The Better Email

On Accessibility

The Better Email On Accessibility

A practical guide to creating accessible email marketing campaigns, from copy to code.

To learn more about email marketing, **visit thebetter.email online**.

Table of Contents

Introduction

What do you think of when you hear the word **accessibility**?

There's a good chance you're picturing wheelchair ramps or handicap parking spaces. There's an even better chance that email isn't anywhere near your thoughts right now. Well, maybe… but that's only because you're reading this guide.

When most people think about accessibility, it's usually in terms of physical affordances for people with differing abilities. It's the wheelchair ramps, parking spaces, brail menus at restaurants, or closer theater seats for people with low vision. For most people, these affordances are for others. They would hardly consider themselves disabled, so accessibility is—at best—usually an afterthought.

Those affordances help plenty of people who think of themselves as able-bodied, too. A curb ramp on a street doesn't just help people in wheelchairs, it helps skateboarders or kids riding bikes, someone trying to get a refrigerator into their first apartment, and parents pushing strollers, hoping to get out of a busy street before disaster strikes. Accessibility—when done right—is about creating better experiences for everyone.

But accessibility is more than making the physical spaces we collectively inhabit better for people with different abilities. It's about making the digital spaces we inhabit better, too. And that includes email.

Chapter 1

What is accessibility?

Before we start digging into making accessible emails, we need to define what accessibility actually is: both in regards to email and the wider world.

There are a lot of definitions for accessibility worth considering. Merriam Webster offers a few options, like:

> *Capable of being used or seen.*

Or, perhaps more relevant to us:

> *Easily used or accessed by people with disabilities.*

But, remember: Accessibility isn't just about people with disabilities. It helps everyone.

The Mozilla Developer Network gets closer to what we need by focusing on website:

> *Accessibility is the practice of making your websites usable by as many people as possible.*

If you're working on behalf of the United States Government, they're quite clear that, "Accessibility is the law."

For email, the one that I used to favor—and which was included in The Ultimate Guide to Email Accessibility I wrote while at Litmus—was:

> *When an email's content is available to—and its functionality can be operated by—anyone, regardless of ability.*

But I think it was a bit naive, since no one can really make an email (or web page, or app, etc.) that can be used successfully by anyone. You will always encounter hurdles, no matter how hard you try.

All of the above definitions talk about functionality and access. It is "access"ability, after all. But I think that we need to focus on what people need to get out of a web page, application, or email beyond mere functionality. What people are really looking to get is some **benefit** from using a piece of technology like an email. Yes, they need to be able to access and act on information in an email, but, more importantly, they need to gain some benefit from it, too. So, at least in my opinion, accessibility goes beyond mere functionality (typically driven by technological solutions like those described in Chapter 5), and needs to take into consideration content strategy, copywriting, visual design, and inclusion, too. Accessibility is a holistic approach to creating emails that are beneficial for your subscribers—disabled or not.

With all of that being said, the definition of accessibility I favor—at least in regards to email—is:

Accessible emails allow subscribers to access and use the information in an email and benefit from their inbox.

I think this definition speaks to our goals more clearly. Sure, we need to make something that's functional and operational, but what we're really trying to do when focusing on accessibility is creating something from which more people can benefit.

Beyond that, there are some more concepts we should define and talk about up front.

The first is **inclusion**. Inclusion is the idea that we want to welcome and include everyone into what we build, regardless of who they are, where they're from, or anything else like their race, religion, gender, sexual orientation, economic status, or any other characteristic humans usually use to classify or discriminate against others. Inclusion goes hand-in-hand with accessibility, but I usually think of them as tactics vs. strategy. Accessibility tends to be more about the tactical work that needs to happen to make an email beneficial for users, while inclusion helps define the overall content and design strategy to ensure that you're not inadvertently making any of your users feel excluded by the content, visuals, or strategy powering your campaigns.

I'll bring up inclusion throughout the book, though, especially when it comes to copywriting and the images used in an email, but just keep in mind that it's a little different from accessibility. Still, two sides of the same coin.

The second is **universal design**. Universal design is a strategy for designing things that are usable by everyone without the need for special design or adaptation. In her excellent book, Accessibility for Everyone, Laura Kalbag uses the wheelchair ramp example to illustrate universal design. While accessibility would involve adding a wheelchair ramp to a building with existing steps, universal design would involve building a structure with a combined ramp and steps, "opening access to all and forcing no one to go out of their way to choose one option or the other."

In the context of the web or email, it's the difference between designing something with accessibility and universal usability in mind from the outset rather than tacking on accessibility features after the fact. This book takes a universal design approach to accessibility and you should, too. Bake usability in from the start and you'll have to hack less in the long run to make your emails work for everyone.

The third concept worth looking at is **POUR**. POUR is an acronym for: Perceivable, Operable, Understandable, and Robust. A part of the World Wide Web Consortium's (W3C) Web Accessibility Initiative, it's a good way to remember the steps we need to take to keep web content (and emails) accessible. Emails need to be perceivable for both sighted and non-sighted users, operable for a wide range of users with different abilities, understandable for audiences with different educational or life experiences, and robust enough to work across different email clients, devices, and network conditions. It's another way of saying access, use, and benefit.

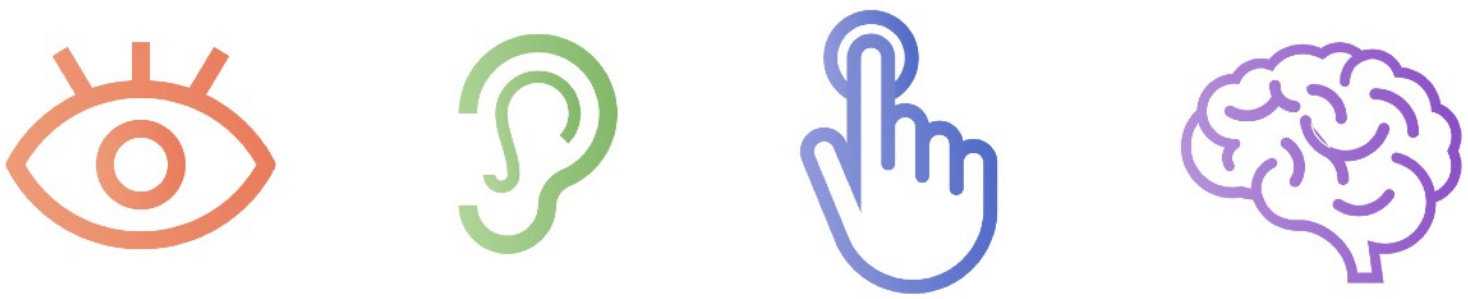
Finally, you'll likely come across the term **a11y**, which is shorthand for accessibility. It looks like the word "ally", which is what we're all trying to be to our subscribers, and the 11 stands for the eleven letters between the A and the Y in the word "accessibility." You'll often find it used as a hashtag in accessibility discussions on Twitter or LinkedIn, or emblazoned on t-shirts at conferences. Worth knowing as you dig deeper into what wonderful world of accessibility.

The Shape of Disability Today

Since accessibility is about designing experiences that work for everyone, it's important to understand the wide range of abilities, and disabilities that people experience around the world—including in your subscriber base.

While most people are fortunate to consider themselves able-bodied and without any cognitive impairments, it's dangerous to mistake that for everyone's situation. Disability is far more prevalent than you'd initially expect. **According to the CDC**, about one in four adults in the United States alone have some type of disability. These disabilities range in cause, severity, and longevity, but generally fall into one of four categories:

- **Visual**: Anything having to do with the eyes, including low-vision, blindness, and color blindness.
- **Auditory**: Anything having to do with the ears, including partial and complete deafness.
- **Motor**: Anything having to do with how the body moves, including limited motor control, tremors, missing appendages, and more.
- **Cognitive**: Anything having to do with how the brain operates, including learning disabilities, memory problems, and distractibility.



A lot of email accessibility tactics are geared towards solving for visual impairments, which makes sense since it's one of the most common types of disabilities affecting people. And the population affected is only growing, with **the CDC predicting** that population to double to more than 25 million people in the US by 2050. While a lot of that growth is driven by aging populations, which **the United Nations describes** as one of the most important global issues, it's also a side effect of unequal access to healthcare and resources around the world.

Visual impairment isn't the only disability email marketers have to worry about, though. While auditory disabilities are less of a concern due to the nature of email, many subscribers deal with motor and cognitive disabilities. Users (again, especially as a result of aging) having wildly differing motor abilities. Some people can use a mouse or scroll on a mobile device just fine, while others lack the fine motor control, fingers, or limbs, making scrolling frustrating or downright impossible. And people's cognitive abilities place limits on how easily they can consume and benefit from the information in email campaigns. Unequal access to education creates disparities between users, and learning disabilities—like dyslexia—make it hard for subscribers to read and understand emails. Learning disabilities are relatively commonplace, too. The Dyslexia Center of Utah estimates that 15-20% of the population suffers from a learning disability, with dyslexia being the most common.

Combined, visual, motor, and cognitive impairments have a huge affect on how we need to strategize, write, design, code, and test our email campaigns.

Assistive Technology

Many people with disabilities rely on tools, devices, and software to help them throughout their days. Collectively, these tools, devices, and software are called assistive technology.

Two common examples of assistive technology are hearing aids and eye glasses. Hearing aids are designed to amplify sound so that people with hearing loss can better sense and interact with the world around them. Eye glasses do the same for people with visual issues. Glasses and contacts are designed to correct things near- and farsightedness so that people can clearly see. Both are common enough for most of us to take them for granted.

Although assistive technology can get extremely specialized and personalized for the individual, most fits into one of four categories designed to help the four major categories of disabilities we just discussed: Visual, auditory, mobility, and cognitive technology.

- Visual assistive technology includes glasses, contacts, braille literature and displays, hardware display magnifiers, software screen magnification tools, screen reader software, and more.

- Auditory assistive technology includes hearing aids, assistive listening devices, amplified phones, captioning and transcripts, and more.
- Mobility assistive technology includes wheelchairs and walkers, transfer devices, prosthetics and exoskeletons, and adaptive seating and positioning devices.
- Cognitive assistive technology can include memory aids, specialized education software, specialized reading software, and even specific fonts.

Due to email's digital nature, email marketers typically only need to focus on a small subset of assistive technology that can access the inbox. For the most part, when it comes to designing and testing for assistive technology, we'll focus on the use of screen reader software and magnification tools like the zoom features included in most operating systems, devices, and browsers.

Screen reader software is exactly what it sounds like: Software that takes the content on a screen and reads it out loud to a user. Again, it's designed for users who have a hard time visually perceiving content on a screen, whether that's due to temporary or permanent low-vision of blindness, or situational issues.

There are a lot of options on the market for screen readers, some free and some prohibitively expensive. Fortunately, most operating system manufacturers have made huge strides towards accessibility in the last decade and include screen reader software preinstalled. Here are the most common:

- Apple created **VoiceOver** for both Mac and iOS devices.
- Microsoft uses **Narrator** on Windows PCs.
- Android features **Talkback** on a wide range of devices.

Beyond those, there are plenty of free and open source options as well as commercial software. For Chrome and Chrome OS users, ChromeVox is one option. For Linux users, Orca can be installed. And for Windows users, two popular options include NVDA (which is free) and JAWS (which costs money).

Regardless of the tool used, each of them looks at the content on the screen and reads it to the user. All of them do so by looking at the underlying code in which the content is marked up. Developers understand (at least good ones do) how different markup means different things to the screen reader software, impacting how the

content is read to the user, and they use appropriate markup to make sure the content can be understood as clearly as possible.

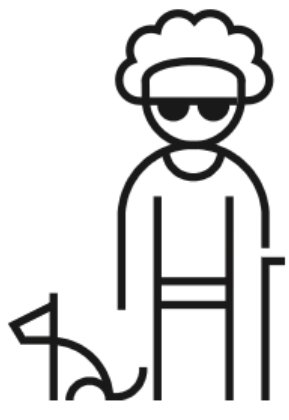
We'll see how the markup we use affects accessibility in Chapter 4. For now, just be aware that assistive technology has real implications on how we design and code our email campaigns.

Ability is a Spectrum

It can be tempting to restrict disabilities into those four categories, but, in reality, human abilities live on a spectrum. The categories above are largely longterm or permanent disabilities, ones that are attributes of specific humans. But some of those disabilities aren't an issue until a person is forced to interact with a society that's not set up to account for those disabilities. In this way, disabilities are often not a personal attribute so much as a result of someones context in a specific time or place. Their ability to benefit from something is largely outside of their control.

Permanent disabilities aren't the only thing that affect people, though. Even if you don't have one of the disabilities listed above, you are only temporarily able-bodied, just like everyone else on Earth. There are plenty of temporary disabilities that can affect how you interact with the world.

There's a great graphic from **Microsoft's Inclusive Design Toolkit** that illustrates this idea. In it, you can see how temporary disabilities like a broken arm can impact how we interact with our devices, as can situational disabilities like a new parent holding their child while trying to do a million other things. Cataracts can temporarily impact someone's vision, as can dilating someone's eyes during an eye exam. And the many, many distractions we encounter throughout the day can hinder our ability to interact with our devices and emails.

	Permanent	Temporary	Situational
Touch	 One arm	 Arm injury	 New parent
See	 Blind	 Cataract	 Distracted driver
Hear	 Deaf	 Ear infection	 Bartender
Speak	 Non-verbal	 Laryngitis	 Heavy accent

That's why we need to think about accessibility as a spectrum. Every single one of us will experience a disability at some point in our lives, whether it's permanent, temporary, or situational. And that's why we want to take a universal design approach to accessibility and build emails that can be accessed, used, and benefited from for everyone—because we'll need those affordances, too.

The Subscriber Value of Accessibility

The subscriber value of accessibility should be clear: Accessibility makes it easier for everyone on your list to access, use, and benefit from your email campaigns. It doesn't matter if it's a monthly newsletter, the twenty emails you send on Black Friday, or a password reset email that's sent when someone gets locked out of their account—you need to make sure each one of those emails works for your subscribers regardless of their permanent, temporary, or situational abilities.

Ignoring accessibility is telling your subscribers that you don't value their time, attention, and business.

When it comes to accessibility in email, a lot of tactics involve the visual design of an email campaign. And, whether or not you want to admit it—looks aren't everything and all—how an email looks absolutely impacts how subscribers view you as a sender and your brand as a trustworthy companion.

The visual design is what people encounter before they actually read an email, and studies have shown that you don't have a lot of time to make a good first impression. An older study from the Taylor & Francis Group showed that people form opinions about web designs in **50 milliseconds** and that those opinions—based on a quick visual judgement—influence the subsequent enjoyment, or lack thereof, that users experience. A Stanford study revealed that 46.1% of people base the credibility of a company on a website's (re: email's) visual design. What's more, people typically spend very little time on any one email. According to Litmus, people spend an average of 13 seconds on an email—meaning scannability, hierarchy, and visual design are paramount.

Accessibility directly impacts the visual design in emails, so it's on us to make sure that we create experiences that look good and work well for our subscribers so that they can get the most value possible out of our emails.

The Business Value of Accessibility

The business value of accessibility should be clear: There are a lot of people with different abilities and focusing on accessibility provides the best experience for all of them. And, when you create better experiences for your users, they tend to place more trust in your brand—creating longer term and more valuable relationships.

