the lightbox.

Note

Progressive enhancement: At its core, the lightbox uses a pure-CSS toggle mechanism that functions even if JavaScript is disabled in the browser. The JavaScript enhancement adds:

- Native `Enter` and `Space` key activation for focused controls.
- `Escape` key to close the overlay.
- **Focus management** — focus moves to the close button when the lightbox opens and returns to the triggering thumbnail when it closes.
- **Focus trap** — `Tab` and `Shift+Tab` cycle only among focusable elements within the open dialog, preventing keyboard focus from escaping to the page behind the overlay.
- Gallery navigation with click, tap, `ArrowLeft`, and `ArrowRight` when a document contains multiple lightboxes.

5.3.4 Focus Indicators

Visible focus outlines appear on both the trigger and close controls:

```
.lightbox-trigger-label:focus-visible {
    outline: 3px solid #4A90D9;
    outline-offset: 3px;
}
```

The `:focus:not(:focus-visible)` pattern ensures outlines appear only for keyboard navigation, not mouse clicks.

5.3.5 Reduced Motion

The hover transition on the thumbnail is disabled when the user has requested reduced motion:

```
@media (prefers-reduced-motion: reduce) {
    .lightbox-trigger-label img.lightbox-trigger {
        transition: none;
    }
}
```

5.3.6 High Contrast Mode

When `prefers-contrast: more` is active, the CSS applies:

- Solid black/white outlines instead of the default blue.
- A white border around the full-size image for clear delineation.
- Increased backdrop opacity (95% instead of 85%).
- A solid black background with white border on the close button.
- Solid black caption and legend panel for readable overlay text.

Live example (the image below respects your system's contrast and motion preferences):

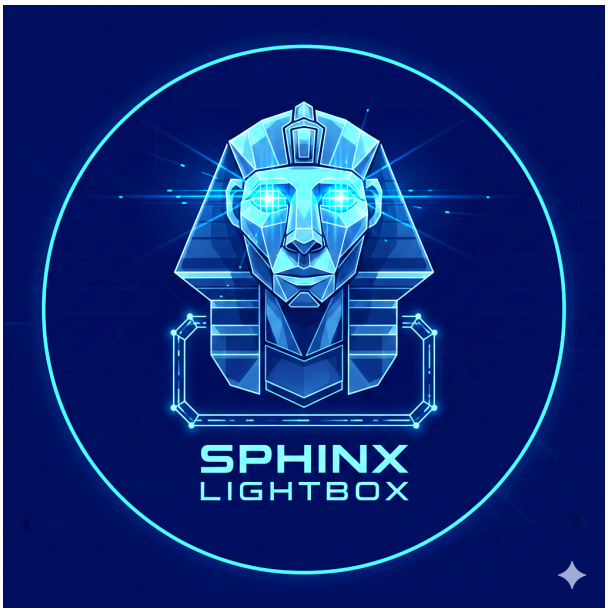


Fig. 3: This image adapts to your accessibility preferences.

5.3.7 Testing Recommendations

When using the lightbox extension, consider testing with:

1. **Keyboard-only navigation** — Tab to the image, Enter to open. Verify that focus moves to the close button. Press Tab and Shift+Tab to confirm focus stays trapped within the overlay. Press Escape, Enter, or Space to close and verify focus returns to the thumbnail.
2. **Screen readers** — verify the dialog is announced (NVDA, VoiceOver, Orca).
3. **Browser zoom at 200%** — ensure the overlay scales properly.

4. **Disabled JavaScript** — the lightbox opens and closes via the CSS checkbox toggle when activated with a pointer. Keyboard activation, Escape handling, focus management, focus trapping, and gallery keyboard navigation are unavailable without scripts.
5. **High contrast mode** — check visibility in Windows High Contrast and `prefers-contrast: more`.
6. **Reduced motion** — verify no transitions occur.

5.4 Directive Reference

The primary API is standard Sphinx `image` and `figure` markup. The `.. lightbox::` directive is also available when directive-specific sizing is useful.

