Van: Erik D. Kennedy erik@learnui.design
Onderwerp: 3 techniques for UI text color contrast
Datum: 3 maart 2020 om 18:42
Aan: Jeroen jeroen@disch.nl

Hey Jeroen,

One aspect of software design that's quickly becoming required knowledge is *accessibility* – that is, designing apps so that people with disabilities can still use them.

[🎉 **ANNOUNCEMENT** 📣: Learn UI Design will open for new enrollments at midnight tonight!]

Obviously, "disabilities" is not a monolith. Making an app that can be navigated by a blind person is a pretty different task from, say, allowing someone with a broken arm to navigate your app entirely by keyboard – yet both fall under the banner of accessibility.

Today I want to talk about one important aspect of accessibility. And it's the one I'm *most frequently* asked about: **text color contrast**.

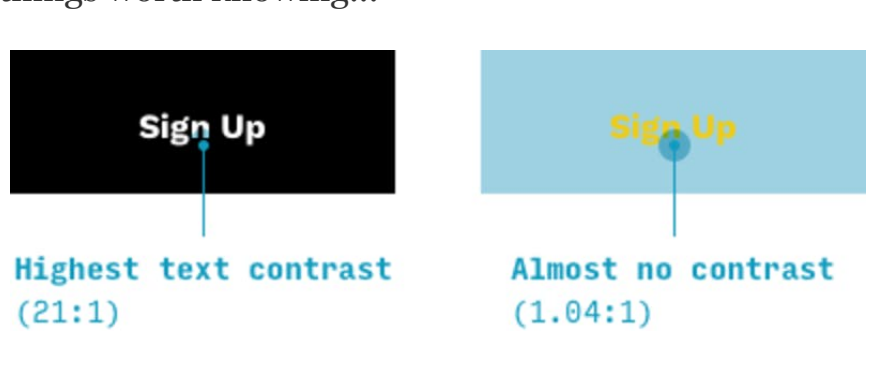
First, I'll give brief explanation as to what the heck text color contrast *is*, then we'll look at fixing contrast issues in an actual design.

What is text color contrast?

Ready? OK, so it turns out you can measure the difference between two colors in a *ratio of their luminosities*. You should think of luminosity as sort of a "natural lightness" to the color (apart from the "lightness" measurement of, say, the HSL color system, if you're familiar with that).

Darker variations of a color contrast *more* with white. And vice versa. I can assume you're with me so far? 😉

Two other things worth knowing...



First, the HIGHEST contrast ratio is 21:1, which is the measure of white-vs-black.

Second, the lowest contrast ratio is 1:1. And there are two ways to get this. The first way is to measure the contrast ratio between the one color *and itself*. Well, duh, that's 1:1.

The second, however, is to measure one color against any other with an *equivalent luminosity*. Since luminosity is about *lightness* – not hue – you can actually find pairs of colors with **different hues but equivalent luminosities** – e.g. one is teal and the other yellow. Again, that would be a contrast ratio of 1:1.

Now **different pieces of text** need to **contrast different amounts** against their backgrounds in order to meet WCAG, the industry standard of accessibility guidelines.

But how much?

Well, usually...

- **Large or bold text** needs to contrast **3:1** against its background
- **Body text (or smaller)** needs to contrast **4.5:1** against its background