Simply put: Focusing on accessibility creates more business opportunities.

As we've already seen, a large portion of the population has some sort of disability. According to The World Bank, it's around 15% of the world. And that 15% accounts for around **$7 trillion in annual disposable income**. When you create accessible experiences, it's easier to tap into that extended market and make more money. Businesses like money and people like better experiences, so it's a win-win situation.

Beyond pure capitalism, focusing on accessibility reduces legal risk for brands, too. As more of our everyday life is tied to the digital world, it's increasingly important to ensure that access is available to all of your users—or risk legal ramifications.

A number of laws exist around the world that dictate accessibility requirements:

- The Americans with Disabilities Act (ADA), Section 508, and the Twenty-First Century Communications and Video Accessibility Act in the United States.
- The Accessibility for Ontarians with Disabilities Act in Canada.
- The Disability Discrimination Act and Equality Act in the United Kingdom.
- The European Accessibility Act in the EU.
- The Act on Equal Opportunities for Disabled Persons in Germany.
- The Stanca Act in Italy.
- The Disability Discrimination Act in Australia.
- The Japanese Industrial Standard in Japan.
- The Convention on the Rights of People with Disabilities (CRPD), which was ratified by the United Nations and affects 175 countries.

- And many, many more.

Although many laws are focused on government websites and digital content, legal action has affected the private sector. **The Wall Street Journal reported** in 2021 that accessibility-related lawsuits were on the rise (at least in the US), from 2,300 in 2018 to 3,500 in 2020, with an expected increase to around 4,000 in 2021. While big-name brands like Target, HR Block, and Dominos Pizza have been in the news for running afoul of accessibility rights, two thirds of lawsuits were aimed at smaller companies with less than $50 million in revenue.

It should be noted that many of the laws above reference or look to the **Web Content Accessibility Guidelines (WCAG)** to determine how to create accessible digital experiences. The WCAG guidelines are an international standard that outline principles for making content perceivable, operable, understandable, and robust—all of the things we're aiming to do in email. Although the principles aren't specific to email, they should be seen as the go-to reference for creating accessible experiences. Many of the strategies outlined in this guide take their cues from the WCAG docs, which will be referenced throughout.

Creating accessible emails will allow you to tap into a larger market and actually keep the money you make by avoiding expensive lawsuits.

Overview of Topics

OK, so we've discussed what accessibility means in a broader sense as well as why it's important for subscribers and businesses alike. But how is it applied to your own email campaigns? And what will you learn in the rest of this guide?

Let's take a look…

In Chapter 2, you'll learn about how to write accessible copy for email campaigns, including tips on using plain and inclusive language, readability, and how to write effective alternative text for images.

In Chapter 3, you'll see how visual design impacts accessibility and learn how to use design principles like hierarchy, color, contrast, spacing, layout, and typography to create aesthetically pleasing—but still accessible—email campaigns.

In Chapter 4, you'll learn about the code and technical considerations that drive accessibility in email, including how to use semantic markup, ARIA, and HTML and CSS to keep emails accessible, especially for folks using assistive technology.

In Chapter 5, you'll be exposed to the many tools and processes out there that allow you to test your emails for accessibility, ensuring that you're providing your subscribers with the best experiences possible.

We'll wrap up with plenty of links to additional resources to help you out on your accessibility journey.

On thing to keep in mind throughout this guide is that accessibility is just that: A journey. It's not a one-and-done task. Email accessibility is an ongoing process that needs to be reviewed and evaluated on a regular basis, just like anything else important to your business. While many of the tactics we use to keep emails accessible can be baked into the code powering campaigns, you still need to be thoughtful throughout the email production process—from strategy to copywriting, coding to reporting, and everything in between.

It's also helpful to remember that—try as we might—there's not such thing as a perfectly accessible email. The sheer range of human abilities and experiences means that there's no way to account for every single situation in the same email. There will always be some aspect that's less than ideal for someone.

But the fact that you're dedicating time and energy to learning about email accessibility and implementing the strategies in this guide in your own emails means that you're already on the path to making better emails for your subscribers.

Chapter 2

Copywriting for Accessibility

Although a lot of the work in making accessible emails comes in while designing and coding a campaign, your overall strategy and the copy you use to convey it plays an important part in creating perceivable, operable, and understandable experiences. That's why it's important to focus on accessibility right from the start of a campaign, before you open Photoshop, Figma, or your favorite code editor.

But what considerations are needed at the planning and copywriting stages to make sure everyone can understand and benefit from a campaign? Let's take a look.

Have a Clear Goal

Every email you send should have a clear goal attached to it. While that's just good marketing advice in general, having a specific goal for an email allows you to create more streamlined, simple, understandable emails—in other words, more accessible email campaigns. The goals of your campaign will, in a very real sense, dictate how you write an email, from headlines and calls-to-action all the way to privacy information, unsubscribe language, and stuff in the footer, not to mention how you design and code the email as well.

So it's important to understand why you're sending every single email.

A non-exhaustive list of goals might include:

- Resetting a password
- Thought leadership
- Encouraging webinar registrations
- Sending shipping information

- Pushing a limited-time sale
- Updating your privacy policy
- Providing product recommendations
- Trying to re-engage lapsed subscribers
- Welcoming new subscribers
- Notifying users of comments on a doc
- Getting user feedback
- Encouraging a service upgrade
- Wishing someone a happy birthday

Or many, many more. But it's important to clearly define and state that goal from the outset—before you write a single word of copy. This will allow you to focus ruthlessly on that goal while you're writing, editing, and building an email.

It's especially important when you're working with multiple stakeholders, copywriters, or designers, too. Keeping everyone on the same page will lessen the likelihood of confusing emails and confused subscribers.

My recommendation is to start with an email brief document of some sort that can collect all of the information that goes into a campaign—including the overall goal—and act as a reference for everyone that touches the email while it's being produced.

Here's an example email brief that can be used for any campaign. You'll notice that the goal of the campaign is one of the first things defined, since it dictates everything that comes after. I prefer to use a goal keyword—like "thought leadership" or "conversions"—and then a quick summary below to clarify what that means for everyone in one or two sentences.

Use this document as a reference when you're actually writing the copy for your campaign. Now, when it comes to writing…

Use Plain Language

One of the most important things you can do in your copy is to use simple, plain language. That sounds easy enough, but what is simple, plain language?

The World Wide Web Consortium (W3C) defines it as language that is clear and concise. Getting more specific, it's language that is easy for people with different educational backgrounds and cognitive abilities to understand, uses limited jargon and acronyms, gets to the point, and is inclusive.

A good way to understand plain language is by looking at an example.

Here's some copy from a real email in my inbox advertising a podcast:

Hear from doctors that are knowledgeable, relatable and bringing you the information you need to live a smarter, healthier life.

It's a mouthful when read out loud. Here's something a bit better:

Listen to real doctors and learn how to live a smarter, healthier life.

By using the word "real," we can replace the wordier, more awkward "knowledgeable, relatable" while still getting at the same meaning. Real doctors are (at least in theory) knowledgeable since they've gone to medical school, practiced, and accumulated a lot of wisdom on health matters. And if they're real people, they'll hopefully be relatable. You're real people, too, right? The "bringing you the information you need" bit is very passive, too. By simply replacing that with "listen," the subscriber becomes a more active participant in the process.

Bonus points for being a lot shorter, too. Brevity is always helpful.

Let's go back to our specific definition of simple, plain language and break it down into its components:

- Understandable and readable
- Free of jargon and acronyms
- Straight to the point
- Inclusive

Write Readable Copy

You've probably heard the term "readability" before. While there are different definitions and criteria for determining readability, it's essentially the measure of how easy a piece of text is to read. Our goal for writing accessible copy is to write readable copy, because readable copy is understandable copy, and understandable copy is a benefit to our subscribers.

Readability can be determined a few ways, but one of the most popular ways is using the Flesch Reading Ease scale, which rates writing on a scale from 0-100. The higher the score—which is calculated from the sentence and word lengths—the easier the writing is to understand. Generally speaking, a score of around 60-70 is defined as plain English that most people 13 years old or above can understand—including your subscribers.

According to TextCompare.org, the paragraph above scores a 59.47, which means it could be a bit difficult for some people to understand.

While it's tempting to put all your copy through a readability test (made easier by tools like Grammarly), the main thing to focus on is what is used to determine that score: sentence length and word length.

Longer sentences, paragraphs, and words are harder to understand. So:

Use shorter sentences, paragraphs, and words.

This aligns with the W3C's guidelines, which clearly say to, "Write in short, clear sentences and paragraphs" and "Avoid using unnecessarily complex words and phrases." Beyond just shortening words and phrases, you can use things like bullet points to break up copy and keep it readable. Or break up what could be longer paragraphs into smaller sections. The online music retailer Reverb does this well in their new subscriber welcome email:

Here Are Five Places to Get Started.

GET THE APP

Buy, sell, make offers, and access special sales and discounts straight from your phone.

Download Now

MAKE REVERB YOURS

Customize your Reverb experience so it features the gear you care about most.

Set Up Your Feed

SEE WHAT WE RECOMMEND

Our resident gearheads are always on the hunt for the best deals and rarest finds on Reverb.

Shop Now

While they could easily have lumped all of those features into one long paragraph, they highlighted each feature individually and made an easier to understand, more accessible (and probably higher converting) experience for their subscribers.

Shorter copy has the added benefit of being more scannable, too, which is important considering people's short attention spans. **According to Litmus' 2021 State of Email Engagement Report**, the average time spent in an email is **only 13 seconds**. That's not a lot of time to engage a subscriber…

Writing shorter, more understandable copy is difficult, though. But it can be made easier by following a few handy rules:

Choose shorter words: Longer words are tempting to use since they make the writer seem smarter (and who doesn't want to be seen as smart?), but they not only add to copy length but serve to confuse people who may not know what the word means. Opt for the shorter, more common words when possible. A good example is picking "use" over "utilize."

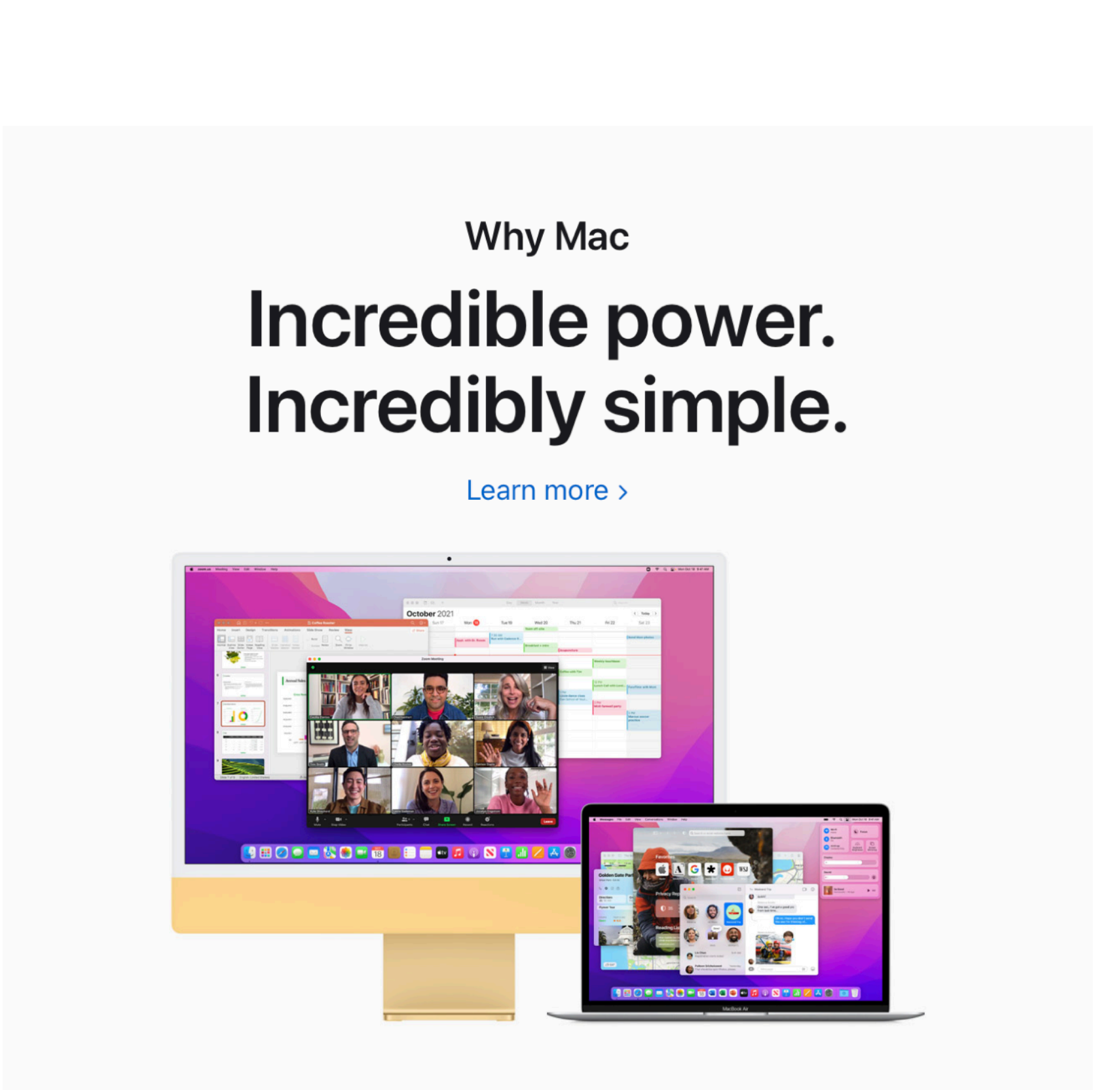
Remove unnecessary qualifiers: For example, instead of saying, "Creating effective email campaigns is really hard," you could say, "Creating effective email campaigns is difficult." Or, "Our cookies are very tasty," could be, "Our cookies are delectable."

Remove redundant information: For example, including words that mean the same thing—like also and too—in the same sentence.

Get rid of cliché phrases: There's almost always a single word that conveys the same thing as a longer, overused phrase. Pick the word instead.

Avoid figures of speech and complicated metaphors: Again, they can make you sound smart, but usually tend to confuse people and lengthen copy.

Make a game of it: Challenge yourself to see how few words or sentences you can use in an email (while still conveying the necessary information, of course). Apple does this really well in their marketing copy. For example:



Why say, “The most powerful, but still user friendly, computers around,” when you can simply say: “Incredible power. Incredibly simple.”? It’s a lot catchier, shorter, and easier to read while still conveying the same information. Do that for your copy.

Read your copy out loud: This is one of the best ways to identify copy that can be improved. Literally read your writing out loud to yourself and note where it feels awkward or hard to get out. Those are the areas where copy can be tightened up and improved for your subscribers.

Avoid Jargon and Acronyms

As professionals, it’s easy to get tunnel vision. We’re in the weeds every day, talking to colleagues and users who are intimately familiar with our respective fields,

companies, and products. As such, we're liable to throw around jargon and acronyms without thinking twice about whether anyone else understands them.

Email marketing is packed with examples. AMPscript, BIMI, blocklist, CASL, conversion rate, DKIM, DMARC, funnel, martech, CTA, WYSIWYG, CTOR… the list goes on. Hell, I even created **a glossary** to try to track and explain them all. But you'd be hard-pressed to find someone who understands them all, let alone a group of people on the same email list that does. And when we throw around jargon or acronyms without care, we risk alienating people that don't understand them.