Common project setup:

```
lightbox_images = "explicit"
lightbox_figures = "all"
lightbox_default_class = "with-shadow"
lightbox_gallery = "document"
lightbox_gallery_wrap = False
```

5.4.1 Standard `image` And `figure` Directives

Add `lightbox` to the directive class list to make an ordinary Sphinx image clickable in HTML output:

```
.. image:: /images/photo.png
   :alt: Photo.
   :class: lightbox
```

Figures can also be configured as explicit, but the default is to wrap all figures:

```
.. figure:: /images/photo.png
   :alt: Photo.
   :class: with-border

   Figure caption.

   Longer explanatory legend text.
```

For `figure` nodes, the normal page caption remains in place and the caption and legend text are copied into the lightbox overlay. Plain `image` nodes do not have built-in Sphinx captions, so their overlays show only the image. Other classes, such as `with-border`, are preserved on the thumbnail and overlay image. The control classes `lightbox` and `no-lightbox` are not copied to the rendered image elements.

Use these settings in `conf.py` to decide what is transformed:

```
lightbox_images = "explicit"   # "explicit", "all", or "none"
lightbox_figures = "all"       # "explicit", "all", or "none"
```

When a policy is `"all"`, opt out of an individual image or figure with `no-lightbox`:

```
.. image:: /images/icon.png
   :alt: Small icon.
   :class: no-lightbox
```

The class-based transform runs only for HTML builders. LaTeX/PDF and other builders keep the original Sphinx image or figure nodes.

`lightbox_default_class` sets classes applied to transformed HTML images in addition to any classes on the source directive. The default is `"with-shadow"`. Set it to an empty string to disable default styling:

```
lightbox_default_class = ""
```

5.4.2 Gallery Options

Gallery mode links transformed lightboxes in source order within a document:

```
lightbox_gallery = "document"  # "document" or "none"
lightbox_gallery_wrap = False
```

When enabled and a document has more than one lightbox, overlays render previous and next controls. Click, tap, ArrowLeft, and ArrowRight navigate between items. When `lightbox_gallery_wrap` is false, the first item has no previous control and the last item has no next control.

Set `lightbox_gallery = "none"` to suppress gallery metadata and controls.

5.4.3 `.. lightbox::`

The dedicated directive creates a click-to-enlarge image using a CSS-driven lightbox overlay with optional directive-specific sizing and LaTeX options.

Argument:

`image_path` ***(required)***
: Path to the image file. Supports absolute paths from the source root (starting with `/`) and document-relative paths.

Options:

`:alt:` ***(string)***
: Alt text for the image. Used for both the thumbnail and the overlay image. Recommended for accessibility.

`:caption:` ***(string)***
: Caption text displayed below the full-size image in the overlay. Also used as the `\caption` in LaTeX output.

`:percentage:` ***(one or two integers)***
: Controls image sizing:

  - **First value** — thumbnail width as a percentage of the container width. Default: `100`.
  - **Second value** — overlay display size as a percentage of the viewport. Used for CSS (`vw`/`vh`) and, unless `:latex-width:` is set, also for the LaTeX `\linewidth` fraction. Default: `95`.

`:latex-width:` ***(float, optional)***
: LaTeX image width as a fraction of `\linewidth`, between `0` and `1` (exclusive/inclusive). When

set, this overrides the second `:percentage:` value for LaTeX/PDF output only — HTML sizing is unaffected.

This option is entirely optional. When omitted, the extension derives the LaTeX width from the second `:percentage:` value (default `0.95`).

Example: `:latex-width: 0.8` produces `\adjustbox{max width=0.80\linewidth}`.

`:class:` *(string)*
Additional CSS class(es) applied to the image elements. The built-in `with-border` class adds a subtle border and rounded corners.