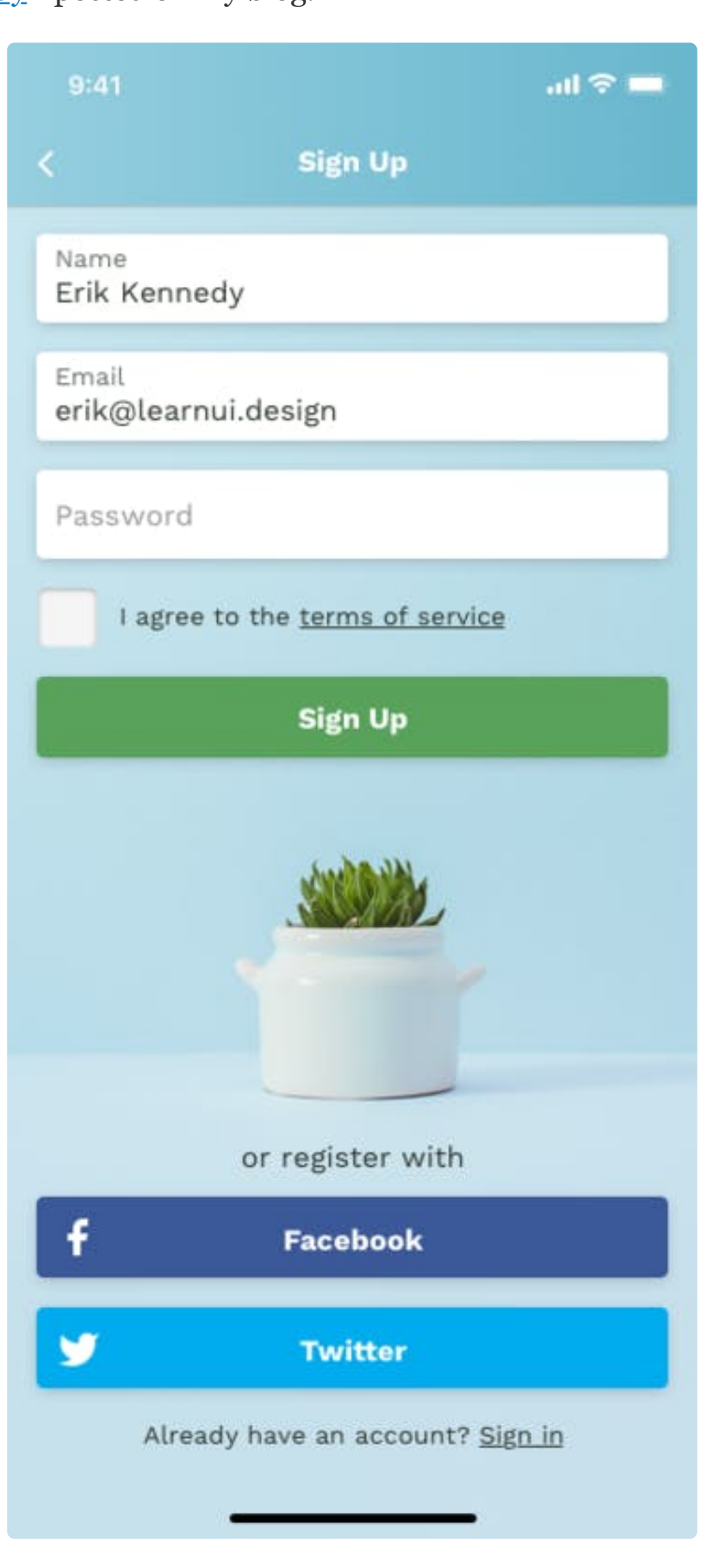
But there are two caveats here:

1. Large text refers to text that is 18pt+ at normal weights, or 14pt+ at bold weights. Anything smaller fits into the "normal text" category. I don't often have bold text at sizes smaller than 14pt. So this basically boils down to: "18pt or bold" means "big" and *everything else* falls under "normal".
2. The numbers given above are to meet WCAG at a "AA" level. If you want to achieve a "AAA" level, replace the "4.5" above with a "7" and the "3" above with a "4.5". This is a bit of a technicality, honestly. In my experience, very few clients are trying to hit AAA accessibility levels.

(I sort of hate rigorously listing exceptions before actually looking at something practical, but we're trying to meet a set of guidelines designed by a committee, so we have to play by their rules here)

Introducing...

So here's the design we'll be looking at today. Maybe you recognize it from this case study I posted on my blog.



