UK WEB MARKETING · RESOURCE LIBRARY

Small-business website accessibility checklist (WCAG 2.2, 2026)

This is the accessibility checklist I run when I build or take over a UK small-business website. It is built around the standard everyone in this space actually uses — WCAG 2.2 at level AA, the Web Content Accessibility Guidelines published by the W3C. I am not interested in the version that gets a logo on your footer and changes nothing. I am interested in whether a real person using a keyboard, or a screen reader, or a phone held at arm's length, can find your phone number and fill in your form. That is the whole job.

A GUIDE FOR UK OPERATORS

Author
Jordan Gilbert, CTO · UK Web Marketing

Edition
v1.0 · June 2026

Audience

UK small businesses and the people who run their websites.

TABLE OF CONTENTS

---

01 The checklist at a glance

WCAG is organised under four principles — content must be **Perceivable, Operable, Understandable, and Robust** (POUR). The checklist below is grouped the same way so you can see where each item belongs.

#	ITEM	LIVES UNDER
1	Every meaningful image has alt text; decorative images are marked empty	Perceivable
2	Text colour contrast meets the AA minimum against its background	Perceivable
3	Information is never conveyed by colour alone	Perceivable
4	Video has captions; audio has a transcript	Perceivable
5	The page reflows to a single column at 320px without horizontal scroll	Perceivable
6	Text can be resized to 200% without breaking the layout	Perceivable
7	Headings follow a logical order (one h1, then h2, h3 in sequence)	Perceivable
8	Everything works with a keyboard alone — no mouse required	Operable
9	There are no keyboard traps you cannot tab out of	Operable
10	The keyboard focus indicator is always clearly visible	Operable
11	A "skip to main content" link is available	Operable
12	Tap and click targets are large enough and not cramped together	Operable
13	Nothing important relies on hover or a precise mouse path	Operable
14	No content flashes more than three times a second	Operable
15	The page has a descriptive, unique title	Operable
16	Link text makes sense out of context (no bare "click here")	Understandable
17	The page declares its language (lang attribute)	Understandable
18	Form fields have visible, programmatically-linked labels	Understandable
19	Errors are identified in text and explain how to fix them	Understandable
20	Navigation and naming are consistent across pages	Understandable
21	The markup is valid and uses correct HTML elements for the job	Robust
22	Custom widgets expose a name, role, and state to assistive tech	Robust

02 Why accessibility matters — the duty and the upside

There are two reasons to care, and they point the same way. One is a legal duty. The other is that an accessible site simply reaches more of the people you are paying to reach.

2.1 The Equality Act 2010 duty — read it properly

Under the **Equality Act 2010**, businesses that provide services to the public are "service providers" and owe a duty to make **reasonable adjustments** so that disabled people are not put at a substantial disadvantage. The widely-held reading — including in guidance aimed at businesses — is that this duty extends to the services you deliver through a website, not only your physical premises. I am stating that as the prevailing interpretation, not as settled fact for your specific situation: what counts as a "reasonable" adjustment is judged case by case and weighs cost and practicality. For anything load-bearing, read the Act itself on legislation.gov.uk and take proper advice.

> A note on scope. This is **not only a public-sector rule.** The Public Sector Bodies (Websites and Mobile Applications) Accessibility Regulations 2018 are a separate, stricter regime that applies to public bodies — councils, the NHS, schools, and so on — and they explicitly require WCAG conformance plus a published accessibility statement. If you are a private small business, those regulations are not your obligation, but they are useful as a benchmark, because they adopt the same WCAG standard I am pointing you at here.

So: the public-sector regime is the strict, named-standard one. The private-business position rests on the Equality Act's general "reasonable adjustments" duty, which is principle-based rather than a checklist. WCAG 2.2 AA is the practical yardstick most people use to demonstrate they have done something reasonable — which is exactly why this guide is built around it.

2.2 The upside is bigger than the duty

I would do this work even if the law said nothing, because the numbers around it are commercial.

- **More customers.** A meaningful share of UK adults report a disability, and many more have a temporary or situational limitation — a broken wrist, bright sunlight on a phone screen, a noisy train. An inaccessible checkout turns those people away silently. You never see the lost sale; it just does not happen.
- **Better SEO.** Search engines and screen readers want the same things: real text instead of text-in-images, descriptive alt attributes, a sane heading structure, meaningful link text, fast and well-marked-up pages. Accessibility work and technical SEO overlap heavily, so you are paying once for two outcomes.
- **A more robust site.** Semantic, valid markup that exposes names and roles to assistive technology is also easier to maintain, test, and hand to the next developer.

I am not going to quote you a precise market-size figure or a percentage uplift — those numbers get repeated until they detach from any source. The direction is well established; the exact size depends on your sector and your audience.

03 WCAG 2.2 AA — the practical subset

WCAG 2.2 at level AA has a lot of success criteria. You do not need to memorise the numbers. You need to get the handful below right, because they are the ones that actually decide whether a small-business site is usable. I am naming each requirement in plain terms rather than reciting criterion numbers — naming the requirement is harder to get wrong.