HTML output:

```
<div class="lightbox-container">
  <label for="lightbox-usage-1" class="lightbox-trigger-label"
         tabindex="0">
    <span class="lightbox-visually-hidden">Enlarge image: Alt text</span>
    <img src="_images/photo.png" alt="Alt text"
         class="lightbox-trigger with-border"
         style="width: 60%;">
  </label>
  <input type="checkbox" id="lightbox-usage-1"
         class="lightbox-toggle" aria-hidden="true">
  <div class="lightbox-overlay" role="dialog"
    aria-modal="true" aria-label="Alt text">
    <label for="lightbox-usage-1" class="lightbox-close"
           tabindex="0">
      <span aria-hidden="true">&times;</span>
      <span class="lightbox-visually-hidden">Close lightbox</span>
    </label>
    <div class="lightbox-content">
         <img src="_images/photo.png" alt="Alt text"
              class="with-border"
              style="width: min(90vw, calc(90vh * 1.3333)); height: min(90vh,␣
↪calc(90vw / 1.3333));">
      <div class="lightbox-text">
        <p class="lightbox-caption">Caption text here.</p>
      </div>
    </div>
    <label for="lightbox-usage-1" class="lightbox-backdrop-close"></label>
  </div>
</div>
```

The overlay image dimensions are calculated at build time from the actual image file. The `width` and `height` use CSS `min()` with the pre-calculated aspect ratio, so the image fits the viewport without distortion. If Sphinx cannot calculate the dimensions of a highly compressed or corrupted image, a warning is emitted during the build and the overlay gracefully falls back to a 1:1 aspect ratio to prevent crashes.

External URLs:

```
.. lightbox:: https://example.com/remote-image.png
```

Note: Because the lightbox requires build-time image dimension calculations to properly size the CSS overlay, external URLs and ``data:`` URIs bypass the lightbox and are gracefully rendered as standard Sphinx ``image`` nodes.

LaTeX output:

```
\begin{figure}[htbp]
\centering
\adjustbox{max width=0.90\linewidth}{\includegraphics{photo.png}}
\caption{Caption text here.}
\end{figure}
```

The `\adjustbox{max width=...}` wrapper prevents small images from being upscaled beyond their natural size while still constraining large images to the specified fraction of `\linewidth`.

Other builders:

A plain `image` node with alt text - no content is dropped.

5.4.4 Directive Examples

Minimal:

```
.. lightbox:: /images/photo.png
```

Fully specified:

```
.. lightbox:: /images/archives/frozen-archives.png
   :alt: Frozen Archives screenshot with example archives.
   :caption: Example of the Archives page with a finalised archive.
   :class: with-border
   :percentage: 80 95
```

Independent PDF sizing:

```
.. lightbox:: /images/architecture.png
   :alt: System architecture diagram.
   :caption: High-level architecture overview.
   :percentage: 60 90
   :latex-width: 0.8
```

Here the HTML thumbnail is 60% wide with a 90% overlay, while the PDF figure is constrained to 80% of `\linewidth`.

Document-relative path:

```
.. lightbox:: ../images/detail-view.png
   :alt: Detail view of the settings panel.
   :percentage: 50
```

5.4.5 Node Architecture

The extension defines four custom docutils nodes:

LightboxContainer
: Outer wrapper grouping all child nodes. In HTML: a . In LaTeX: handles the complete `figure` environment and raises `SkipNode` to prevent children from rendering a second time.

LightboxTrigger
: The thumbnail <img> and its <label> wrapper. Clicking opens the overlay via the CSS checkbox toggle. Suppressed outside HTML.

LightboxOverlay
: The full-size image, caption paragraph, close button, and backdrop label. Suppressed outside HTML; the container visitor handles LaTeX output directly.

LightboxCollector
: A wrapper around a standard `image` node. In HTML, it is present in the doctree during Sphinx's read phase so that `ImageCollector` registers the file and copies it to `_images/`; its visitor then raises `SkipNode` so it is not rendered. In LaTeX, the container visitor handles output and skips the children. In epub, text, man, and texinfo builders, the collector's child image supplies the plain image fallback.

Each node has visitor function pairs registered for `html`, `latex`, `epub`, `text`, `man`, and `texinfo` builders.

5.4.6 Content Security Policy (CSP)

The lightbox extension uses an external JavaScript file (`lightbox.js`) for keyboard activation, Escape-to-close, gallery navigation, and focus management. It does not inject inline JavaScript.