In first starting to look for text contrast issues, it's often a good idea just to ask "**What's most difficult to read?**" and just list out some guesses.
In this example, I think the following strings are candidates, in roughly this order of difficultly to read:

1. The "Password" textbox hint
2. The "Name" and "Email" labels
3. The "Sign Up" page title
4. Maaaaybe the "Twitter" button label

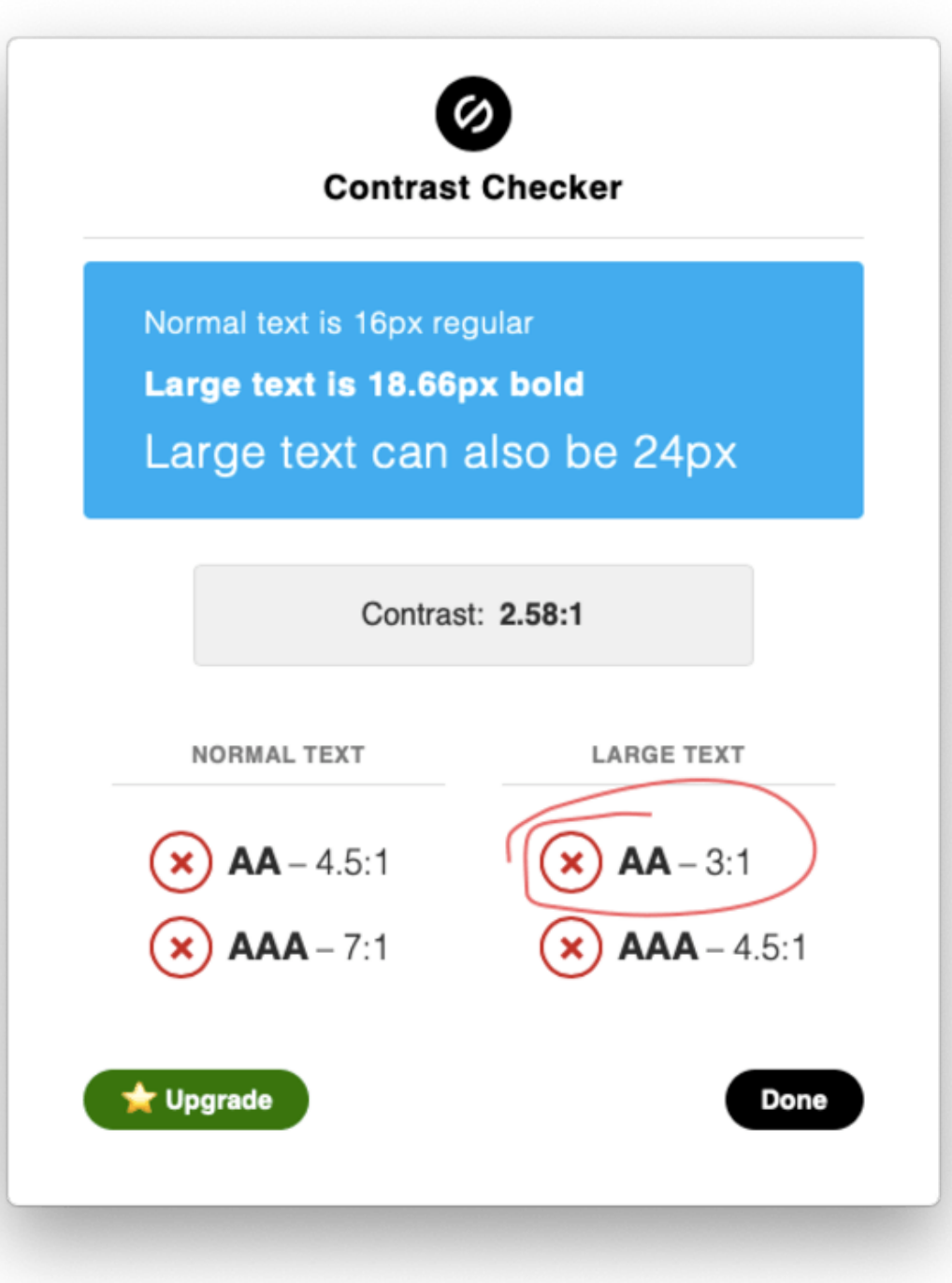
You can see I'm not even sure about the "Twitter" button. Maybe it's 3:1, maybe it's not (which is the target number, as this text is *bold*). I'm just going off my past experience measuring contrast here.