3.1 Colour contrast

Body text must stand out clearly from its background; the AA standard sets a minimum contrast ratio for normal text and a slightly lower one for large text. Light-grey text on white is the single most common failure I find. Use a contrast checker (the WebAIM one is free) and stop guessing.

3.2 Text alternatives (alt text)

Every image that carries meaning needs an `alt` attribute describing what it conveys — not "image1.jpg". Images that are purely decorative get an empty `alt=""` so screen readers skip them rather than announcing a filename. A logo's alt is the business name. A photo of a product is what the product is.

3.3 Keyboard operability

Everything you can do with a mouse, you must be able to do with the keyboard alone — Tab to move, Enter or Space to activate, arrow keys inside menus and sliders. Unplug your mouse and try to complete a booking or a purchase. If you get stuck, so does everyone who cannot use a mouse.

3.4 Visible focus

When you Tab through a page, the element you have landed on must show a clear visible outline. A shocking number of "designer" sites strip the focus ring off with CSS because it is considered ugly. Removing it without replacing it makes the site unusable by keyboard. Style it if you must; never delete it.

3.5 Form labels

Every input needs a label that is both visible and programmatically associated with the field (a `<label for="…">`, or an appropriate `aria-label`). Placeholder text is not a label — it vanishes when you start typing and many screen readers ignore it.

3.6 Logical heading structure

One `<h1>` per page, then `<h2>` and `<h3>` in a sensible nesting order. Headings are how screen-reader users skim a page; they jump heading to heading. Do not pick a heading level because of how big it looks — pick it for the structure, then style the size.

3.7 Link purpose

Link text should make sense read on its own. Screen-reader users can pull up a list of every link on the page; a list of fifteen identical "click here" links is useless. "Read our returns policy" tells you where it goes; "click here" does not.

3.8 Resize and reflow

Content must reflow into a single readable column on a narrow viewport (the standard checks down to 320px wide) without forcing horizontal scrolling, and text must survive being zoomed to 200% without overlapping or getting cut off. If your site already works properly on a phone, you are most of the way there.

3.9 Target size

Interactive controls — buttons, links, menu items — need to be large enough to hit comfortably and not packed so tightly that a fat finger or a tremor catches the wrong one. WCAG 2.2 added a specific minimum-target-size requirement; the practical version is "make your tap targets generous, especially on mobile".

3.10 No keyboard traps

If keyboard focus enters a component — a modal, a date picker, a video player — it must be possible to get back out with the keyboard alone. A modal that traps Tab inside itself with no way to close it by keyboard is a dead end. Test every popup by trying to escape it without the mouse.

---

04 The common failures I find on small-business sites

These are the ones I see again and again. None is hard to fix; most were introduced by a theme or a plugin nobody audited.

- **Carousels and auto-rotating sliders.** The hero slider that flips every four seconds is an accessibility and usability problem at once: it moves before slow readers finish, it is fiddly to control by keyboard, and the content hidden on later slides is content most visitors never see. If you keep one, it must be pausable, keyboard-operable, and not the only place important information lives. I usually just remove it.
- **Poor contrast.** Pale grey body text, white text over a busy photo, brand-colour buttons that fail against their background. It looks elegant in the designer's eye and is unreadable in sunlight or to anyone with low vision.
- **Missing or junk alt text.** Whole galleries with no alt attributes, or alt text that is the camera's filename. Product photos with no description. The site looks fine to you and is half-invisible to a screen reader and to Google.
- **Click-only menus.** Dropdown navigation that only opens on mouse hover and cannot be reached or opened by keyboard. The whole second level of the site becomes unreachable without a mouse.

- **PDF-only content.** Your menu, your price list, or your opening hours living only inside a PDF — often a scanned image of a document with no real text in it at all. That content is hard for screen readers, bad for mobile, and invisible to search. Put it in HTML on the page; keep the PDF as an extra if you like.
- **Icon buttons with no label.** A bare hamburger icon, a magnifying-glass search button, a basket icon — with no accessible name. A screen reader announces "button" and nothing else. Add an `aria-label` ("Open menu", "Search", "View basket") so the control says what it does.

---

05 Forms + interactive elements

Forms are where accessibility either earns its keep or costs you the enquiry. A contact form or a checkout that a screen-reader or keyboard user cannot complete is a lost customer at the exact moment they were trying to buy.

5.1 Labels — visible and linked

Every field has a visible label, and that label is tied to the field in the markup so assistive tech announces it. Do not rely on placeholder text as the only label, and do not rely on layout proximity alone — the association has to be in the code, not just on the screen.

5.2 Error messages that help

When a submission fails, the errors must be communicated in text — not by turning a field border red and nothing else, because colour alone is invisible to many users. Say what went wrong and how to fix it: "Enter an email address in the format name@example.com", not "Invalid input". Place the message next to the field and reference it from the field so a screen reader reads it out.

5.3 Focus management

When an error appears, or a modal opens, move keyboard focus somewhere sensible — to the first error, or into the modal — and make sure focus can return to where it started when the modal closes. Focus that silently jumps to the top of the page, or vanishes entirely, strands keyboard users. This is the subtle one most themes get wrong, and it is worth testing by hand.