That's why it's important to avoid jargon (the technical terms used by an in group) and acronyms (a word or abbreviation formed by letters from other words) as much as possible when writing accessible copy.

That being said, it can't always be avoided. If you do need to include jargon, acronyms, or other things like product or feature names, you should define them on first use.

When I first mentioned the W3C, it was in parentheses after what that acronym actually stood for: The World Wide Web Consortium. Use the format:

Spelled Out Acronym (SOA)

For jargon, you can likewise use parentheses or the surrounding copy to either define that jargon (like I did above for the term jargon itself, so meta) or provide additional contextual clues as to what that jargon means.

For things like product or feature names, link out to a landing page of some sort that provides additional information. This will not only help you to explain confusing terminology to subscribers, but could provide additional clicks and conversions, too.

Use Inclusive Language

Beyond writing shorter, more understandable copy, you should strive to make your writing inclusive, too. Inclusive writing is using language that avoids offending or excluding people based on things like their race, religion, gender, sexual orientation, economic status, or any other characteristic humans usually use to classify or discriminate against others. Remember, we're aiming for emails that everyone can access, use, and benefit from. And the language we choose directly impacts that.

As a baseline, you should avoid language that is racist, sexist, or in any other way prejudiced against any group of people. While some obvious words and phrases come to mind that should be avoided, inclusive language can be more nuanced, too. Especially when it comes to personalization in email.

Over the last few years, there has been a bigger push to use people's preferred pronouns. There are **a lot of reasons why it's important** in the grand scheme of things, but for us email folks, it boils down to respecting our subscribers.

If you collect your subscriber's pronouns—either at signup, through a preference center, or your product—you should sync that data to your ESP and make sure they're used properly when personalizing campaigns. And, if you're using fallback content, try to stick to gender-neutral pronouns like: They, them, theirs, and yours. Additionally, you should probably cut out using honorifics like Mr., Mrs., and Ms., as most imply a gender or relationship status that may no longer apply.

Beyond that, inclusive language can also mean properly translating and localizing content depending on your audience.

Translating content usually means using a service (often automated) that takes input in the form of one language and outputs the same content in a different language. If you have significant subscriber bases in countries other than your own, it can be beneficial to translate your emails into their languages to ensure that they can more easily understand your content.

Localizing content takes translation a step further and makes sure that you're taking into account not only the grammar and syntax of a language, but the cultural norms of a specific area, too. It usually involves a human touch as opposed to being pure automation, but can mean the difference between success and disaster.

One of my favorite examples of translation vs. localization is when Pepsi entered the Chinese market back in the 1960s and opted to translate their slogan, "Come Alive With the Pepsi Generation." If they'd done the work to put that slogan into context, they wouldn't have ended up with the straight translation of, "Pepsi Brings Your Relatives Back From the Dead."

As a general rule, automated translation works well for shorter, more informative pieces of content—think of disclaimers and privacy policies in the footers of emails. But headlines, body copy, calls-to-action, and any of the flashier marketing-type copy

or larger campaigns probably need a good review from an actual native speaker to make sure you don't run into embarrassing or potentially costly problems.

On the flip side of translation and localization: If you're not translating content, be thoughtful about what phrases you use when you have potential non-native speakers in your audience. Even everyday phrases for English speakers can be confusing to people who know English as a second language. Heck, there are even disparities between American audiences and other English-speaking countries, or even within parts of the same country. I'm from Michigan and grew up referring to soda as pop. I've gone down to the south and asked someone what kind of pop they had only to be met with a confused look.

It all comes down to being thoughtful about what words you use in your email, and understanding how those words could impact someone's ability to access, use, and benefit from your emails.

Write Descriptive Links

Remember how every good campaign starts with a clear goal? That goal is almost never to keep people in the email campaign itself. You usually want to link out to a landing page, news article, app, or something else, and you use hyperlinks to do so.

We'll discuss better practices for visually designing links in the next chapter, but copywriting is an important consideration for links.

The most accessible links are ones that are descriptive, meaning the copy describes the content of the linked page or what will happen when you act on the link.

You've probably seen plenty of bad, non-descriptive links in email. Think of the "Click Here" or "Learn More" CTAs that plague inboxes. Sure, both kind of tell the user what to do (click) or what will happen (they'll learn), but they still have issues. First, not everyone can actually click—people tap on mobile devices or use blinks with the help of assistive technology to navigate online. Secondly, what will you be learning more about? Without any context, that link means nothing. And since people largely scan emails instead of reading from top to bottom, that context is probably lost when someone's scanning an email in the limited time they have it open.

A better approach is to link key pieces of information in a sentence or, in the case of standalone CTAs like buttons, describe the benefit of activating that link.

An bad example of linked text would be:

The Better Email On Accessibility is your guide to creating email campaigns that work for everyone. Click here to learn more.

A better version would ditch the "click here" stuff, shortening the copy in the process:

The Better Email On Accessibility is your guide to creating email campaigns that work for everyone.

This is especially important for people with disabilities who rely on assistive technology (which we'll discuss more in Chapter 5) like screen readers to read content out loud to them. Most screen readers, when encountering a link, will literally speak the word "Link" followed by the text included in the hyperlink. For the first example, it would read as:

Link Click here to learn more

Which isn't very descriptive. However, the second example sounds better:

Link The Better Email On Accessibility

Which tells the user exactly where that link it taking them.

The same principle applies to standalone calls-to-action, like buttons. Again, some examples are helpful. Which is more descriptive here?

Again, the first option tells you what to do but suffers from the same problems described above. The second is clearer, more actionable, and works well for people using screen readers. And that's the criteria we're using to grade accessible links:

- Is it clear and descriptive?
- Is it actionable?
- Does it work well for screen readers?

If you can answer yes to those three questions, you're well on your way to making more accessible experiences for your subscribers.

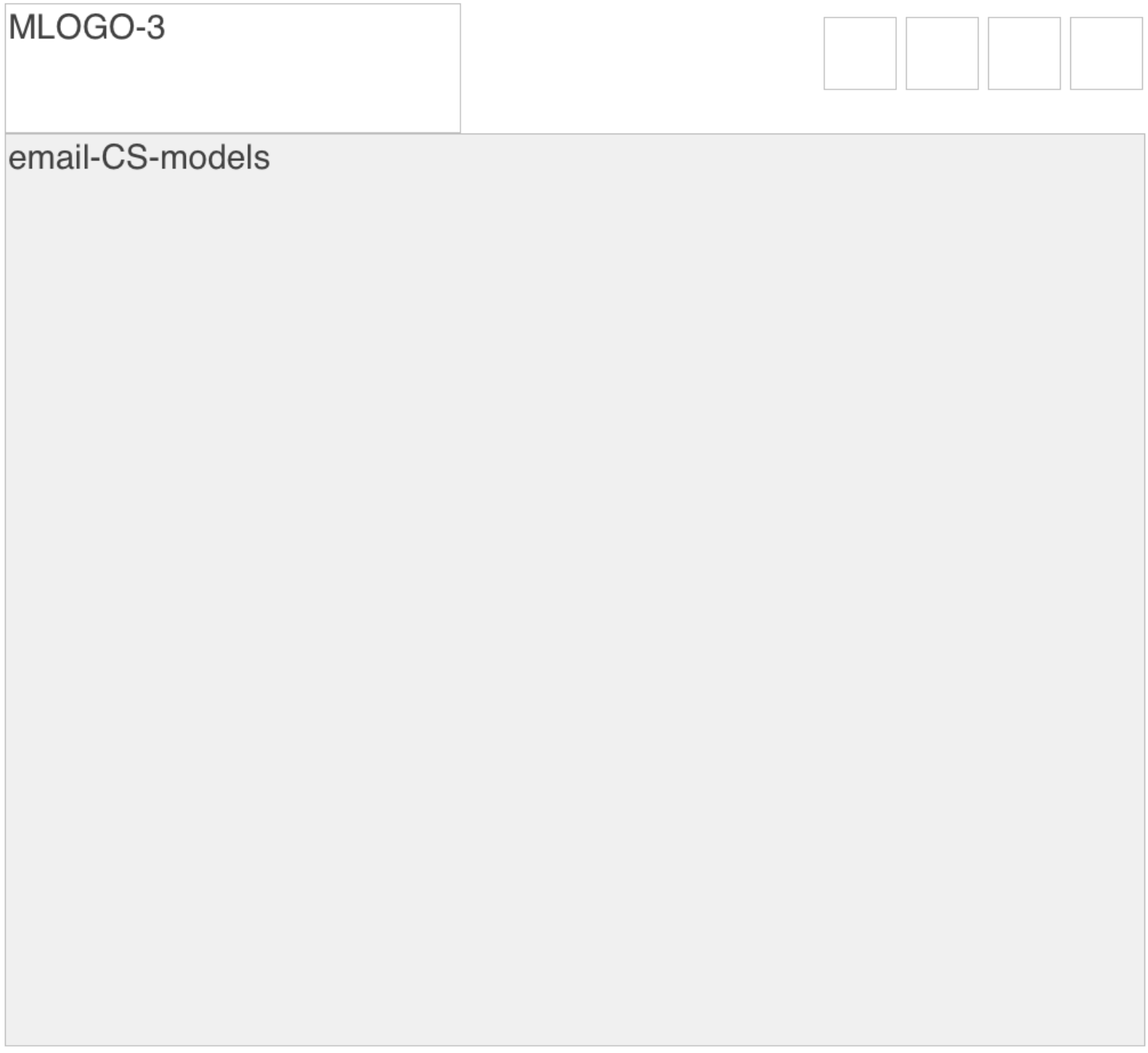
Write Good Alternative Text for Images

The last bit of copy that you need to worry about is the alternative text for any images included in a campaign.

Alternative text (often called ALT text) is additional text that describes the content of an image for non-sighted users who rely on screen readers to consume content. In the email world, ALT text is especially important considering that many email clients block the loading of images by default, or users do the same for security or bandwidth reasons.

A few years back, **Litmus did a study** that showed that image blocking in email clients affects around 43% of all emails.

In those situations, you'll run into emails that look like this:



While ALT text is absolutely vital for non-sighted users, it's still extremely helpful for sighted users with images turned off. Most email clients will display the ALT text (in this case it's the "email-CS-models" bit) when images are blocked, so it can be used to provide additional context in those cases. If you're running a sale and your image has a big "50% Off Until Friday", that's a great candidate for ALT text.

We'll see how you actually include ALT text in the markup of an email in Chapter 4, but it's something that should be crafted with as much care as the rest of the copy in your email.

Your ALT text should:

- Repeat the text in an image if there is any, unless that text is repeated in copy surrounding the image.
- Describe the contents of an image when there isn't text in that image.
- Act as a replacement for the image and work in the context of the overall email without changing its meaning.
- Be empty if the image is purely decorative.
- Describe the result of an intended action if it's an actionable image like a button or interface icon (social media icons, follow icons, etc.).
- Include proper punctuation, as screen readers will use it to read content naturally.
- Be succinct while still providing enough information.
- Be fun if it works for your brand!

Similar to links, screen readers identify images before reading the ALT text, so you don't need to say it's an image in your copy. Far too many people use something like, "Image of the author" for the ALT text, which would be read as:

Image Image of the author

Instead, something like this would be more descriptive for the user:

Image Jason Rodriguez smiling at the camera while sitting in his office surrounded by computers, guitars, and music making equipment.

The second option is wordier—which flies in the face of our cardinal rule of writing shorter copy when possible—but it's far more descriptive and creates a richer, more rewarding experience for a user. Not every image would need that descriptive of ALT text, but it can be useful for hero images or more editorial emails like newsletters.

My two main pieces of advice when writing alternative text are to:

1. Take writing it off of autopilot. Put actual thought into your ALT text.
2. Read it out loud along with the surrounding copy to make sure it makes sense.

Good alternative text can make or break a campaign for a lot of users, so it's worth spending the time and energy to get it right.

Chapter Wrap Up

That's a lot to take, so let's summarize. If I were to provide a list of tips for writing accessible copy, here's what it'd look like:

Do	Don't
Have a clear goal.	Try to do too much in any one email.
Use plain language.	Use lots of jargon or acronyms.
Make copy readable.	Use lots of clichés or complex metaphors.
Use shorter words, sentences, and paragraphs.	Write too much!
Translate and localize when needed.	Blindly automate translations.
Write descriptive, actionable link text.	Use "click here" or "more" for your links.
Write descriptive, actionable alternative text	Repeat words like "link" or "image" for screen readers.
Read ALL of your copy out loud.	Forget to read your copy out loud. Seriously.

In the next chapter, we'll take our copy and look at how to visually design it to create the most accessible experience possible for sighted users.

Chapter 3

Accessible Visual Design

This is the part where we take the content of an email—the copy, links, images, and alternative text—and put it together to create what people will actually open in their email inboxes.

From a visual perspective, there's a lot you can do to create better experiences for sighted users. While it's easy to think most subscribers will see the same way you do, that's absolutely not the case. Visual disabilities are among the most common, and cover a wide range of disabilities—permanent, temporary, and situational. Here are the most common:

Color Blindness

Color blindness affects around 8% of men and 0.5% of women and comes in many varieties. Most people think color blindness means that someone only sees in shades of grey (**monochromacy**), but that's actually the most rare form of color blindness. More common are:

- **Tritanopia**, where people confuse blue with green and yellow with violet.
- **Deuteranopia**, where reds look lighter and get confused with green.
- **Protanopia**, where pinks appear blue and dark reds and blacks are confused.
- **Red-green color blindness**, which is when people suffer from both deuteranopia and protanopia and can't distinguish between reds, greens, browns, and oranges.

Above: Normal color perception, protanopia, deuteranopia, protanopia, and monochromacy or total color blindness.

Color blindness is one of the major reasons we need to worry about the proper use of color, contrast, and affordances like underlines on links when designing accessible emails. We'll see what that means in just a bit.

Visual Acuity

Your visual acuity is your eye's ability to distinguish shapes and details of objects, and works alongside your color and depth perception to determine how well you see. Ever have an eye exam where you read this letters at a distance? That's called a Snellen test and measures your visual acuity. Common visual acuity disabilities include:

- **Astigmatism**, which is a distortion of the cornea and results in blurry or distorted shapes and objects.
- **Myopia**—or nearsightedness—which allows you to see close objects clearly but distant objects appear blurry.
- **Hyperopia**—or farsightedness—which allows you to see distant objects clearly but nearer objects appear blurry.

Since visual acuity is largely a result of the curvature and smoothness of the cornea and lens in the eye, these disorders can usually be corrected with eye glasses or contact lenses. There's a good chance you or someone you know deals with one of the above on a daily basis.

Eyesight Loss

More severe—and usually less treatable—are diseases and disorders that damage the nerves or blood vessels in and around the eye.

- **Macular degeneration**—which is usually age-related—is the biggest cause of blindness in adults. It leads to a loss of vision in the center of your field of vision, making it difficult to read, watch videos, look at pictures, or do common tasks.
- **Glaucoma** comes from damage to the optic nerve and makes it difficult or impossible to see the edges of your field of vision, kind of the opposite of macular degeneration.
- **Diabetic retinopathy** is a result of diabetes damaging the blood vessels in your eye's tissue and causes dark or blind spots across your entire field of vision.
- **Cataracts** occur when the lens of the eye develops cloudy areas, which can result in faded or blurry vision, halos, and trouble seeing in bright or dark environments.