Nonetheless, let's start with the Twitter button 🙂

Fixing color contrast with Stark & The Accessible Color Generator

To measure contrast, I use the free Sketch/Figma/XD plugin Stark.

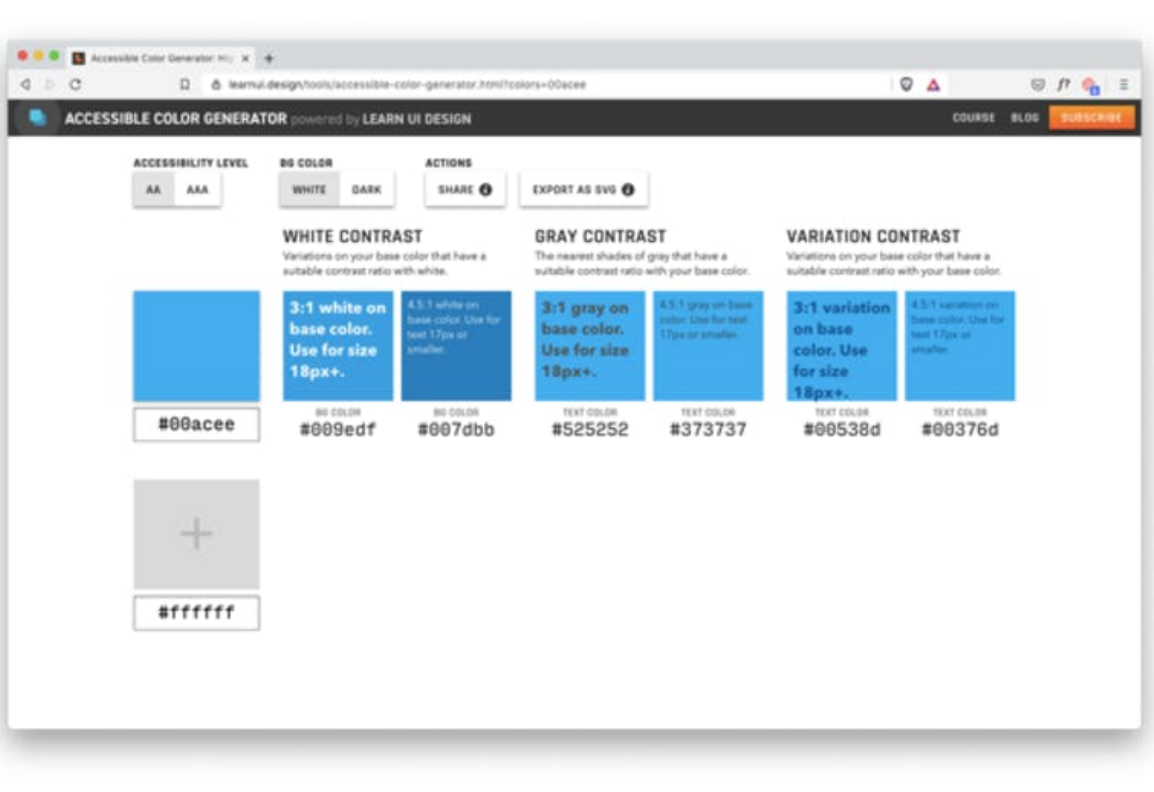
All you have to do is select 2 layers and hit cmd+p (the Stark Sketch shortcut), and a handy pop-up appears. So, for that Twitter button...



Uh oh! It should be clear from all those red X's that white-on-twitter-blue doesn't even pass the most lenient contrast requirements – that of *large* text at *AA*-level.

The slightly more complicated question is: *now what*? Stark doesn't tell you what to adjust your button color *to* to meet AA.

