WCAG 2.2 Accessibility in Blogr

Target level: WCAG 2.2 Level AA — W3C Recommendation

Why accessibility matters

Web accessibility is not a feature — it is a fundamental property of the web. The World Wide Web was designed to work for everyone, regardless of hardware, software, language, location, or ability. Blogr builds on this foundation.

Who benefits

Group	Share of population	Disability type
Blind or low vision	~2.2bn worldwide	Visual
Deaf or hard of hearing	~430m worldwide	Auditory
Motor / mobility impairments	~1bn worldwide	Physical
Cognitive / learning disabilities	~15% of population	Cognitive
Temporary impairments (broken arm, bright sun, noisy environment)	Everyone at some point	Situational

Happytodev

In practice, accessibility improvements help **every user**. Captions help non-native speakers. High contrast helps in bright sunlight. Keyboard navigation helps power users. Skip links help everyone bypass repetitive navigation.

Legal context

WCAG 2.2 Level AA is the reference standard for:

- **European Accessibility Act (EAA)** — mandatory from June 2025 for all digital products sold in the EU
- **Americans with Disabilities Act (ADA)** — US case law consistently applies WCAG 2.1 AA as the benchmark
- **EN 301 549** — EU procurement standard for ICT, directly references WCAG 2.2 AA
- **Accessibility for Ontarians with Disabilities Act (AODA)** — mandates WCAG 2.0 AA by default, WCAG 2.2 AA recommended

How Blogr implements WCAG 2.2 AA

Blogr's accessibility implementation is built into every layer: templates, components, helpers, and tests. Each technique is chosen to be **maintainable**, **testable**, and **invisible** to the majority of users while being indispensable for users who rely on it.

Happytodev

Perceivable — Information must be presentable to users in ways they can sense

1.1.1 Non-text Content (Level A)

All images in Blogr carry `alt` attributes:

- Blog post thumbnails and feature images — `alt="{{ $post->title }}"`
- Gallery images (all 5 layouts: grid, horizontal, filtered, masonry, bento) — generated from image metadata
- Carousel slides — descriptive alt text per slide
- Team member photos — `alt="{{ $member->name }}"`
- Logo — `alt="{{ $siteName }}"`

No image in Blogr is ever rendered without an accessible name.

1.3.1 Info and Relationships (Level A)

Blogr uses semantic HTML5 elements throughout: `<nav>`, `<main>`, `<article>`, `<section>`, `<aside>`, `<h1>`–`<h6>`, `<ul>`, `<ol>`, `<label>`, `<time>`. ARIA roles supplement semantics where native HTML is insufficient:

- `role="menu"` / `role="menuitem"` on language switcher and mega menu
- `role="button"` on interactive gallery images
- `role="switch"` on plugin toggles
- `aria-current="page"` on breadcrumb active page
- Schema.org `BreadcrumbList` JSON-LD for structured navigation

1.3.5 Identify Input Purpose (Level AA)

Contact forms use `autocomplete` attributes so browsers and assistive technology can auto-fill user data:

Happytodev

- `autocomplete="name"` on the name field
- `autocomplete="email"` on the email field
- `autocomplete="subject"` on the subject field

1.4.1 Use of Color (Level A)

Color is never the sole means of conveying information:

- Required form fields use `*` + `sr-only "(required)"` text indicator
- Focus is indicated by `focus-visible:ring-2` (visible ring), not just color change
- Links are underlined in body text (not just coloured)

1.4.3 Contrast (Minimum) (Level AA)

Blogr ships with a **real-time colour contrast validator** built into the admin settings panel. When an admin changes any theme colour, the system:

1. Computes the relative luminance of the foreground and background colours using the WCAG formula (sRGB linearization)
2. Calculates the contrast ratio
3. Checks against the 4.5:1 threshold for normal text and 3:1 for large text (18px bold or 24px regular)
4. Displays a **warning notification** if the combination fails

Default colours are chosen to pass AA out of the box. The primary button colour was deliberately darkened from `#c20be5` to `#9b0ab8` to achieve 4.5:1 on white backgrounds.