The generated HTML still uses inline styles for thumbnail width and overlay sizing. A strict CSP that disallows inline styles will block those declarations. If your documentation is served with CSP headers, allow inline styles for the documentation pages:

```
Content-Security-Policy: style-src 'self' 'unsafe-inline';
```

5.5 API Reference

This page documents the internal Python API of `sphinx-lightbox`. The public authoring interface is standard Sphinx `image` and `figure` markup plus the `.. lightbox::` directive. The Python API below is primarily useful for contributors.

5.5.1 Extension Setup

`lightbox.lightbox.`**`setup`**(*app: Sphinx*) → ExtensionMetadata

5.5.2 Nodes

class lightbox.lightbox.**LightboxContainer**(*rawsource: str = ''*, **children*, ***attributes: Any*)

Outer wrapper grouping the trigger thumbnail and the overlay.

class lightbox.lightbox.**LightboxTrigger**(*rawsource: str = ''*, **children*, ***attributes: Any*)

Thumbnail image that opens the lightbox when clicked.

class lightbox.lightbox.**LightboxOverlay**(*rawsource: str = ''*, **children*, ***attributes: Any*)

Full-size image overlay with caption and close control.

5.5.3 Directive

class lightbox.lightbox.**LightboxDirective**(*name*, *arguments*, *options*, *content*, *lineno*, *content_offset*, *block_text*, *state*, *state_machine*)

final_argument_whitespace = True

May the final argument contain whitespace?

has_content = False

May the directive have content?

option_spec = {'alt': <function unchanged>, 'caption': <function unchanged>, 'class': <function unchanged>, 'latex-width': <function unchanged>, 'percentage': <function positive_int_list>}

Mapping of option names to validator functions.

optional_arguments = 0

Number of optional arguments after the required arguments.

required_arguments = 1

Number of required directive arguments.

5.5.4 Transforms

lightbox.lightbox.**transform_lightbox_images**(*app: Sphinx*, *doctree: document*, *docname: str*) → None

Convert standard image/figure nodes with class `lightbox` for HTML builds.

lightbox.lightbox.**assign_lightbox_gallery**(*app: Sphinx*, *doctree: document*, *docname: str*) → None

Assign per-document gallery metadata to lightbox overlays.

5.6 Changelog

5.6.1 Version 0.5 Beta (2026-07-01)

First public beta of `sphinx-lightbox`.

- Standard Sphinx `image` and `figure` directives can be transformed into lightboxes for HTML output.
- The `.. lightbox::` directive is available with `:percentage:` and `:latex-width:` options.

- HTML output includes keyboard activation, Escape-to-close, focus management, focus trapping, and optional per-document gallery navigation through the external `lightbox.js` enhancement.
- Figure captions and legends remain visible on the page and are copied into lightbox overlays.
- Non-HTML builders keep standard image/figure behavior for class-based transforms. The `.. lightbox::` directive renders LaTeX figures for PDF output and plain images for other builders.
- Local image paths are constrained to the Sphinx source tree. Remote and `data:` images bypass lightbox transformation.
- Generated IDs and user-controlled HTML attributes are sanitized or escaped.
- Documentation is built with Sphinx and published at `https://aputtu.github.io/sphinx-lightbox/`.

5.7 Development Workflow

This project includes a small Makefile and shell helpers for repeatable local development. The Makefile is the public entry point; the scripts underneath it are useful when a single command needs to be run directly.

5.7.1 Environment Setup

Create the local virtual environment and install development, documentation, and editable package dependencies:

```
make setup
```

`make setup` recreates `venv/` and removes local build, tox, coverage, and packaging artifacts before installing dependencies. By default it uses `python3`. To choose another interpreter, set `PYTHON_BIN`:

```
PYTHON_BIN=python3.12 make setup
```

After setup, Makefile targets use tools from `venv/bin` instead of relying on globally installed commands.

The pytest configuration enforces 100% line and branch coverage for the `lightbox` package.

5.7.2 Make Targets

Run `make` or `make help` to print the available targets.