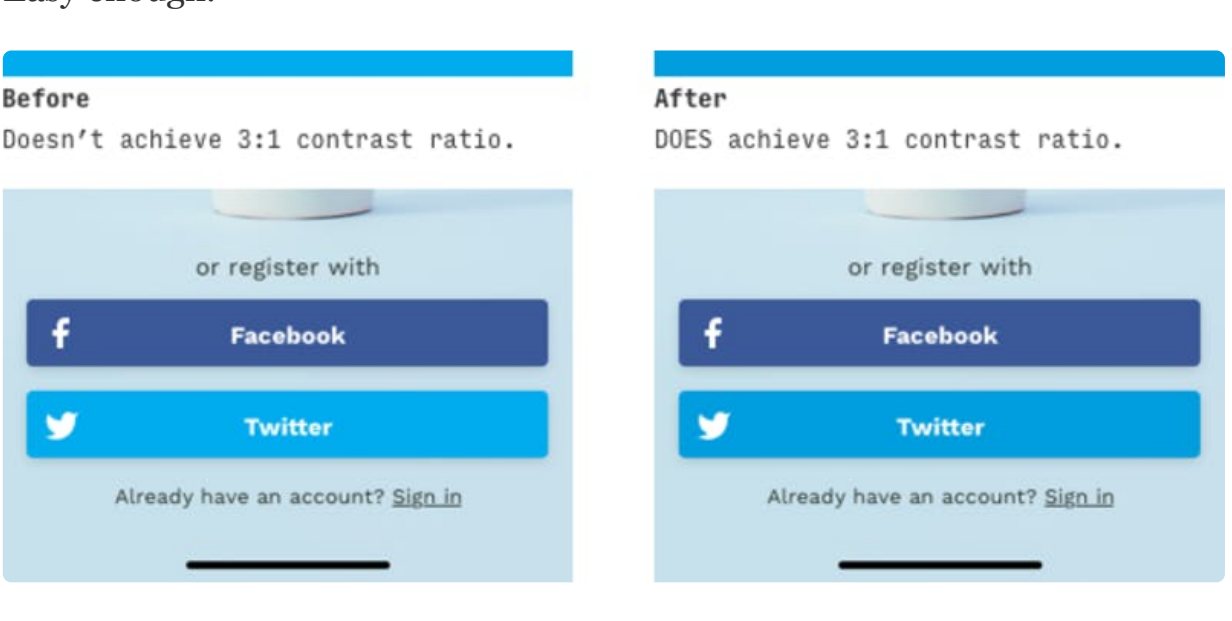
That's when the free Learn UI Design Accessible Color Generator tool that I've created comes in handy!



If I enter the button **background color** (#00acee) into the box, it tells me ***the closest variations of that color*** that meet contrast guidelines.

In this case, I want contrast against white, and 3:1 ratio, so I should use... **#009edf** *instead*.

Easy enough.



Wow. OK. Not so big a change, eh?

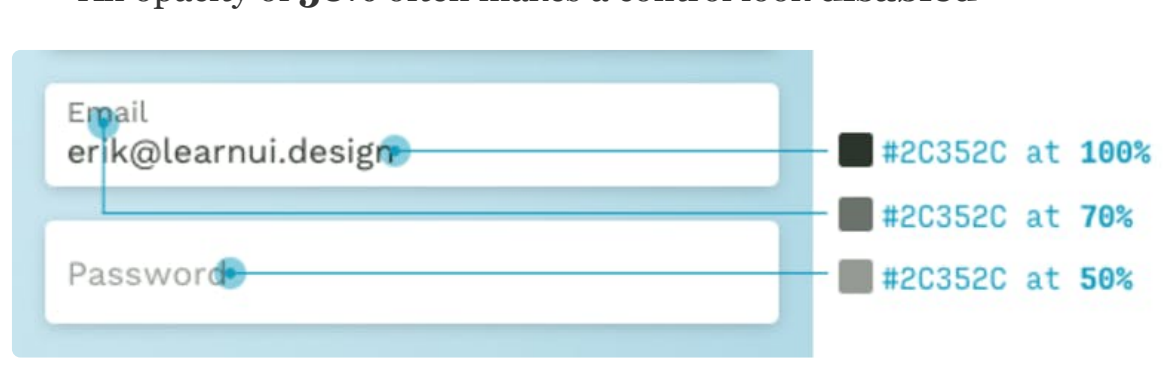
It's a real example; this stuff happens 🤷‍♂️

Fixing Color Contrast with Opacity

Let's move our sites towards a **frequent culprit** of inaccessible text contrast – **the textbox hint text!**

I'm a huge proponent of using opacities of various controls to represent different states/variations:

- An opacity of **70-80%** makes text look "**secondary**"
- An opacity of **50%** often makes a control look **disabled**



Now, while **disabled text** doesn't need to meet any contrast guidelines (per WCAG), translucent labels and field hints **often fall below 4.5:1 contrast**.