1.4.11 Non-text Contrast (Level AA)

Interactive UI components meet the 3:1 contrast threshold:

- Input borders use `border-gray-400` (not `border-gray-300`) for minimum 3:1 against white backgrounds
- Focus rings use the primary colour with `ring-offset` for clear visibility
- Icon-only controls (social links, theme switcher) have adequate contrast against their backgrounds

Happytodev

1.4.13 Content on Hover or Focus (Level AA)

Dismissible overlays follow WCAG requirements:

- Mega menus use a **300ms close delay** (not instant) so users moving the mouse across the gap do not lose content
- Language switcher stays open until explicitly dismissed via `@click.away` or the Escape key
- All hover-triggered content is also triggerable via keyboard focus

Operable — User interface components and navigation must be operable

2.1.1 Keyboard (Level A)

Every interactive element in Blogr is keyboard-accessible:

Component	Keyboard controls
Carousel	Arrow keys (← →) to navigate slides, Tab to reach prev/next/dots
Gallery (all layouts)	Tab to images, Enter/Space to open lightbox, Escape to close, Arrow keys to navigate
Mega menu	Tab to trigger, Enter/Space to open, Escape to close, Tab within menu items
Language switcher	Tab to trigger, Enter/Space to open, Escape to close, Arrow keys between options
Mobile menu	Tab to hamburger, Enter/Space to toggle, Escape to close
Lightbox	Escape to close, Arrow keys to navigate images
Back-to-top	Tab to reach, Enter to activate

Happytodev

2.2.2 Pause, Stop, Hide (Level A)

The carousel respects user control:

- **Pause button** with `aria-label="Toggle autoplay"` — users can stop motion at any time
- **Focus-based pause** — when keyboard focus enters the carousel, autoplay stops automatically (`@focusin` / `@focusout`)
- **Hover-based pause** — mouse hover also pauses autoplay
- **`prefers-reduced-motion`** — if the user's system is set to reduce motion, autoplay never starts

2.4.1 Bypass Blocks (Level A)

A **skip-to-content link** is the very first focusable element on every page. It is visually hidden (`sr-only`) and appears only on focus (`focus:not-sr-only`). It targets `<main id="main-content">`, letting keyboard and screen reader users skip navigation, banners, and theme switcher in a single keystroke.

2.4.4 Link Purpose (In Context) (Level A)

Every link in Blogr has a clear purpose, identifiable from the link text alone or via an accessible name:

- Blog post cards: `aria-label="Read more about {{ post title }}"` — each Read More link is unique even when read out of context
- Social links: `aria-label="Twitter/X"`, `aria-label="GitHub"`, etc. — the icon-only links have descriptive labels
- RSS feed: `aria-label="RSS Feed"` + `title="RSS Feed"`
- Breadcrumb items: link text clearly describes the page

Happytodev

2.4.7 Focus Visible (Level AA)

Every interactive element has a visible focus indicator:

```
focus-visible:ring-2 focus-visible:ring-primary-500
focus-visible:ring-offset-2 focus:outline-none
```

This pattern is applied to: navigation items, mega menu items, language switcher, social icon links, blog post card links, category/tag badges, carousel controls (prev, next, dots, pause), gallery images, gallery filter buttons, lightbox controls, back-to-top button, footer links, and plugin toggles.

2.4.11 Focus Not Obscured (WCAG 2.2, Level AA)

When the skip-to-content link is activated, the target scrolls to `<main id="main-content">` with `scroll-margin-top: 5rem` — enough to clear the sticky navigation bar. Focused elements are never hidden behind the fixed header.

2.5.3 Label in Name (Level A)

The accessible name of every control matches its visible label:

- Theme switcher buttons: `aria-label="Light mode"`, `aria-label="Auto mode"`, `aria-label="Dark mode"` (matches the icon meaning)
- Social icons: `aria-label` matches the platform name visibly identifiable from the icon

Screen reader users and voice control users can reliably predict what each control does.