Above: Normal vision and simulated eyesight loss with macular degeneration, glaucoma, and diabetic retinopathy.

Even among people with "good" vision, there are differing levels of ability, especially when we account for situational factors, like dimmed screens, sunny days, or every day distractions.

It's not uncommon for people to dim the screens of their devices to conserve their batteries, and that dimming typically lowers the contrast of elements on the screen, making them harder to distinguish. A similar effect happens when viewing a screen—especially glass screens like on our phones—in direct sunlight or bright light. And the dozens of distractions we experience at any given moment detract from our brain's ability to quickly consume and make sense of visual information.

Apart from visual impairments, many people contend with cognitive and motor disabilities that make it hard to perceive, understand, and operate email campaigns.

Dyslexia makes it hard for people to read longer, more complicated blocks of copy. While we took steps to write shorter, more readable copy, the visual display of that copy matters to those users. And people with limited motor control could find it difficult to interact with links or buttons in an email depending on their size and position relative to other email elements.

For all of these reasons, we need to employ some well-tested design principles to make sure that emails can be used by the widest range of users. These include:

- Creating a strong hierarchy to aid scannability.
- Using sufficient contrast between elements.
- Using color to effectively to keep emails operable.
- Aligning text appropriately to enhance readability.
- Using inclusive imagery.
- Being thoughtful about the layout of the campaign.
- Adhering to common affordances and norms to keep emails usable.

Let's dig in and see how each of those concepts is applied.

Fair warning: This is where we'll start seeing actual code.

The intricacies of coding emails is outside the scope of this guide, but if you want to dig in deeper, you should check out The Better Email On Design—an in-depth guide to coding email campaigns from scratch.

Create an Email Hierarchy

One of the best things you can do to increase the usability of your emails is to create a visual hierarchy in your campaign.

A hierarchy is simply an arrangement of information to clarify the relative importance of different pieces of information. In the context of email, it's arranging

the copy, images, links, logos, buttons, and anything else so that subscribers can quickly determine what's primary, secondary, and tertiary information.

Creating a strong visual hierarchy allows subscribers to scan an email (remember, different abilities and short attention spans matter!) and find what they need quickly. Essentially, it consists of making important information bigger and more noticeable and less important information smaller or less noticeable. And, before you go telling me that everything in your email is important, just remember:

If everything is important, nothing really is.

Something will always need to take precedence in your campaign. It's usually the headline of the email, a key offer, a specific article, or the call-to-action itself. If you went through the process of creating an email brief for your campaign, you will have identified a clear goal for your campaign. Anything supporting that goal is the most important information. Focus on prioritizing that in the hierarchy of your email.

Let's look at an example to illustrate the point.

In the mockup above, the email on the left essentially boils down to a large wall of text that very few subscribers are likely to read. In the email on the right, you can see clearly distinguished sections and elements that allow the subscriber to quickly skim

the campaign and find what's important. And the larger block at the top—maybe a headline or image—immediately draws the subscriber's attention. Which do you think will be more understandable and usable?

When it comes to creating a strong hierarchy in email, we have a few tools available:

- Content size
- Content color
- Content placement
- Spacing between content

Using Size

The size of a piece of content is typically the first thing someone is drawn to. That's why headlines are usually bigger than the body copy of an article. When a newspaper wants to catch your eye, they do it using massive headlines.

"All the News That's Fit to Print"

The New York Times

LATE CITY EDITION

Weather: Rain, warm today; clear tonight. Sunny, pleasant tomorrow. Temp. range: today 80-66; Sunday 71-66. Temp.-Hum. Index yesterday 69. Complete U.S. report on P. 50.

VOL. CXVIII . No. 40,721 © 1969 The New York Times Company. NEW YORK, MONDAY, JULY 21, 1969 X 10 CENTS

MEN WALK ON MOON

ASTRONAUTS LAND ON PLAIN; COLLECT ROCKS, PLANT FLAG

Voice From Moon: 'Eagle Has Landed'

EAGLE (the lunar module): Houston, Tranquility Base here. The Eagle has landed.

HOUSTON: Roger, Tranquility, we copy you on the ground. You've got a bunch of guys about to turn blue. We're breathing again. Thanks a lot.

TRANQUILITY BASE: Thank you.

HOUSTON: You're looking good here.

TRANQUILITY BASE: A very smooth touchdown.

HOUSTON: Eagle, you are stay for T1. [The first step in the lunar operation.] Over.

A Powdery Surface Is Closely Explored

By JOHN NOBLE WILFORD

Special to The New York Times

HOUSTON, Monday, July 21—Men have landed and walked on the moon.

Two Americans, astronauts of Apollo 11, steered their fragile four-legged lunar module safely and smoothly to the historic landing yesterday at 4:17:40 P.M., Eastern daylight time.

Neil A. Armstrong, the 38-year-old civilian commander, radioed to earth and the mission control room here:

Above: The July 21st edition of The New York Times announcing the moon landing. From The New York Times archive.

The same applies to email. If the headline is the most important piece of information (usually outside of the CTA itself) in an email, make it bigger. This happens via HTML and CSS, for example:

```
<h1 style="color: #000000; font-size: 3em; font-weight: bold;
line-height: 1.2; margin: 2em 0 3em 0;”>This is Really Important,
Folks</h1>
```

In the example above, a heading level 1 HTML element is used since it is semantically correct (which we'll discuss more in the next chapter). Using inline CSS, the color is set to black, the **`font-size`** is set to a generous 3 ems, the **`font-weight`** is set to bold to contrast with surrounding text, and there is ample spacing on the top and bottom of the headline using the **`margin`** property.

A similar approach can be applied to any heading element, as well as paragraphs and other text elements to change their sizing and reinforce your hierarchy.

Generally speaking, your body copy should be between 16 and 20 pixels, with headlines larger to denote their importance. Tertiary information—like that in the footer of your email—can be smaller, but it's typically not a good idea to go below 14 pixels for your font-size. Anything below that is extremely difficult to read and some email clients, like Apple Mail on iOS, will automatically bump up the font-size to 14px for small copy, which could lead to unintended display issues in your email. We'll talk more about font sizing in the technical accessibility section.

Using Color

The color of elements can be used to reinforce hierarchy, too. But you need to remember not to rely on color alone to convey the importance of an element. As we've already seen, people perceive colors differently, so what is apparent to you may not be as apparent to all of your subscribers.

For the most part, darker, more saturated colors draw the eye and are used for more important information. Lighter, less saturated colors are easier to glance over, so should be used for secondary information.

In CSS, color is defined using the **`color`** keyword, which takes a keyword (like **`black`** or **`rebeccapurple`**), hexadecimal value (**`#000000`** which is black), or another value (which typically isn't as well supported in email).

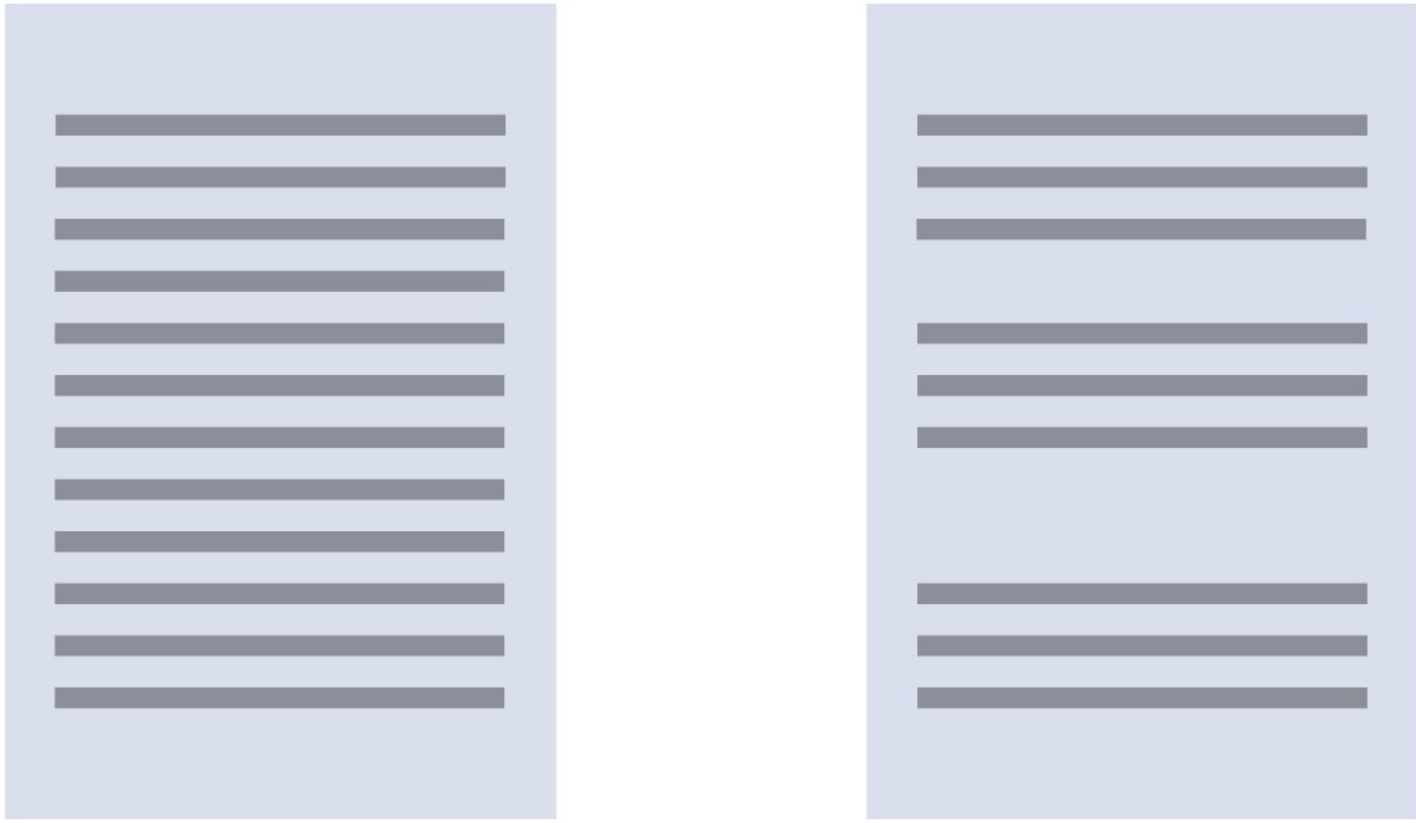
In a basic example, a headline could be pure black or a dark color with a subtitle slightly lighter and the body copy even lighter than that.

It's important to keep contrast high enough for people to read, though, so be careful when picking your colors. We'll discuss contrast later on in this chapter.

Using Placement & Spacing

The placement of content is important in establishing hierarchy, too. Although I don't subscribe (pun intended) to the concept of "the fold" in email mattering that much, things that are placed higher up in an email are typically seen as the most important since that's what people see when they first open a campaign. In that sense, the fold is real, but people are natural scrollers these days, so you can still place important information lower in an email.

Placement goes hand-in-hand with spacing around pieces of content. Placement and spacing allow you to group related information and create sections within your email. Those sections—you guessed it—reinforce the hierarchy.



Although the lines in the example above are the same size, adding spacing and grouping content creates distinguishable sections that are easier to scan. In HTML, grouping can happen in a variety of ways, from using HTML sectioning elements like **`div`**, **`section`**, or **`article`**, or through structural elements like **`table`**, **`tr`**, and **`td`**. Spacing is applied by adding **`padding`** and **`margin`** to different elements.

Combined, the use of size, color, placement, and spacing help you to create a hierarchy in an email. More importantly, they help you create more accessible experiences for your users.

Use Contrast & Color Effectively

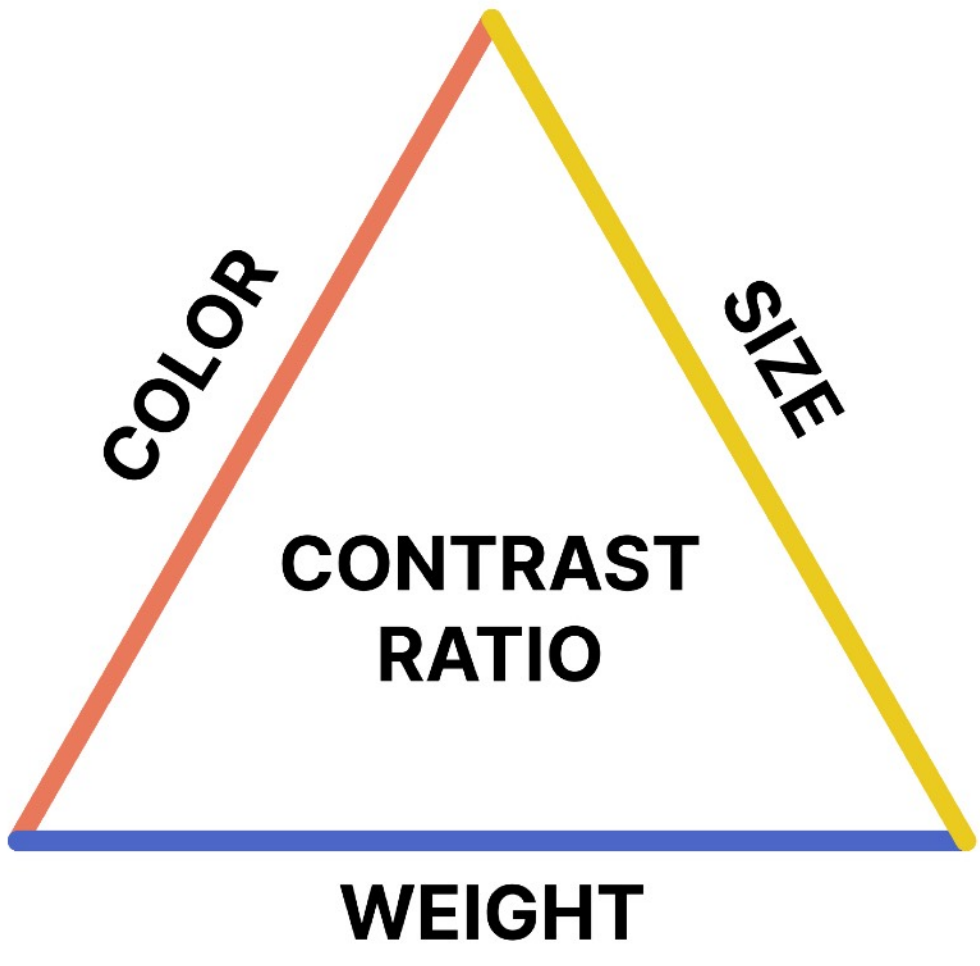
Contrast is the difference between two elements. It's directly related to a user's ability to perceive content, the P in our POUR acronym. For the most part, it means the color difference between the background and foreground elements in an email. That being said, size and font weight play into contrast, too.