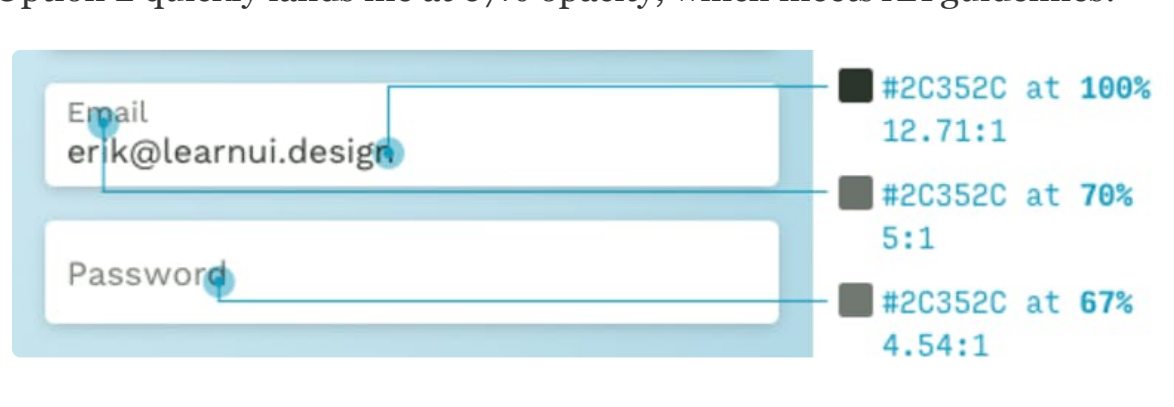
Neither Stark nor the Accessible Color Generator handles opacities of colors. Instead, you have to use the color picker to grab the *color as displayed*.

If I plug in the lightest shade (the "Password" color) to the Accessible Color Generator, I get a variation I need to make:

Since this is regular text, we want the 4.5:1 variation – #727772. We have two ways of getting that:

1. Setting the label color directly to #727772
2. Adjusting the opacity up from 50% to whatever makes it *look* like #727772

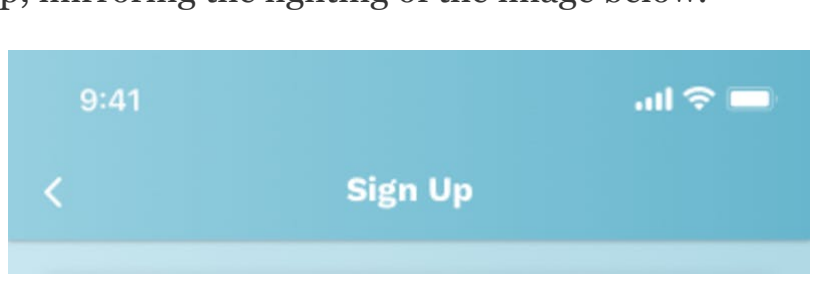
Option 2 quickly lands me at 67% opacity, which meets AA guidelines!



Nice!

Fixing Color Contrast with Manual Variations

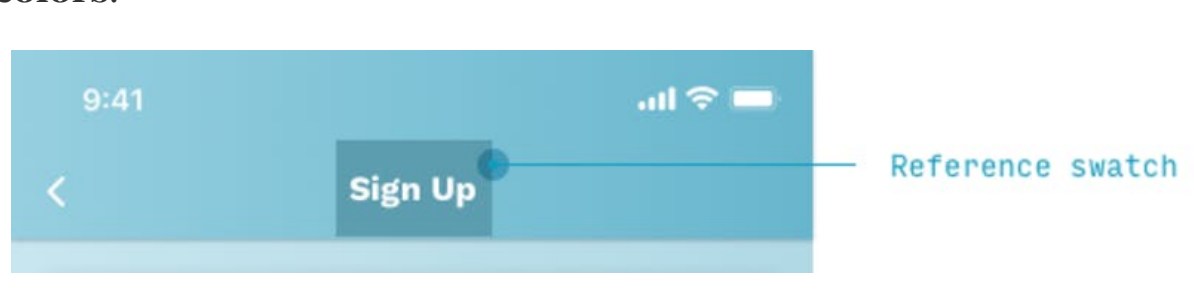
Now for the tough part. In the header, I have a gentle gradient of colors across the top, mirroring the lighting of the image below.