Happytodev

2.5.8 Target Size (Minimum) (WCAG 2.2, Level AA)

All interactive targets are at least **24×24 CSS pixels**:

- Social icon links: `min-w-[24px] min-h-[24px]` on every anchor
- Combined with `inline-flex items-center justify-center` for proper centring within the minimum bounding box

This ensures users with motor impairments, tremors, or using touch interfaces can reliably activate controls without accidental taps.

Understandable — Information and operation of the UI must be understandable

3.3.2 Labels or Instructions (Level A)

Every form control has an associated label:

- **Newsletter**: `<label for="newsletter-email" class="sr-only">Email address</label>` — visually hidden but available to screen readers
- **Contact form (name, email, subject, message)**: visible `<label>` elements with matching `for` / `id` pairs
- **Required fields**: `*` + `sr-only "(required)"` indicator
- `aria-required="true"` on all required form fields

3.3.4 Error Prevention (Legal, Financial, Data) (Level AA)

Form submissions validate on the server side. The contact form includes server-side validation with clear error messages. Destructive actions in the admin panel require confirmation.

Robust — Content must be interpretable by a wide variety of user agents

Happytodev

4.1.2 Name, Role, Value (Level A)

All custom interactive widgets expose proper name, role, and value to assistive technology:

Widget	Role	Name	Value
Theme switcher buttons	button	`aria-label`	`aria-pressed` (true/false)
Gallery images	button	`aria-label="Open image N"`	—
Plugin toggles	switch	`aria-label="Toggle ExtensionName"`	`aria-checked` (true/false)

Widget	Role	Name	Value
Mega menu trigger	button	link text	`aria-expanded` (true/false)
Mobile menu toggle	button	`aria-label="Open menu" / "Close menu"`	`aria-expanded` (true/false)
Language switcher	menu/menuitem	flag + language name	`aria-expanded`
Carousel dots	button	`aria-label="Go to slide N"`	—
Carousel pause	button	`aria-label="Toggle autoplay"`	—
Breadcrumb current	—	—	`aria-current="page"`

Happytodev

4.1.3 Status Messages (Level AA)

Dynamic content updates are announced to screen readers:

- **Contact form**: success/error messages use `role="status"` — the message is announced when it appears
- **Newsletter**: "Subscribed!" message uses `role="status"`
- **Code copy button**: "Copied!" dynamically sets `role="status"` on the confirmation text

None of these messages require the user to move focus — they are announced automatically.

Impacts for Blogr site owners

Immediate benefits

Area	Impact
SEO	Semantic HTML, alt text, heading hierarchy, and descriptive links improve search engine understanding and rankings
Audience	~15–20% of web users have some form of disability — an accessible site serves them directly
Legal compliance	EAA (EU, June 2025), ADA (US), AODA (Ontario), EN 301 549 (EU procurement) — WCAG 2.2 AA compliance mitigates legal risk
Performance	Skip links, keyboard navigation, and reduced motion improve experience for ALL users
Brand perception	Public accessibility commitment builds trust and demonstrates inclusion
Future-proofing	WCAG 2.2 is backward-compatible — the same techniques apply to future versions

Happytodev

What Blogr does not require from you

- **No extra configuration** — WCAG features are built into every template and component. They work out of the box
- **No third-party services** — no overlay widgets, no accessibility plugins. Blogr is natively accessible
- **No developer effort** — colour contrast validation is built into the admin settings. Upload your logo, pick your colours, and Blogr tells you if they pass AA

What you should still verify

- **User-provided content** — images uploaded by authors must have alt text. Blogr requires it in the post editor but does not add alt text automatically

- **Custom blocks** — if you build custom blocks, follow the same patterns (semantic HTML, ARIA labels, keyboard handlers, focus styles)
- **Third-party integrations** — embedded iframes, widgets, or tracking scripts are outside Blogr's control