The Web Accessibility In Mind (WebAIM) group has an excellent article covering the WCAG rules that determine the appropriate contrast and color for different elements. Fortunately, the WCAG rules are fairly clear about what works for users. Most of their

guidelines boil down to determining the contrast ratio between elements, which varies on a scale from 1:1 to 21:1. 1:1 is equivalent to white text on a white background, which is imperceivable. 21:1 is equivalent to black text on a white background, which is very perceivable to most sighted users. While there are different levels of acceptable contrast ratios that correspond to the three levels of compliance (A, AA, and AAA), generally speaking, the higher the contrast ratio, the better—at least to a certain extent.

Some people do suffer from scotopic sensitivity syndrome (SSS), which is a rarer visual disorder that leads to super high contrast text (like black on white) appearing to shimmer, swirl, or wobble on a screen. It's sometimes linked to dyslexia, which can cause further challenges for comprehending text. For that reason, it might be worthwhile to choose dark grey text instead of black, or change the background color to something other than pure white to slightly reduce the contrast ratio.

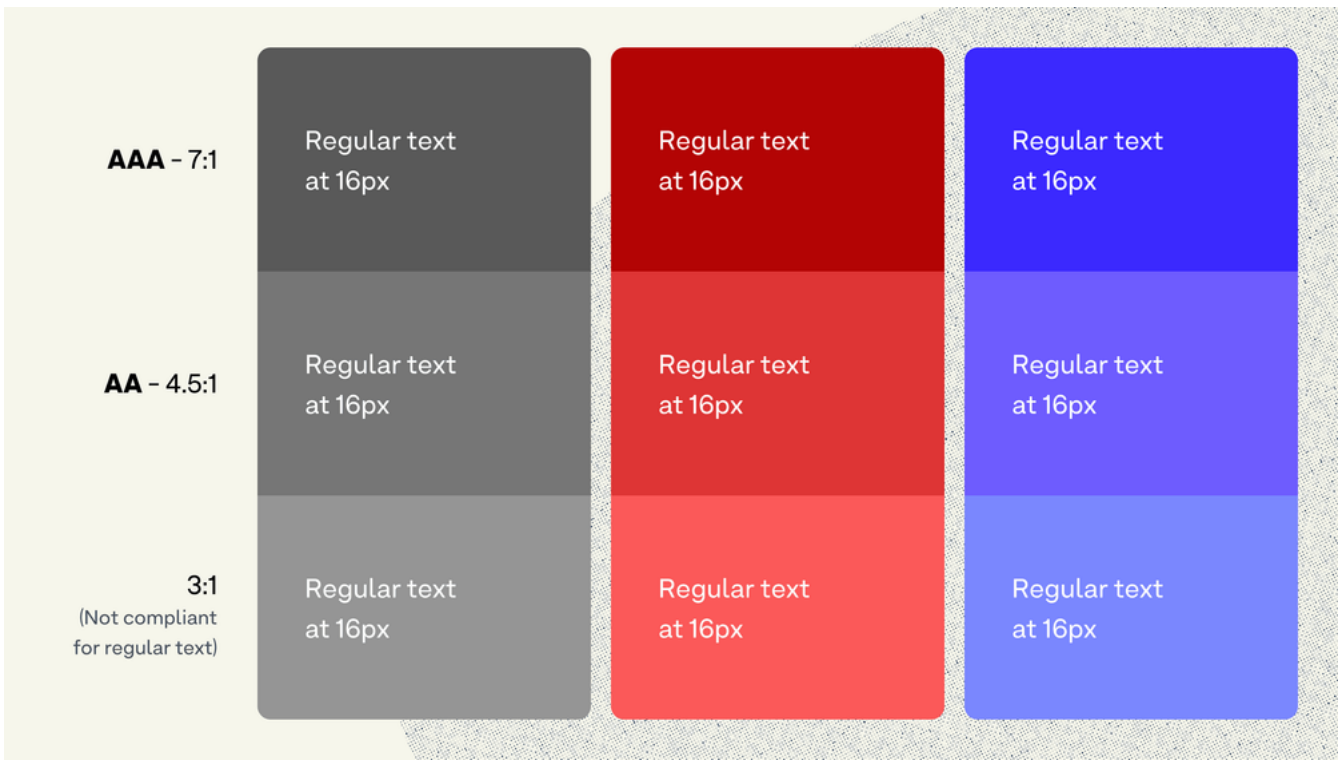
Font size and weight affect the contrast ratio, too. Smaller, normal weighted text typically needs a higher contrast color to be perceivable, while larger, bolder text can get away with using lower contrast colors since the size and weight of the text makes up for the lower color contrast. They're essentially three sides of the contrast triangle.



There are plenty of tools out there that allow you to test the contrast ratio of your content. Many of them are free and we'll discuss a few options in Chapter 5. Once you start testing contrast ratios, you can compare them to the different compliance levels set forth in the WCAG 2.0 specification:

- **Level A** doesn't have hard contrast ratio rules, and should be avoided.
- **Level AA** requires a contrast ratio of at least 4.5:1 for normal size copy and 3:1 for larger copy or bold text.
- **Level AAA** requires a ratio of at least 7:1 for normal size copy and 4.5:1 for larger copy and bold text.

The contrast ratio doesn't just apply to black or grey text on a white background, but all colors of text on all background colors. Something to keep in mind when styling your content or creating dark mode emails (which we'll discuss in Chapter 4). The accessibility tool **Stark** has a great graphic that illustrates different contrast ratios, taken from **their excellent blog**.



Apart from worrying about contrast ratios, you need to ensure that you're not relying on color alone to convey key information. This mostly applies to links within the text of an email campaign.

As you'll recall, different types of color blindness affect how people perceive colors. So what looks like blue to you may look like green or red to me. Unfortunately, designers far too often use color alone to style a text link. Take this link for example:

This is a bit of copy with a link in it. Can you find the link? It might not be where you expect.

If you don’t suffer from color blindness, you’ll see some of the copy displayed in blue, denoting the linked text. However, if you’re color blind, it’s harder to see:

This is a bit of copy with a link in it. Can you find the link? It might not be where you expect.

But, by combining the **`color`**, **`font-weight`**, and **`text-decoration`** CSS properties, we can make a link that’s far more perceivable:

This is a bit of copy with a link in it. Can you find the link? **It should be clear** when it doesn’t just use color.

Even when color blindness is accounted for, the link is still perfectly usable:

This is a bit of copy with a link in it. Can you find the link? **It should be clear** when it doesn't just use color.

My recommendation is to combine those three properties on each of your text links to ensure that all of your subscribers can benefit from your emails.

Create Readable Text

We talked about readability from a copywriting perspective, but readability can be aided by our visual design, too. In particular, the text alignment, line length, and spacing between lines and paragraphs play an important role in creating readable copy. Additionally, text alignment is an important consideration when copy is translated into some languages like Arabic, which is read right-to-left (RTL) instead of left-to-right (LTR) like English.

Ideal Line Lengths

Let's start by looking at line lengths. The line length is exactly what it sounds like: how long a line of text is before it wraps to the next line.

If lines are too short, it forces the user's eyes to move too rapidly and can cause fatigue or an anxious feeling. It also means that you'll end up with more lines and force the user to scroll a lot. If lines are too long, it makes reading difficult since the user has to spend more time finding the start of the next line—there's physically more distance for the eye to cover. It may seem like a small thing, but it adds up, especially for people who deal with cognitive disorders that make reading difficult to begin with.

Too Short
~ 30 characters

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Et ante amet sollicitudin vitae facilisis. Sed orci consectetur felis nisi, pharetra sit sollicitudin porttitor. A diam nisi interdum etiam.

Too Long
~ 105 characters

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Et ante amet sollicitudin vitae facilisis. Sed orci consectetur felis nisi, pharetra sit sollicitudin porttitor. A diam nisi interdum etiam. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Et ante amet sollicitudin vitae facilisis. Sed orci consectetur felis nisi, pharetra sit sollicitudin porttitor. A diam nisi interdum etiam. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Et ante amet sollicitudin vitae facilisis. Sed orci consectetur felis nisi, pharetra sit sollicitudin porttitor. A diam nisi interdum etiam.

Above: Some example less than ideal line lengths.

Line length is measured in the number of characters on a line. While there's no universal arbiter deciding what the ideal line length is, there's a general consensus that it sits between around 50-85 characters. Some sources go as low as 40 characters and as high as 100 characters.

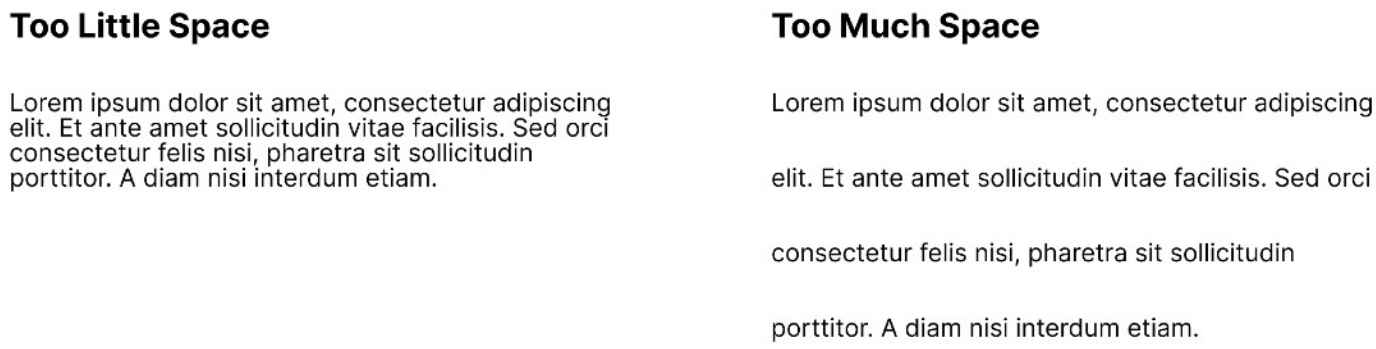
In email, line length depends on two things: **1)** The width of the container element in which the text sits and **2)** the size of the text as determined by its **`font-size`**. To complicate things, different fonts have different sized characters. Even when they're set at the same pixel size, fonts will vary in their actual size.

The average container width of an email sits around 600 pixels. At that width, a basic font like Arial set at 16px yields a line length of around 85 characters. At 18px, you get about 75 characters, and at 20px, it clocks in at around 70 characters. All of those are good options and provide text that is big enough for most users to read.

Although there is a **ch** unit in CSS, which would allow you to specify the width of a container in characters, it's not well-supported across email clients, so you'll mostly find container widths specified as pixels, percentages, ems, or rems.

Ideal Line Spacing

Equally important as line length is the spacing between lines. Again, we're faced with a Goldilocks situation: Too little space between lines and text is cramped and hard to read, too much and spacing and the lines seem unconnected. You need to find the amount of space that feels just right.



Above: Some example less than ideal line spacing.

In email, line spacing is set with the CSS **`line-height`** property, which can take a number of values but is usually set in either pixels or a unitless number. A few guidelines when setting your line spacing:

- Smaller text—like body copy—usually requires more spacing between lines.
- Larger text—like headlines—can handle less spacing between lines.
- According to the W3, you should use line spacing of "a space-and-a-half" within paragraphs.

In code, that might look something like this with pixels:

```
<p style="color: #000000; font-family: sans-serif; font-size: 1em; font-weight: normal; line-height: 24px; margin: 0 0 1em 0;"></p>
```

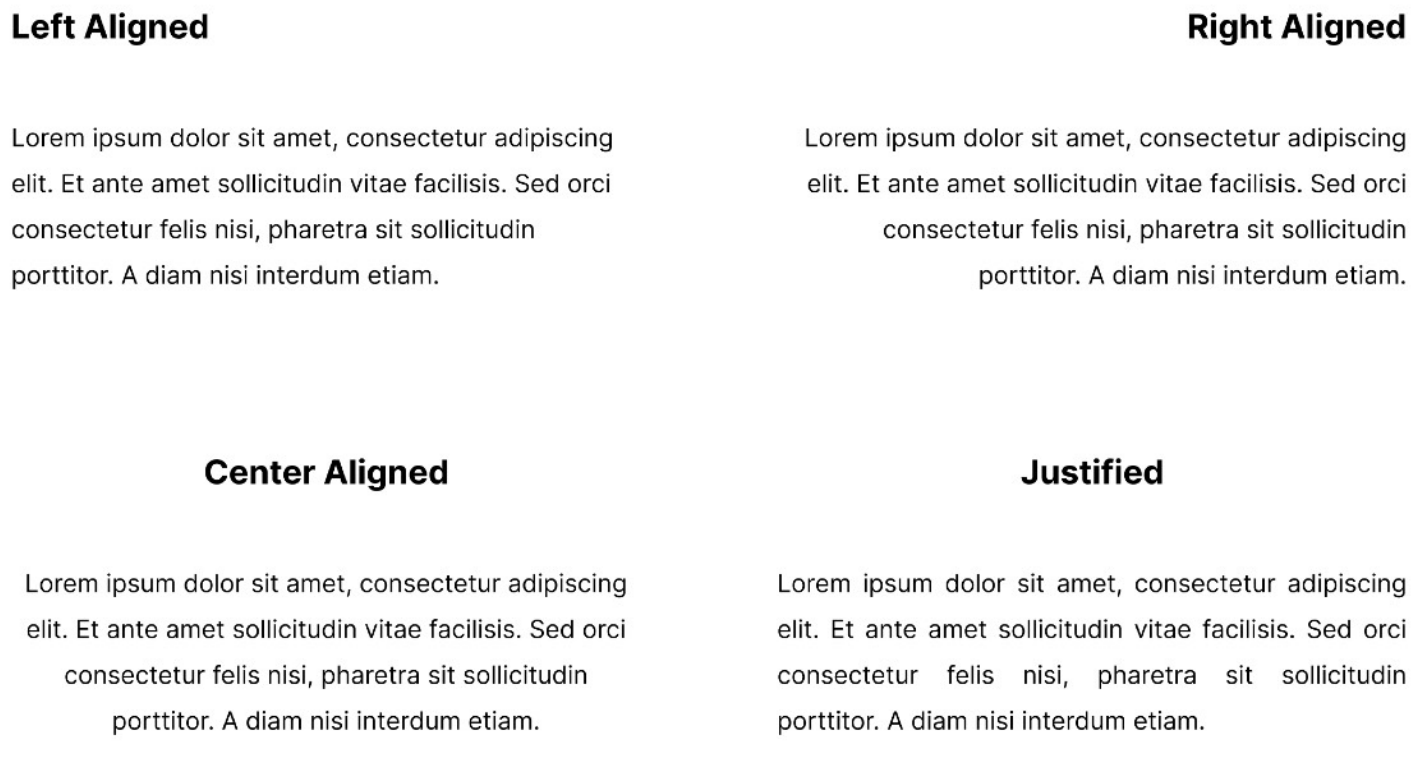
Or this with a unitless value:

```
<p style="color: #000000; font-family: sans-serif; font-size: 1em; font-weight: normal; line-height: 1.8; margin: 0 0 1em 0;"></p>
```

In practice, determining the ideal line length and height is as much art as it is science. It depends heavily on the fonts used, the length of the content itself, and what aesthetic you are chasing. Experiment until you find the ideal line length that works for your email campaign.