To make sure this text is appropriately contrasting, I will **sample a color at the left (lighter) side** – just to the left of the "S" – and plug it into the Accessible Color Generator.

Looks like I need to make some adjustments, even for 3:1 contrast!

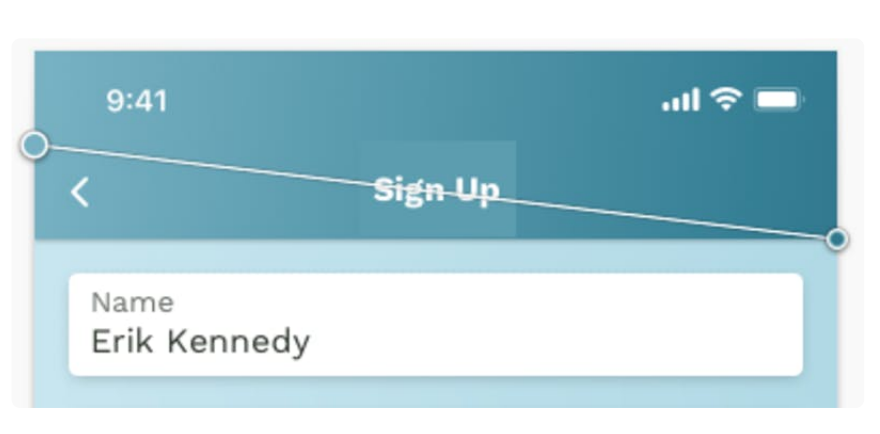
But since the gradient is based on endpoint colors, and I can't adjust the middle directly, I will **add a reference swatch** and **hand-adjust the colors**.



I've written about how adjusting colors is the **most important color skill** in UI design. In the linked article, I explain how, to adjust towards **darker** variations on a color, you want to...

- Increase saturation
- Decrease brightness
- Optionally shift the hue towards 240°

(Seriously, read Color in UI Design: A Practical Framework, if you haven't already – and start with this HSB primer if you don't know what "saturation" means 😁)

