# How to Check Text Contrast From a Picked Color Before Using It in UI

Before you commit any color to your UI, run a quick contrast check. Here's why it matters, how WCAG ratios work, and the exact workflow to go from a picked HEX value to an accessible, readable design.

Author: Lubor Šimůnek · Published: August 29, 2026 · Design

Online version: https://appstoryo.com/en/blog/check-text-contrast-picked-color-ui



## Why Text Contrast Matters More Than You Think

Color makes a UI feel alive, but the wrong combination can make it completely unreadable. Around 300 million people worldwide have some form of color vision deficiency, and many more struggle with low-contrast text in bright sunlight or on low-quality screens. Designing for contrast is not just a nice-to-have — in many regions it is a legal requirement. The EU Accessibility Act, the US Section 508 standard, and similar legislation increasingly mandate that digital products meet minimum contrast thresholds.

The practical standard everyone follows is **WCAG 2.1** (Web Content Accessibility Guidelines). It defines two conformance levels that apply directly to text:

- **Level AA** — 4.5:1 contrast ratio for normal text, 3:1 for large text (18 px regular or 14 px bold and above). This is the baseline target for most products.
- **Level AAA** — 7:1 for normal text, 4.5:1 for large text. Aim here when your audience includes older users or low-vision users.

Graphic decorations, logos, and disabled UI elements are exempt, but every piece of meaningful text should clear at least Level AA.

## How Contrast Ratio Is Actually Calculated

The ratio is based on *relative luminance* — a measure of how much light a color appears to emit, weighted to reflect how the human eye perceives brightness. Pure white has a luminance of 1.0, pure black is 0.0. The contrast ratio formula is simply `(L1 + 0.05) / (L2 + 0.05)`, where L1 is the lighter color. You don't need to do the math yourself — every contrast checker handles it — but understanding the concept explains why two similarly bright colors always produce a low ratio even if they look different in hue.

## Common Mistakes Designers Make

The most frequent offenders are subtler than you'd expect:

- **Light gray on white.** A popular minimalist choice that often fails at ratios as low as 1.5:1. Placeholder text in forms is a repeat offender.
- **Brand-colored buttons with white text.** A vivid medium-blue (#4A90E2) paired with white (#FFFFFF) yields roughly 3.1:1 — it feels fine visually but fails WCAG AA for body-sized text.
- **Colored badges on colored backgrounds.** A green success badge on a light-green panel can disappear entirely for users with deuteranopia.
- **Dark mode assumptions.** A color that passes in light mode may fail completely when the background flips to dark.

## A Practical Workflow: From Picked Color to Passing Ratio

Here is the step-by-step process that fits into any design workflow, whether you are working in Figma, a code editor, or a native app project.

1. **Pick the background color.** Open Color Picker Camera: HEX RGB and point the camera at the surface — a physical swatch, a printed mockup, or another screen. Tap once to lock the reading and copy the HEX value. The app also gives you the HSL breakdown immediately, which is useful in the next steps.
2. **Pick the foreground (text or icon) color** the same way. If you are starting from scratch, note the HSL lightness value (the L component) for both colors.
3. **Run the pair through a contrast checker.** Options include the WebAIM Contrast Checker, the Accessibility panel in Chrome or Firefox DevTools (right-click an element → Inspect → Accessibility), or the Contrast plugin in Figma. Paste both HEX values and read the ratio.
4. **Adjust lightness in HSL until you pass.** This is where HSL shines. If your blue button (#4A90E2, HSL roughly 210°, 72%, 59%) fails, keep the hue and saturation identical and simply lower the L value — say to 38% — giving you a darker, still-recognizably-blue button that clears 4.5:1 with white text. No guessing, no hue drift.
5. **Verify in context.** Paste the adjusted color back into your design and check it at actual font sizes. A ratio that barely passes at 4.51:1 gives you no safety margin — aim for 5:1 or above when possible.

## Why HSL Is the Right Model for Contrast Work

RGB values give you no intuitive handle on brightness. HEX is compact but opaque. HSL splits color into hue (the color itself), saturation (how vivid), and lightness (how bright). When you want to darken a color to improve contrast without changing its identity, you touch only the L channel. Color Picker Camera: HEX RGB displays HEX, RGB, HSL, and CMYK simultaneously, so you always have the HSL values ready the moment you pick a color — no manual conversion needed.

## Real-World Examples

Consider two common button choices. A medium blue (#4A90E2) with white text looks vibrant and modern, but the contrast ratio is approximately 3.1:1 — failing AA for normal text. Shift the blue to a dark navy (#1A3A6B) and the ratio jumps to about 9.4:1, comfortably exceeding even AAA. The button still reads as blue; it simply reads as text too.

The same logic applies to body copy. Medium gray (#9B9B9B) on white (#FFFFFF) sits at roughly 2.8:1. Darken it to #767676 and you hit exactly 4.54:1 — the minimum passing AA value and a well-known benchmark in accessibility circles.

## A Note on WCAG 3.0 and APCA

The next generation of guidelines introduces **APCA** (Advanced Perceptual Contrast Algorithm), which accounts for font weight, size, and polarity (light-on-dark vs dark-on-light) in a more nuanced way than the current ratio formula. APCA is already available as an experimental mode in some tools. WCAG 2.1 AA remains the enforceable standard for now, but if you are building something designed to last, it is worth testing against APCA thresholds as well.

## Start With an Accurate Color Reading

None of the above workflow works if your starting HEX value is wrong. A screenshot color-picked in a design tool, a color sampled from a compressed JPEG, or a value typed from memory can all be subtly off — enough to make a failing combination appear to pass. Using Color Picker Camera: HEX RGB to capture colors directly from the source — physical materials, rendered screens, or reference images — gives you the precise HEX and HSL values you need before you even open a contrast checker. It is free, has no ads, and the one-tap copy workflow means you spend your time designing, not transcribing hex codes.

**Cite as:** Lubor Šimůnek (2026). *How to Check Text Contrast From a Picked Color Before Using It in UI*. AppStoryo, lapnito.cz s.r.o.. https://appstoryo.com/en/blog/check-text-contrast-picked-color-ui
© 2026 lapnito.cz s.r.o.