5.4 Group related fields

Radio buttons and checkboxes that belong together (a set of delivery options, for example) should be grouped with a `<fieldset>` and a `<legend>` so the question is announced along with each choice. A lone "Standard / Express" pair with no group label leaves a screen-reader user guessing what they are choosing between.

---

06 How to test — the tools I actually use

Automated tools are fast and they catch real bugs. They do not catch everything — by common estimates the automated tools find only a portion of all issues, and the rest need a human. So I run the automated pass first to clear the obvious faults, then I test by hand for the things a machine cannot judge: whether alt text is *meaningful*, whether focus order makes *sense*, whether the page is actually *usable*.

6.1 Automated scanners

TOOL	WHAT IT IS, IN PLAIN TERMS
axe DevTools	Browser extension from Deque; runs a rules-based scan and explains each issue. The engine behind many other tools
Lighthouse	Built into Chrome DevTools; its Accessibility audit uses the axe engine and gives a quick score plus a checklist
WAVE	WebAIM's evaluation tool (browser extension and web version); overlays icons on the page so you can see issues in context

Run all three if you like — they overlap but each surfaces slightly different things. Treat the scores as a floor, not a finish line.

6.2 Keyboard-only navigation

Put the mouse down. Use Tab, Shift+Tab, Enter, Space, and the arrow keys to get through the whole site — open the menu, fill the form, complete the booking, dismiss the cookie banner. Watch the focus indicator the entire time. If you ever lose track of where you are, or get stuck, that is a bug a scanner will not always report.

6.3 A real screen reader

Turn one on and listen to your own page. **VoiceOver** is built into macOS and iOS (free, already installed). **NVDA** is a free, widely-used screen reader for Windows. You do not need to be an expert — even a clumsy ten-minute pass will reveal unlabelled buttons, meaningless link lists, and images announced as filenames. Nothing substitutes for hearing your site read aloud.

6.4 The human checks a machine cannot do

Is the alt text actually describing what matters in the image? Does the heading order tell the real story of the page? Can someone with a tremor hit your buttons? Is the contrast fine on a cheap phone in daylight? These are judgement calls. Budget time for them; they are where the real accessibility lives.

07 What I do NOT do — the anti-checklist

A few opinionated negatives. These are positions I take when I am the one building the site.

- **I do not install third-party accessibility "overlay" widgets.** These are the bolt-on scripts — sold under names like accessibility toolbars or one-line "make your site compliant" plugins — that add a floating button claiming to fix accessibility automatically. I do not use them, and I will talk a client out of them. They are **widely criticised by accessibility practitioners and disabled users**: they frequently fail to fix the underlying problems, can interfere with the assistive technology people already use, and a large coalition of accessibility experts has publicly recommended against them (see the "Overlay Fact Sheet" at overlayfactsheet.com, signed by hundreds of practitioners). An overlay treats the symptom — the score, the badge — not the cause. The fix is in the site's own markup, and that is where I put it.
- **I do not chase a perfect automated score and call it done.** A 100 in Lighthouse with unreadable alt text and a broken focus order is not an accessible site; it is a passing robot test. The score is a starting point.
- **I do not bury content in image-only PDFs.** Opening hours, prices, menus, and key information go into real HTML text on the page. A scanned PDF is the least accessible way to publish anything.
- **I do not strip the focus outline for looks.** If the default ring clashes with the design, I style a better one. I never remove it, because that single line of CSS quietly locks every keyboard user out.
- **I do not treat accessibility as a one-off launch task.** Every new page, plugin, and theme update can reintroduce problems. It is a habit, not a milestone — which is why the checklist above is meant to be re-run, not filed away.

---

08 Sources + further reading

- W3C — Web Content Accessibility Guidelines (WCAG) 2.2 · `w3.org/TR/WCAG22/`
- W3C Web Accessibility Initiative — WCAG 2 overview (plain-language introduction) · `w3.org/WAI/standards-guidelines/wcag/`
- W3C WAI — "How to Meet WCAG" quick reference (filterable checklist) · `w3.org/WAI/WCAG22/quickref/`
- WebAIM — articles, the contrast checker, and the WAVE evaluation tool · `webaim.org`
- GOV.UK — Understanding accessibility requirements for public sector bodies · `gov.uk/guidance/accessibility-requirements-for-public-sector-websites-and-apps`
- GOV.UK Service Manual — Making your service accessible (dos and don'ts, testing) · `gov.uk/service-manual/helping-people-to-use-your-service/making-your-service-accessible-an-introduction`
- Equality Act 2010 — the primary legislation · `legislation.gov.uk/ukpga/2010/15/contents`

- Equality and Human Rights Commission — guidance on the Equality Act · `equalityhumanrights.com`
- axe DevTools (Deque) · `deque.com/axe/devtools/`
- The Overlay Fact Sheet — practitioner statement on accessibility overlays · `overlayfactsheet.com`

A note on the long-form version

This is the v1.0 edition. The long-form version (planned for late 2026) will include worked examples — a real small-business site before and after, the keyboard-test script I run on every build, and the accessibility statement template I give clients who want one. If you want to be told when it ships, message me at `ukwebmarketing.com` — same person who wrote this.