Target	Purpose
`make setup`	Recreate `venv/` and install development dependencies.
`make test`	Install the package in editable mode and run pytest.
`make lint`	Run `ruff check` and `ruff format --check`.
`make type`	Run `mypy lightbox`.
`make check`	Run lint, type checks, and tests.
`make html`	Refresh the PDF download if needed, build HTML docs, then validate the generated HTML.
`make pdf`	Build the LaTeX documentation and compile the PDF download.
`make docs`	Build PDF docs, build HTML docs, and validate the generated HTML.
`make validate`	Validate an existing `docs/_build/html` tree.
`make build`	Build source and wheel distributions, run `twine check`, and validate archive contents.
`make audit`	Run `pip-audit` against the local environment.
`make all`	Run checks, docs, package build, and dependency audit.
`make watch`	Start `sphinx-autobuild` for the documentation.
`make clean`	Remove generated build, cache, coverage, and packaging artifacts.
`make clean-all`	Clean generated artifacts, then run the full local gate.

5.7.3 Documentation Builds

The published GitHub Pages site includes a downloadable PDF, so the local docs targets keep the PDF and HTML output in sync. `make html` rebuilds the PDF first when `docs/_downloads/sphinx-lightbox.pdf` is missing or older than the documentation, Python, CSS, or JavaScript sources. That path requires a working LaTeX installation.

For HTML-only validation without LaTeX, use the same pattern as the CI docs check:

```
mkdir -p docs/_downloads
touch docs/_downloads/sphinx-lightbox.pdf
python -m sphinx -W --keep-going -E -a -b html docs docs/_build/html
python scripts/validate_docs.py docs/_build/html
```

`tox -e docs` also uses an HTML-only path with a placeholder PDF download.

5.7.4 Direct Script Usage

The Makefile delegates most commands to `scripts/dev.sh`. The direct form is useful in CI experiments or when invoking the workflow from another script:

```
./scripts/dev.sh check
./scripts/dev.sh docs
./scripts/dev.sh build
```

The script supports the same command names as the Makefile targets, without the `make` prefix.

5.7.5 Tox Matrix

Use tox when checking the supported Python and Sphinx combinations:

```
tox -p auto
```

The default matrix includes linting, type checking, documentation validation, and tests across Python 3.10 through 3.14 with supported Sphinx releases.

5.7.6 Release-Quality Local Gate

Before tagging a release, run the broad local gate when the system has the required tooling:

```
make all
tox -p auto
```

`make all` expects LaTeX for PDF generation and `pip-audit` for dependency auditing. The release checklist has the complete tagging and publishing steps.

5.8 Release Checklist

Use this checklist before tagging a release.

1. Confirm the package version in `pyproject.toml` and `lightbox/lightbox.py`.
2. Add a dated entry to `docs/changelog.rst` with user-visible changes, support notes, and security-relevant updates.
3. Run the local quality gates:

   ```
   python -m ruff check lightbox tests docs/_ext scripts/validate_docs.py
   python -m ruff format --check lightbox tests docs/_ext scripts/validate_
   ↪docs.py
   python -m mypy lightbox
   python -m pytest
   ```

4. Build and validate the HTML documentation:

   ```
   python -m sphinx -W --keep-going -E -a -b html docs docs/_build/html
   python scripts/validate_docs.py docs/_build/html
   ```

5. Rebuild the PDF download and then rebuild HTML so the download link points to the fresh PDF:

   ```
   ./scripts/dev.sh docs
   ```

6. Build and check distributions:

```
python -m build
python -m twine check dist/*
python scripts/validate_dist.py dist
```

7. Install the built wheel in a fresh environment and build a minimal Sphinx project using `extensions = ["lightbox"]`.
8. Run `pip-audit` for the release environment.
9. Confirm PyPI project name availability immediately before publishing.
10. Create the release commit, tag it as `vX.Y.Z`, and push the tag.
11. Verify that GitHub Actions finishes the test matrix, docs validation, GitHub Pages deployment, and PyPI trusted-publishing job.

5.9 License

`sphinx-lightbox` is licensed under the **GNU General Public License v3.0 or later (GPL-3.0-or-later)**.

```
Copyright (C) 2024-2026 Aputsiak Niels Janussen

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

The full license text is available in the `LICENSE` file in the source tree.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- search

INDEX