Aligning Text Effectively

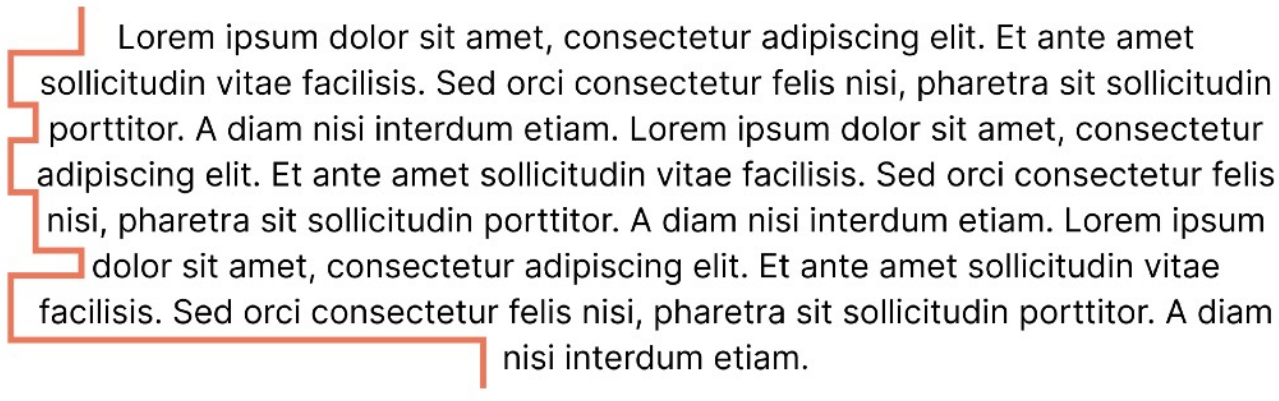
Finally, let's talk about text alignment. Text alignment is the placement of the text in regards to the left and right edges of a container. Although, within CSS, there are a number of different types of alignment, you really only see four in practice:



Alignment is important because it helps a reader's eyes have a consistent starting point for new lines, especially for longer bits of copy. Without that consistency, text becomes difficult to read—especially in the limited time spent in most emails.

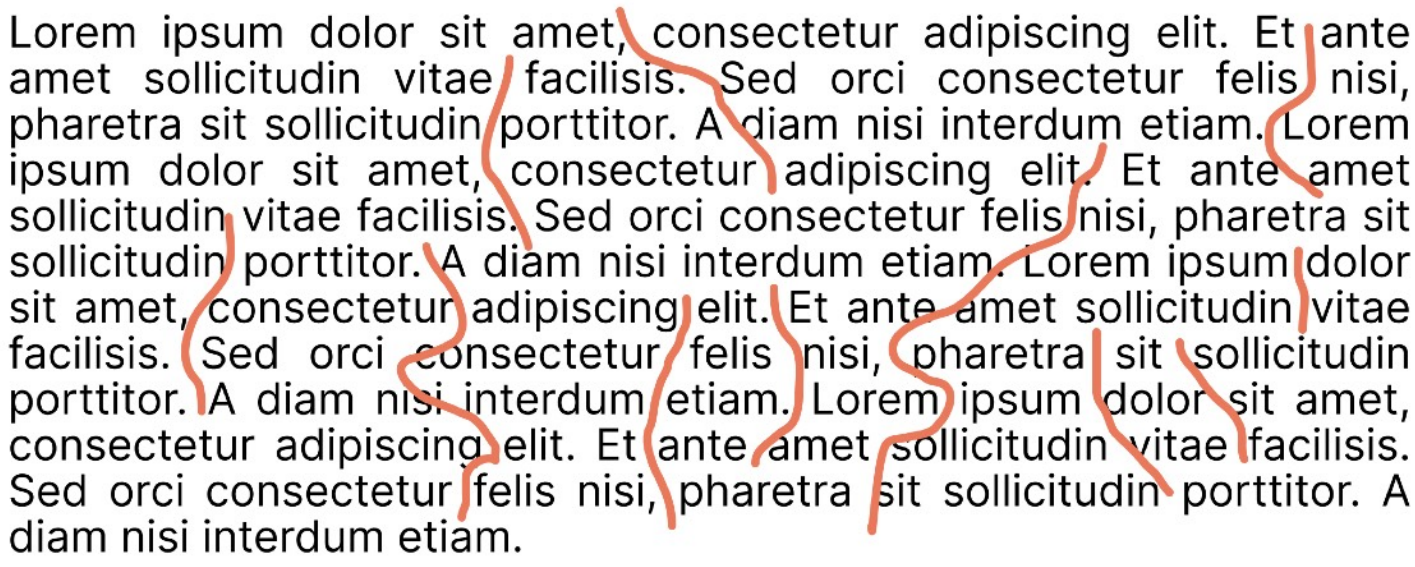
Therefore, **you almost always want to use left or right aligned text**, depending on what language your copy is written in. For languages that are read left-to-right, like English, most of your copy should be left aligned. For languages that are read right-to-left, like Arabic, most of your copy should be right aligned.

Although centered text makes an email look pleasingly symmetrical, longer blocks of copy become very difficult to read. Look at the inconsistency:



Those are a lot of hurdles for someone to overcome. As a general guideline, copy that is at most 3 lines long can be centered, while anything longer should be left or right aligned, again depending on the language used. This typically means headlines, shorter paragraphs or standalone sentences, and captions.

While it can be tempting to chase symmetry with justified text—which forces spacing between words so that both the left and right sides are aligned—another problem arises:



Justification typically introduces “rivers of whitespace,” like what’s pictured above. Squint your eyes and you’ll see even more rivers than the ones I highlighted. Again, theses extra spacing creates small hurdles that add up to make reading difficult for some users, limiting the value they can get out of your campaigns.

In email, text alignment is set using the CSS **`text-align`** property, and can take values like **`left`**, **`right`**, and **`center`**.

Most browsers and email clients use left aligned text by default, so you wouldn't even need to set the property unless you're overriding it. For example, with a headline:

```
<h1 style="color: #000000; font-family: sans-serif; font-size: 60px; font-weight: bold; line-height: 1.5; margin: 0 0 60px 0; text-align: center;"></h1>
```

There are other technical considerations when using right-to-left languages, but we'll look at those in the next chapter.

Create Accessible Layouts

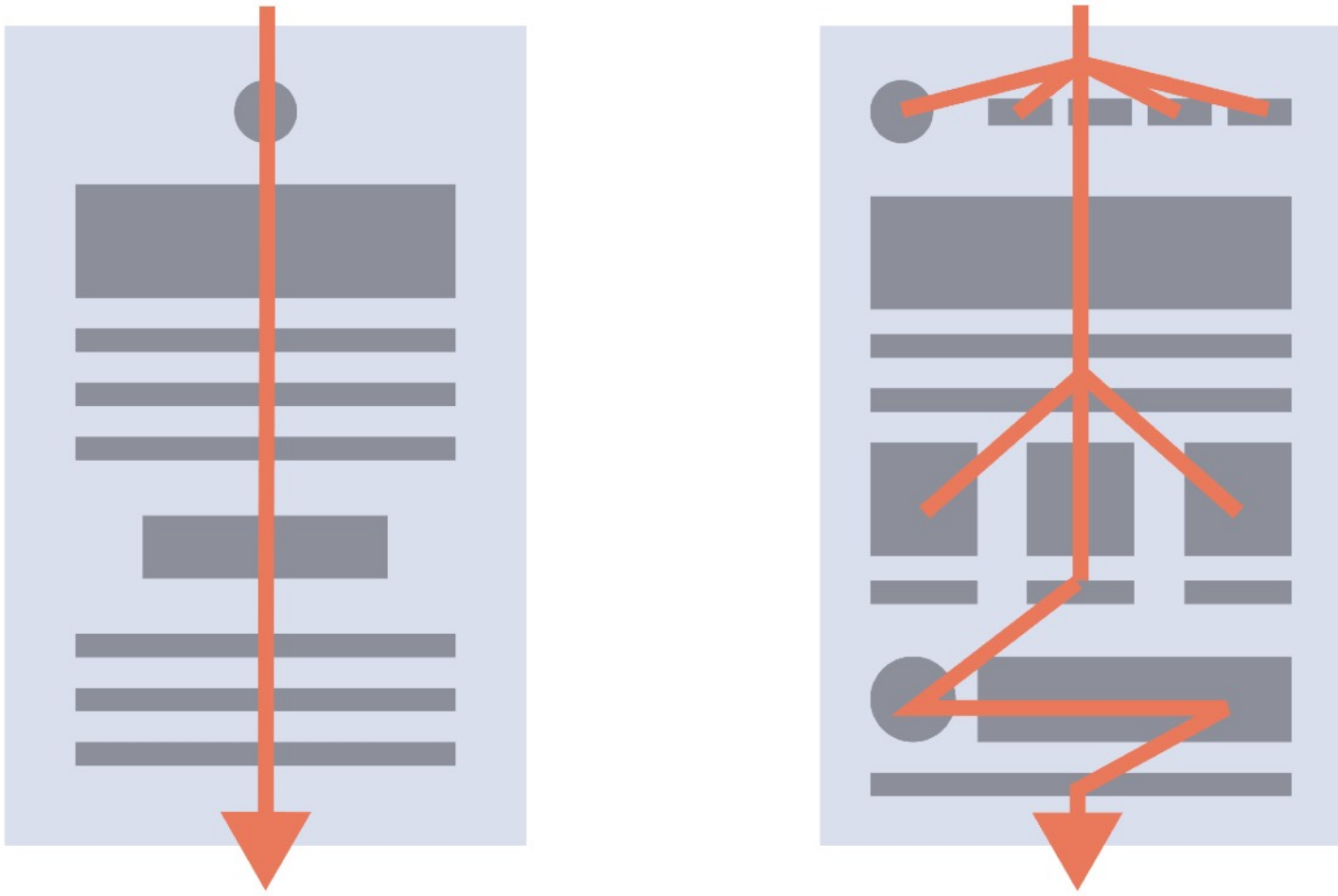
Finally, let's talk about how layouts can impact accessibility.

Email layouts can range from simple, single-column designs to complex ones with multiple sections, columns, and varying element sizes. Those elements can be spaced out or cramped together. And every variation impacts how easily someone can scan and understand the information in a campaign.

As with copy, layouts tend to work best when they are simple and logical. Complex layouts can lead to a sort of overwhelmingness for users. They can get easily distracted by different elements and it can be hard to focus on any single one. Complex layouts—at least when not designed thoughtfully—can also dilute the hierarchy you've up until now spent so much time worrying about.

That's why I'm typically a fan of single-column layouts, with the addition of two- or three-column sections as needed. A single-column layout streamlines the presentation of information, making it easier for people to scan and understand.

Laying out an email is all about leading someone through a predefined, ideal path. A single-column design makes that easy. More complex layouts give subscribers multiple paths down which they can go:



That’s not to say that every email needs to be a single-column design. Some emails can benefit from more complex layouts to lead a user’s eyes. In particular, a zigzag or “S” type layout is effective for laying out lots of information in a relatively compact way while directing users through that information.

Fender does this well in their emails. They use a two-column design with each stacked section flip-flopping the position of the images and text.

CHROME COVERED SHAWBUCKERS HH

Featuring d[illegible] dual-coil pickups with alnico 2 magnets for vintage-style tone.

CORONA CLASSIC

Delivers modern Strat tone with glassy h[illegible]ghs, pronounced m[illegible]range and tight lo[illegible]s.

TEXAS SPE[illegible]AL

Classic Stra[illegible]ound punched up with higher output — perfect for sending your amp into overdrive.

GENERATION 4 NOISELESS™ HSS

Single-coil [illegible]ups paired with a Sha[illegible]ker™ humbucki[illegible]g pickup for vintage-style tone with added bite.

The eyes naturally move from element to element, section to section, and let you work your way through the campaign with minimal effort. If all of that content was stacked into a single column, it'd be excessively long—at least on the desktop.

For mobile, longer emails are usually fine (within reason), as people are used to scrolling on smaller devices. But the limited space of mobile devices means that complex layouts feel extra cramped. Therefore, you should strive to use responsive design techniques to stack elements into a single-column layout on mobile.

Chapter Wrap Up

Visual design comprises a lot of different things and can get complex quickly. But, by following the strategies described above, you can ensure that your emails are accessible to the most sighted users as possible.

When in doubt, follow these guidelines:

Do	Don't
Understand the range of disabilities that affect how you design an email.	Mistake your abilities for everyone else's.
Create a strong email hierarchy to call attention to important information and make scanning easy.	Make every element the same size and color.
Use size, color, position, and spacing to reinforce that hierarchy.	Rely on color alone to convey meaning.
Use high contrast for low vision users.	Forget to test your contrast ratio against WCAG accessibility guidelines.
Use color effectively to convey meaning.	Use choose text alignment for it's aesthetic symmetry.
Write descriptive, actionable link text.	Create overly complex, confusing layouts.
Create readable text by combining font size, line length, line spacing, and text alignment.	
Layout content for to aid accessibility.	

In the next chapter, we'll dig deeper into the HTML and CSS that powers every email campaign and uncover the technical considerations that affect accessibility.

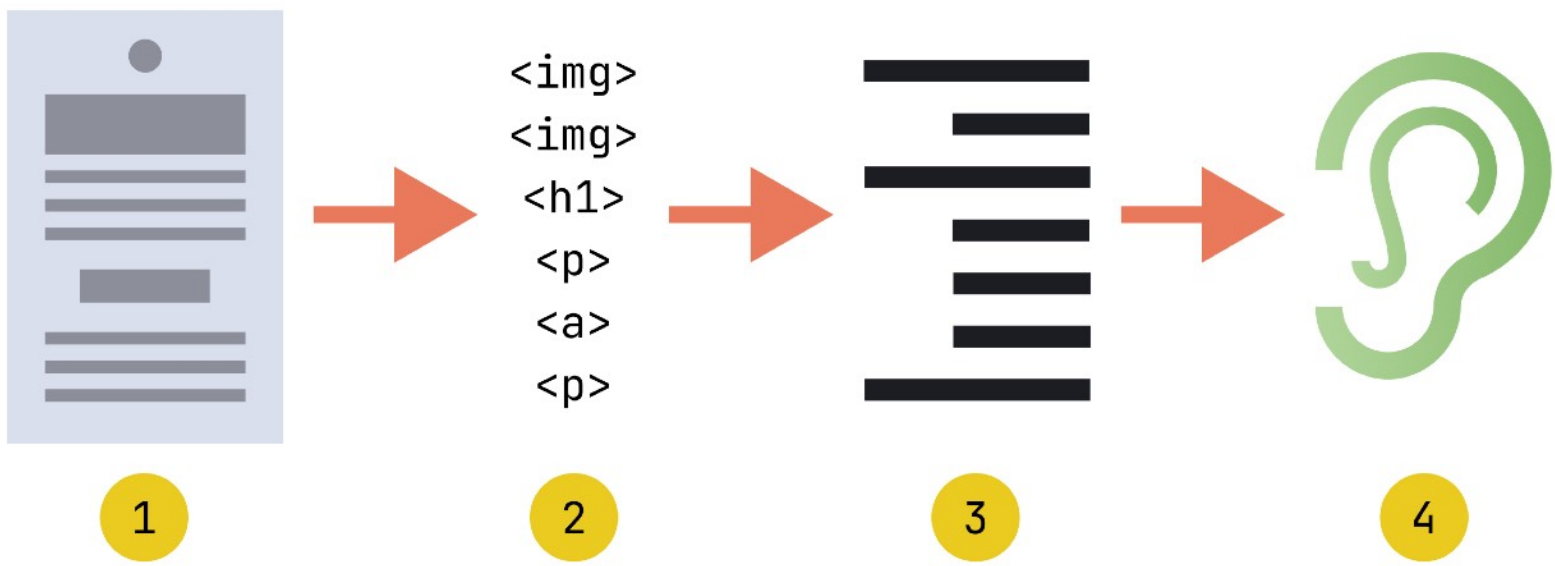
Chapter 4

Technical Accessibility

Remember in the first chapter when we discussed what accessibility actually means? There was one part that dove into assistive technology and how tools have been developed to help people with disabilities.