Conformance status

Level	Status	Details
A	✅ Complete (18/18 criteria)	All Level A success criteria implemented
AA	✅ Complete (20/20 criteria applicable to a blog system)	All implementable AA criteria addressed
AAA	⬜ Not targeted	Not required by any current regulation

Happytodev

Criteria not applicable

Some WCAG 2.2 Level AA criteria do not apply to Blogr because they concern functionality the system does not offer:

- **3.3.3 Error Suggestion** — Blogr validates server-side with clear messages; no auto-correction suggestions (not a search engine or e-commerce checkout)
- **3.3.4 Error Prevention** (Legal/Financial) — Blogr does not process legal transactions, financial data, or user-data deletion requests (those belong to the hosting application)
- **3.2.1 On Focus** / **3.2.2 On Input** — No context-changing events occur on focus or input

Testing

Accessibility tests live in `tests/Feature/Accessibility/` (13 test files) and run as part of the standard test suite:

```
vendor/bin/pest --filter "Accessibility"
```

Each fix follows strict TDD: the test is written first (RED), the implementation makes it pass (GREEN), then the implementation is temporarily reverted to confirm the test fails without it (anti-false- positive gate).

All tests are **static HTML assertion tests** — they render Blade components and verify ARIA attributes, roles, keyboard handlers, and CSS classes in the output. No browser is required, which keeps them fast and suitable for pre-commit hooks and CI.

Test coverage by criterion

Happytodev

Criterion	Test file
2.4.1 Bypass Blocks	`SkipToContentLinkTest.php`
3.3.2 Labels / 1.3.5 Input Purpose	`NewsletterFormLabelTest.php`, `FormAutocompleteTest.php`
2.1.1 Keyboard / 4.1.2 Name Role Value	`GalleryKeyboardTest.php`, `CarouselKeyboardTest.php`, `MegaMenuKeyboardTest.php`, `LanguageSwitcherKeyboardTest.php`
2.2.2 Pause Stop Hide	`CarouselPauseButtonTest.php`
2.4.4 Link Purpose	`ReadMoreLinkAriaLabelTest.php`
4.1.2 ARIA states	`ThemeSwitcherAriaPressedTest.php`

Criterion	Test file
4.1.3 Status Messages	`StatusMessagesAriaLiveTest.php`
2.4.11 Focus Not Obscured	`ScrollMarginTopTest.php`
2.5.8 Target Size	`TargetSizeTest.php`

Architecture decisions

Why not an overlay / plugin?

Accessibility overlays (widgets that claim to "fix" accessibility via JavaScript) are widely criticised by the disability community. They cannot fix underlying code issues, often introduce new barriers, and create a false sense of compliance. Blogr's approach is **native accessibility** — the HTML, CSS, and Alpine.js behaviours are correct at the source.

Happytodev

Why static HTML tests instead of browser tests?

Browser-based a11y tests (axe-core, Lighthouse) are more thorough but require a running server, Playwright, and significant CI resources. For a package that runs 600+ tests in <30s, adding 13 quick HTML assertion tests ensures accessibility is checked on every commit without slowing down development. Browser-level audits are reserved for the release process.

Why `focus-visible` instead of `focus`?

The `:focus-visible` pseudo-class applies focus styles only when the browser determines the user needs to see it (keyboard navigation, not mouse click). This avoids "focus soup" for mouse users while providing clear focus indicators for

keyboard users. The `focus:outline-none` fallback ensures the default UA outline is removed only when the custom `focus-visible` ring replaces it.

A note from the author

This documentation reflects the current state of Blogr's accessibility implementation. Despite thorough testing, WCAG conformance is complex and context-dependent — user-provided content, third-party integrations, and custom configurations may introduce gaps that automated checks and static tests cannot catch.

If you encounter an accessibility barrier while using Blogr, please open an issue so it can be addressed. I remain available for questions, bug reports, and feedback.

References

Happytodev

- WCAG 2.2 Specification — the standard
- Understanding WCAG 2.2 — official techniques and examples
- WebAIM Contrast Checker
- WCAG 2.2 Map (GitHub)
- European Accessibility Act
- EN 301 549