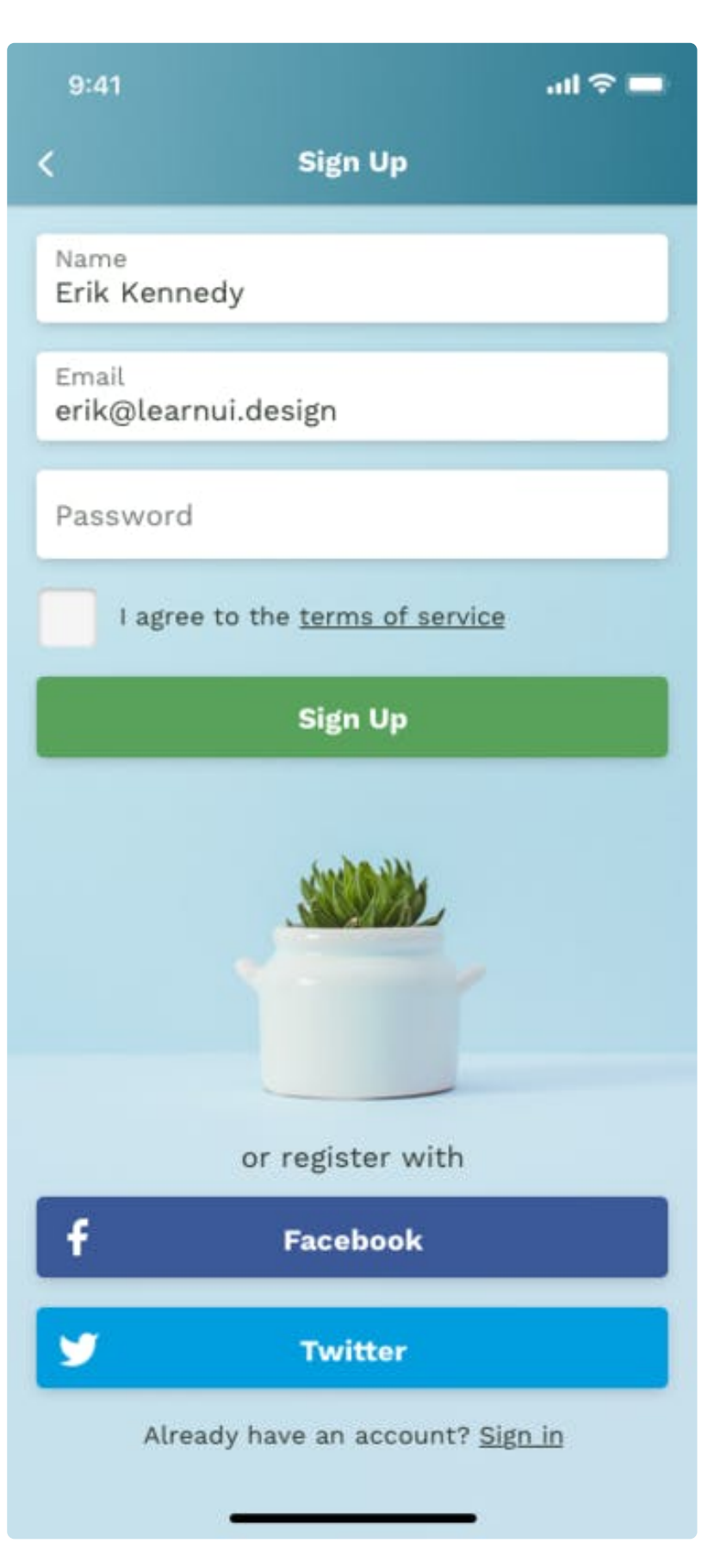


My reference swatch shows that we're at least as dark as we need to be now! But let's double-check just in case.

Success!

The final design

Here's our finished product:



Maybe you've heard of accessibility getting a bad rap ("It forces me to use colors I don't want to! It's impossible to work with!") – but you can see here, the **changes needed** to meet official WCAG guidelines **were all quite small**.
And now, you've got three techniques for coming up with accessible color pairings:

1. Adjusting with Stark and the Accessible Color Generator
2. Adjusting opacity
3. Adjusting colors by hand

This is a skill that SO MANY **government, educational, and large user-base clients** need for their websites... and it's only getting more common.

And Accessibility is just one of the many lessons in Learn UI Design, which opens tomorrow for new enrollments.

Learn UI Design can help you **get a job**...

> Hey Erik, I've got a success story – I landed a **dream job** in a **world-class design agency**. Learn UI Design helped make it happen. Thanks again for the valuable course – I return to the videos often."

– BILL BARHAM
PRODUCT DESIGNER

... or just **understand the "why"** behind visual design – so you can apply it to your *current* job.

> Learn UI Design is the ONLY course I've found that provides the reasoning *why* this design choice is better than another... I feel like I owe you a personal thank you for taking the time to provide me with the *why* behind the design decisions... The course is worth every single penny."

– LAURA STRADER
UX DESIGNER, BLUEHOST

And, best of all, all students have **LIFETIME ACCESS**. Once you're in, *you're in for life* – and can see all future updates and additions I make in the course.

> Learn UI Design was 100% **a great investment**. I've been working on an iOS app and website, and I can't tell you how many times I referred to your video lessons (**especially the typography ones**). Totally worthwhile."

– WEMBLEY LEACH
SOFTWARE ENGINEER, SMITHSONIAN CHANNEL

Stay tuned. Learn UI Design opens at midnight tonight – and will be open for one week only 😎

Cheers,
Erik D. Kennedy