The one we focused on most—due to email's digital and visual nature—was screen reader software, which is used to read the content on a screen out loud to a visually impaired user.

All screen readers work by analyzing the underlying code of an app, web page, or—in our case—email campaign, using that code to develop a structure and meaning to the content displayed, and then using that to read the content out loud while allowing users to navigate around the structure and interact with content as needed.



The process illustrated above includes:

1. The email being opened by the user.
2. The underlying HTML being analyzed by the screen reader.

3. The screen reader assembling what's called "the accessibility tree," which is a navigable outline of the content.
4. The screen reader reading navigation waypoints and content to the user through speakers or headphones.

From there, the user can use their keyboard to navigate through the email, interact with specific content like links, and take action within the email client, like archiving the email or replying to it.

It sounds technical and it is. But, unlike the copywriting and visual design processes, technical accessibility is relatively straightforward and includes a handful of best practices to create accessible emails. This section is where we'll look at those practices and how to code effective, accessible email templates.

Email's Underlying Code

First, let's refresh ourselves on what an email actually is.

Every email campaign is a collection of text and, usually, images. While the text can be the copy you wrote, it's always marked up in some kind of code that a computer can read and make sense of. Even plain text emails to friends and family are transmitted along with some code that the email client on the other end can understand. For marketing campaigns—everything from newsletters to transactional emails—emails are usually built in a code editor or visual editor that produces HTML and CSS as the final output.

HTML—or HyperText Markup Language—is the code used to describe and structure the content of an email. It consists of individual tags, each of which denotes a specific type of content, and attributes, which add additional information to those tags.

CSS—or Cascading Style Sheets—is the code used to style the elements in an HTML document. CSS allows you to select HTML elements and use rules and values to describe to a browser or email client how those elements should be displayed.

HTML documents, saved as files with the **`.html`** file extension, consist of a **`head`** and **`body`**. The head of the document is where you include information to describe the overall document and how it should be rendered, while the body is where the actual content of your email lives. Here's a basic HTML document:

```
<!DOCTYPE html>
<html>
<head>
  <style>
  </style>
</head>
<body>
</body>
</html>
```

The first line is setting the **`doctype`**, or document type, of the HTML. In the example above, we're using the HTML5 doctype. There are a number of different doctypes from past versions of the HTML spec, but the HTML5 one works well across email clients and is the shortest one, so it's recommended.

The next line opens up the HTML with the **`<html>`** tag, which is closed again at the bottom. Within the HTML tags are the **`<head>`**, where we'll add a bunch of information soon, and the **`<body>`**. Each is closed out appropriately.

Again, all of those tags can take attributes, which provide additional information to computers. And both the head and body sections can include any number of further HTML elements, each with their own use case and importance. We've already seen attributes in some of the code examples earlier in the guide, for example when we set the line-height on a paragraph tag:

```
<p style="color: #000000; font-family: sans-serif; font-size:
1em; font-weight: normal; line-height: 1.8; margin: 0 0 1em 0;"
></p>
```

The **`style`** attribute is used to include inline CSS that tells the email client how the paragraph should be displayed.

The Head Section

In order to create accessible emails, it's critical that we include key information—both tags and attributes—in the head of an HTML document. Let's see what that looks like before explaining what each part accomplishes.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-
scale=1 user-scalable=yes">
  <meta name="x-apple-disable-message-reformatting">
  <meta name="color-scheme" content="light dark">
  <meta name="supported-color-schemes" content="light dark">
  <title></title>
  <style>
    :root {
      color-scheme: light dark;
      supported-color-schemes: light dark;
    }
  </style>
</head>
<body>
</body>
</html>
```

We've already explained the doctype. Moving on the the **<html>** element, we've included:

The **lang** attribute, which describes which language the document is set in. In this case, it's value is **en** for English. If you're writing in Spanish, it'd use **es**. Afrikaans would be **af**, and so on. [Here's a list of all of the available language codes](#).

Similarly, we set the text direction of the language using the **dir** attribute. In this case, it uses **ltr** for left-to-right. For a right-to-left language, like Urdu, it'd be set to **rtl**.

The language attribute has a big effect on most screen reader software, as it can trigger different language profiles. Those language profiles use different accents and pronunciations to ensure that the user can understand the content as intended. While we're setting both the language and text direction globally for the email in the head, you can use the lang and dir attributes on individual HTML elements within the body of your email if you are mixing multiple languages in one campaign.

A common example is a footer that includes legal language in multiple languages. You may have a paragraph set in English, which would inherit the language from the global declaration in the head, but then a following paragraph of the same copy translated into Spanish right below it. In that case, you'd include the lang attribute on that specific paragraph tag to override the global declaration:

```
<p lang="es"></p>
```

Next, within the head itself, we include a number of **`meta`** tags that accomplish different things.

- The first sets the **`charset`**—or character set—which helps characters render correctly.
- The second provides the initial dimensions and scale of the email and allows users to override those settings by pinching or zooming.
- The third disables a "feature" in Apple Mail on iOS that can cause weird display issues and make it hard for users to read an email.
- The next two both tell different email clients that both light and dark versions of the email are supported, so they'll respect the user's preference.
- The **`title`** tag is used to provide a title for the email, which is used to populate tabs when an email is opened in a browser, which happens far more often than you probably think. It's good to include the sender name or subject line here.
- Finally, a **`style`** element is used to add some embedded CSS into the document that, again, declares support for both light and dark mode emails.

After that, we get to the body section, which is where you'll add all of the actual content to your email campaign.

Use Semantic Markup

For a long time, HTML emails were marked up entirely with tables. This was due to a lack of support for a lot of HTML elements across email clients, with Microsoft Outlook and Lotus Notes being the major villains. Both lacked (and in some cases *still* lack) full support for proper HTML and CSS like what you'd see on modern websites, especially

when it came to positioning elements within an email, so email developers had to hack their way to decent looking campaigns. The result: Nearly everything being implemented with the **`table`**, **`tr`**, and **`td`** HTML elements and spacer GIFs.

While tables are still widely used in email, they don't inherently lend themselves to an accessible experience (except when used for their intended purpose).

A more accessible way to code emails—and one that's been in use for decades on the web—is to use semantic markup, which is inherently more accessible and helps people that rely on assistive technology to better consume and understand the content in a campaign.

What is semantic markup? If semantics is the study of meaning, then semantic markup is code that conveys meaning. Simply put, it's using HTML elements as they're supposed to be used, which is to describe the content within them. HTML has most of what we need for accessibility built in, most email developers just don't take advantage of it, yet.

Visual users use the information we designed into the email to navigate it, make sense of content, determine what's tappable, etc. The hierarchy allows them to quickly scan headings, determine what's important, and then read a specific section's body copy as needed. Colored, underlined links let them know they can be activated. Images give them additional information. All of that happens visually, but what about for non-sighted users? People using screen reader software or braille displays?

They rely on the underlying code to make sense of the content. Unfortunately, when non-semantic code is used, it's harder for them to do that. Semantic code, though, provides landmarks for easier navigation, summarizes links so that they can easily be cycled through, and all around provides much-needed context for non-sighted users.

Let's look at a simple example:

```
<table>
    <tr>
        <td>
            The summer sale is on!
        </td>
    </tr>
    <tr>
```

```
        <td>
            Get exclusive deals between now and July 30th. Save
big on all of your favorites and get ready for the best summer
ever!
        </td>
    </tr>
    <tr>
        <td>
            <a href="https://example.com">Start Saving Today</a>
        </td>
    </tr>
</table>
```

This is a structure a lot of emails use to mark up content. While it doesn't have any styling applied to it for simplicity's sake, you can see that there are three pieces of content: A heading, a paragraph of text, and then a link. For sighted users, CSS would be used to visually create hierarchy between the headline and paragraph and show that the link was clickable.

But for non-sighted users, most would either hear the content read out as if the markup didn't exist (not too terrible of an experience) or they'd hear something like:

Table table row one of three table cell The summer sale is on! Table row two of three table cell Get exclusive deals…

And so on. There's no real structure provided other than that of the table itself, and it's a terrible experience for the user. A better approach would be to use semantic markup to convey that there's a headline, a paragraph, etc.

```
<table>
    <tr>
        <td>
            <h2>The summer sale is on!</h2>
            <p>Get exclusive deals between now and July 30th.
Save big on all of your favorites and get ready for the best
summer ever!</p>
            <a href="https://example.com">Start Saving Today</a>
```

```
        </td>
    </tr>
</table>
```

You can still use a table for the underlying structure (we'll see how to take it out of the accessibility tree in the next section), but the content itself uses the HTML **h2** tag to denote a level 2 heading and a **p** tag to show a paragraph of text. Someone using a screen reader would then be able to quickly navigate through the different headings and drill down into each section as needed.

Different Types of Semantic Markup

Now that we've seen an example of semantic markup, let's look at the different types of semantics that HTML can natively convey. These are broken up into four major groups: Document structure, textual meaning, media, and correlating tags.

Document structure tags do exactly what they say: They provide a structure to the document itself so that it can be easily navigated both by computers and assistive technology users. Think of these as the containers for the different sections of your email. They consist of the following tags:

- **`header`**
- **`footer`**
- **`main`**
- **`nav`**
- **`section`**
- **`aside`**
- **`article`**

When it comes to structural markup, things are complicated in email. This is because emails are almost always displayed as part of a webpage or application, not standalone. And most webpages and applications have their own structure which a user is trying to navigate—the email is simply content within that larger structure.

Therefore, it doesn't necessarily make sense to use the larger page structure for an email (or email clients themselves just don't support the HTML). So, you can—at least of now—ignore most of those HTML elements in favor of the humble **`div`** element, which is a generic container element. The exceptions are **`article`**, which can be used to wrap an entire email since that's kind of what an email is anyways, **`section`** to break up smaller sections in an email if they need the additional structure (could be helpful for longer emails), and the **`nav`** element if there are navigation links within the email.

In the next section, we'll look at using Accessible Rich Internet Applications (ARIA) to add some of that semantic value back in when needed, but you'll mostly be using tables and divs in emails.

Textual meaning tags are used to classify text according to what that text is. These are the headings and paragraphs which give the user context. These tags include:

- Headings like **h1**, **h2**, **h3**, **h4**, **h5**, and **h6**
- Paragraphs **p**
- Strong (usually bolded) text **`strong`**
- Emphasized (usually italicized) text **`em`**
- Quoted text **`blockquote`**
- Addresses **`address`**
- Time **`time`**
- A generic span of text **`span`**

And plenty more that are typically more specialized and won't make their appearance in an email campaign.

Headings are particularly important. The are one of the main mechanisms for navigating a document. Generally speaking, every web page should have one and *only* one level 1 heading (**`h1`**), followed by as many additional levels as needed to structure the rest of the content on the page. Headings descend in level of importance from 1 to 6.

You may be asking yourself: If web pages should only have one **h1**, then isn't it problematic to include on in an email that's displayed in a webmail client like Gmail? Good question.

Most email clients don't actually add an **h1** into their interface. The ones that do aren't widely used (with the exception being web.de). In most cases, that heading is wrapped around the company logo or uses the site's name, not the title of any one document displayed on the page.

A study from a few years ago showed that most screen reader users preferred having an **h1** with the document's title as opposed to a site name, or even *two* **h1**s with both pieces of information (which is against the HTML spec, but that's how it goes). So, even when an **h1** is already included on a page, one in the email that's used to provide information about the email is still helpful to users. It's rare that it'll cause any major issues at all.

So, it's far more helpful to include an **h1** in an email than it is to leave it out.

The other textual tags are more straightforward to use. Paragraphs of text should be wrapped with the **p** tag. If you need to target specific text within a sentence or paragraph, the generic **span** tag is useful. It doesn't really *mean* anything, but can be used to apply inline styles as needed. If you want something bolded and made to stand out, the **strong** text is a good choice. Emphasizing text? Use the **em** tag.

The **blockquote**, **address**, and **time** tags aren't used often, but have their places. Quotes can be good when using social proof like testimonials, but a lot of email clients apply their own styling to the **blockquote**, so you'll need to override that styling with inline CSS. Addresses can be included (and should be to be compliant with laws like CAN-SPAM) in the footer of your email or when displaying store or location information, and the **time** tag can be included for things like events.

Media tags are used to mark up media elements like images, audio, and video. Since video and audio don't work consistently across email clients, they can be ignored. Images, though, are used frequently in email. Although there are newer elements like the **picture** and **figure** elements, nearly all of the emails you'll come across (or code yourself) will simply use the **img** tag. The **img** tag takes a source attribute which points to the absolute location of the image file itself, and should contain appropriate

alternative text and styling information as needed (see **The Better Email On Design** for more info on marking up images).

Finally, **correlating tags** group pieces of information together. Within email, the ones you'll see are in the form of lists:

- Ordered lists **`ol`**
- Unordered lists **`ul`**
- List items **`li`**
- Description (or definition) lists **`dl`**
- Description terms **`dt`**
- Description details **`dd`**

There are lots of other HTML elements, but most aren't well-supported or commonly used within the context of email. If you want to see a full list of HTML elements, **The Mozilla Developer Network (MDN) Web Docs** is a great place to dig deeper. If you want to see how each is supported across the most popular email clients, there's no better place than **Can I Email?**, a website developed by **Rémi Parmentier** and **Tilt Studio**, which allows you to search for individual HTML tags and CSS properties and see how well they're supported.

We'll see how all of these elements are put together into a full email template at the end of this chapter. But for now, let's return to ARIA…

Understanding ARIA

I mentioned Accessible Rich Internet Applications (ARIA) before, but let's look at what that actually is.

ARIA is an initiative from the W3C Web Accessibility Initiative (WAI) that includes a spec consisting of **roles** and **attributes** that can be included within HTML to provide additional clarification on the roles of different elements in a document. It's another layer that can enhance the semantics already provided by HTML or add semantics where none natively exist.

It was developed in response to the web being used to deliver increasingly complex applications, many of which created new usage cases that posed accessibility problems. Hence the name Accessible Rich Internet *Applications*. Although an email campaign itself isn't an application, we can still use ARIA to make it more accessible.

But first, it's important to understand **the first rule of ARIA**:

> ***If you can use a native HTML element or attribute with the semantics and behavior you require already built in, instead of re-purposing an element and adding an ARIA role, state or property to make it accessible, then do so.***

ARIA is not a replacement for good semantic markup, and in some cases can even make a document or app *less* accessible. That's one of the reasons that you'll often hear the phrase, "No ARIA is better than bad ARIA." You should never just apply ARIA roles to random elements in an attempt to improve accessibility. Instead, you should rely on semantic HTML first, and override role as needed due to the limitations of email clients.

Again, ARIA consists of **roles** and **attributes**. Roles are used to describe the role of an element to a user, similar to semantic HTML. They can be applied to individual HTML elements like so:

```
<table role="presentation">
```

There are 87 different ARIA roles, which can be broken up into 6 categories:

- **Document structure roles** like article, header, footer, feed, presentation, directory, table, etc.
- **Widget roles** that describe common interactive elements like scrollbar, slider, tab, searchbox, etc.
- **Landmark roles** which provide document organization and include banner, form, main, navigation, region, etc.
- **Live region roles** which describe elements that will be dynamically updated like alert, log, status, timer, etc.

- **Window roles** that define sub-windows of a main window, like alertdialog and dialog.
- **Abstract roles** which aren't intended for us to use, but for the underlying browser itself.

For email, the two categories you'll see used are document structure roles and landmark roles—both of which are designed to make navigation of a document easier for users relying on assistive technology.

The example above of a table using the **`presentation`** role is a good example. As we saw in the section on semantic markup, tables have their own semantics and can lead to poor experiences for subscribers using screen reader software. Tables are used to mark up tabular data (think of an Excel spreadsheet), not to layout a web page or email. But, email clients being what they are, we use tables in an unintended way. To override their intended role, we include **`role="presentation"`** which takes the table itself out of the accessibility tree. It's essentially saying that this markup is for presentation purposes only and doesn't have any inherent semantic value. [MDN has a great description] of what happens:

> *The content of the element will still be available to assistive technologies; it is only the semantics of the container — and in some instance, required associated descendants — which will no longer expose their mappings to the accessibility API.*

Now, instead of a screen reader reading out each individual table, table row, and table cell, it skips over that markup and focuses on the content instead. If you're using semantic markup, that'll be the headings, paragraphs, lists, images, and links that we want people to actually focus on.

Presentational roles need to be included on every `table` tag in an email—unless a table is presenting actual tabular data. Unfortunately, nested tables don't inherent the role from its parent. This is one of the single best things you can do to create a more accessible experience for your users.

Apart from tables, we can include various roles to convey semantic value elsewhere in an email, especially in regards to the document structure.

Although you could use HTML elements like **header**, **footer**, **section**, **main**, and **article** in an email, a number of email clients don't support them. In particular, some versions of Microsoft Outlook, Yahoo! Mail, AOL Mail, and Russia's Mail.ru will update or remove those tags entirely, including any classes or inline styles, leading to broken layouts.

In these cases, it's more helpful to use ARIA roles on a generic element like a **div** or **span** to provide that context to screen readers. For the most part, the naming conventions are similar, just applied via the **role=""** pattern:

- **<header>** becomes **role="banner"**
- **<footer>** becomes **role="contentinfo"**
- **<main>** becomes **role="main"**
- **<article>** becomes **role="article"**
- **<section>** becomes **role="region"**
- **<nav>** becomes **role="navigation"**
- **<aside>** becomes **role="complementary"**

So, instead of:

```
<article>Some Content</article>
```

You'd have:

```
<div role="article">Some Content</div>
```

Like we saw in the section on semantic markup, it doesn't make a ton of sense to use larger page structures in an email since those will be included in the email client itself. Therefore, you're mostly going to use **role="article"** and, occasionally, **role="section"** in an email.

A common approach is to wrap the entire email content (everything between the opening and closing **body** tags) in an article, like this:

```
<body>
    <div role="article" aria-roledescription="email" aria-
label="the name of your email">
       EMAIL CONTENT GOES HERE
    </div>
</body>
```

That will expose the entire email as an article to the accessibility tree, making it a landmark which can be easily navigated to by a user. There are two additional ARIA-related snippets included on that div, too.

Both are ARIA **attributes**, which is the second half of the ARIA spec. Attributes (or states or properties) allow you to modify an elements properties, as opposed to describe the content inside of that element. There are 36 different attributes, most of which you'll never encounter in an email. They are bucketed into categories like widget, live region, drag-and-drop, relationship, and global attributes. The two exceptions are the **`aria-roledescription`** and **`aria-label`** attributes seen above.

`aria-roledescription` overrides the description of the **`role`** to make it more human readable. In our case, since the campaign is an email and not necessarily an article, it makes more sense to describe it as an email to the user, although the underlying code will expose it as an article within the accessibility tree. It doesn't override the functionality of the **`role`**, it just allows you to make it more friendly when read out to a subscriber.

`aria-label` allows you to define the name of an element. In the case of an email, it's used to provide a name or title to the email article. Yes, we include the **`title`** tag in the head of the email, but that's only really applicable to browser tabs in most cases. The **`aria-label`** will allow you to title the email itself when it's read in the context of the email client.

While you can put anything you want in there, it's typically a good idea to use something that won't change from email to email or use something that's dynamically inserted via a merge tag in your email service provider, like the subject line. Most people will forget to update this boilerplate code from email to email, so making it adaptable to every email campaign is a good safeguard against creating confusion for subscribers.

In my own newsletter, I use the following container:

```
<div role="article" aria-roledescription="email" aria-label="A Regular Communication from Jason Rodriguez" lang="en" dir="ltr" style="background-color: #ffffff; font-size: medium; font-size: max(16px, 1rem);">
```

Since my email newsletter is called Regular Communication, I use that to help populate the **`aria-label`**. You'll also notice that both the **`lang`** and **`dir`** attributes are set here as well as on the **`html`** tag itself. This is because some email clients will strip the **`html`** tag, so having it here will ensure that the proper language and text direction are used.

Finally, you can see some inline styles that, most notably, set the **`font-size`** twice, once to **`medium`** and once to **`max(16px, 1rem)`**, which is a method to ensure that the font sizing respects the user's preferences. Since many users rely on browser or OS-level screen zooming to better see content, we want to allow them to utilize that behavior in our emails. Using rems allows them to zoom successfully, and the **`medium`** declaration acts as a fallback for clients that don't support **`max`**. Mark Robbins has a great writeup of this on his Good Email Code website.

Font Sizing in Email

Fonts (and the sizing of lots of things, including margins and padding) can be set in a variety of values in CSS. Some are absolute—in that they don't change—like pixels (**`px`**), points (**`pt`**), and inches (**`in`**). Some are relative—which will change based on the sizing of other things—like ems (**`em`**), relative ems (**`rem`**), viewport width (**`vw`**), and viewport height (**`vh`**). There are even keywords like **`small`**, **`medium`**, **`large`**, **`x-large`**, and **`xx-large`**.

Most emails traditionally use pixels for setting the font size, which is very consistent between most email clients, but doesn't respect users settings as well as relative values like **`rem`** and **`em`**.

Users can set their browser, email client, or entire operating system to use different zoom settings. Some users make text, or everything on their screen, smaller. Some

make it all bigger by zooming in. Regardless, we want to respect that decision, and relative units allows for easier scaling.

Root ems (**`rem`**) are flexible units that scale elements relative to the root of the document, in an email's case the HTML itself. Ems (**`em`**) are relative to the current font size of an elements container, so it can vary between elements if their containers use different sizes. And it can compound if you're not careful, leading to absurd effects like massively large text or spacing between elements.

On the web side of things, rems are more and more the default for setting element sizing, with pixels used when specific spacial relationships need to be kept in tact. **Josh W. Comeau has a great writeup of how this relationship should work**. Unfortunately, not all email clients support rems well, so, for most things, you should define them as ems.

What are ems? We can think of 1 em as the default size for a browser or email client. But that default size differs between email clients, and can depend on how the user themselves has enabled zoom features for scaling their display.

If an email client has a default font size of 16px, then 1 em = 16px. This also maps to the keyword **`medium`**. Email clients range on the small side with Apple Mail on Desktop (default is 12px) to Outlook on Windows (default is 18px) on the larger side of things. That's just for the base body copy. Every browser and email client has its own specific default sizes for different elements like headings and the **`small`** tag.

If the default is 16px (which is a good baseline), then text set at 20px would be equivalent to 1.25em. 32px (which is double the default) is equivalent to 2em.

Most existing email templates and visual mockups rely on pixels to determine sizes, but you'll want to get into the habit of converting those values to ems instead. If you don't want to remember the math, then use **a handy conversion tool like the one from W3Schools**.

Ems can be used for spacing between elements, too. For things like **`margin`** and **`padding`**—which are the most commonly used elements to add whitespace to a design—use ems to preserve scaling when zoom settings are enabled.

I'll admit that I still use pixels from time to time—out of habit, reusing older code, and sheer laziness—but I'm trying to get into the habit of using ems to make my emails

more accessible. Like I've written before, accessibility is a journey and requires constant updates (especially to existing code), so there's always going to be work to do.

Coding Alternative Text

During the copywriting chapter, we looked at what makes good alternative text for an image. Now it's time to look at how to include that alternative text on your images.

The good news it that it's extraordinarily easy. All that's needed is the addition of the **`alt`** attribute to an **`img`** element, along with the copy of the alternative text. If I had my headshot in an email, it'd look like this:

```
<img src="headshot.jpg" alt="Image Jason Rodriguez smiling at the camera while sitting in his office surrounded by computers, guitars, and music making equipment.">
```

What's cool about email (at least in some email clients) is that you can actually use inline CSS to style that text, too. This styling is displayed when images are turned off, and can help your email stay on brand. You simply use the same styles you would with any other text, just applied to the `img` tag:

```
<img src="headshot.jpg" alt="Jason Rodriguez smiling at the camera while sitting in his office surrounded by computers, guitars, and music making equipment." style="color: rebeccapurple; font-size: 1.4em; font-weight: bold; line-height: 1.2;>
```

For sighted users, they'll see some nice, big, purple text. For non-sighted users, screen reader software will pick up the alternative text and read it out loud as:

> *Image Jason Rodriguez smiling at the camera while sitting in his office surrounded by computers, guitars, and music making equipment.*

Two other scenarios can occur with alternative text. The first is including the **`alt`** attribute but not providing a value of that text:

```
<img src="example.jpg" alt="">
```

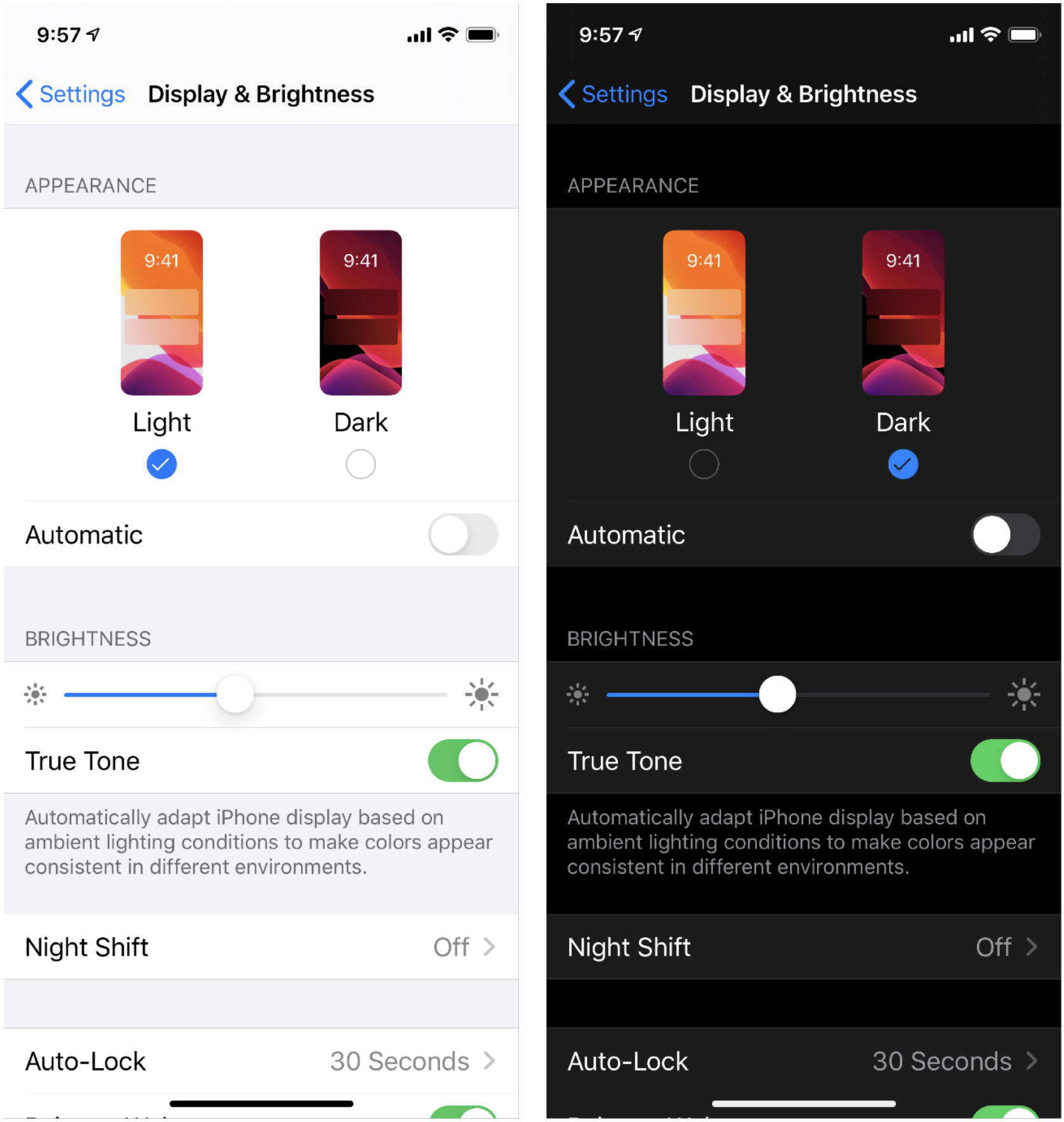
In this case, you're saying that, while there is an image here, it is used only for presentational purposes. It doesn't have any real semantic value, so it can be skipped over entirely. This is very useful for things like borders or dividers, which are mostly decorative and don't have a purpose beyond styling the content visually.

The second scenario is leaving the **`alt`** attribute out entirely. In this case, most screen readers will tell the user that there is an image there, but not be able to provide any additional context about that image. This is a less than ideal situation and should be avoided by provided appropriate alternative text or context clues in the text surrounding the image.

It boils down to *always* including an **`alt`** attribute on every image element. If the image is decorative, leave the attribute value empty. If it's not, then provide descriptive alternative text per the guidelines in Chapter 2.

Creating Dark Mode Emails

The last step in creating an accessible email is adding dark mode support. Dark mode is an alternative display setting that applies a darker color palette to a screen, app, or web page that can be less taxing on people's eyes—especially in lower light environments or at night.



You can see dark mode in action in the screenshot above, which compares the settings page on an Apple iOS device in both light and dark mode settings.

Although dark mode is commonly seen as a simple user preference—some people just like the darker look—there are a number of reasons someone might choose to enable dark mode.

Depending on what colors the designer implements in dark mode, the contrast on the screen can be higher or lower than the equivalent light mode experience. This

enhanced or reduced contrast can make it easier or harder for the user to consume content, depending on their own abilities.

Dark mode can also be helpful for people around the user, as the darker screen means less ambient lighting bothering others. For example when on an airplane at night and fellow passengers might be sleeping, dark mode can help reduce disturbances.

Finally, some users enable dark mode to help preserve battery life on their devices. The reduced screen brightness is less taxing on the battery, making it more efficient when frequent charging isn't an option.

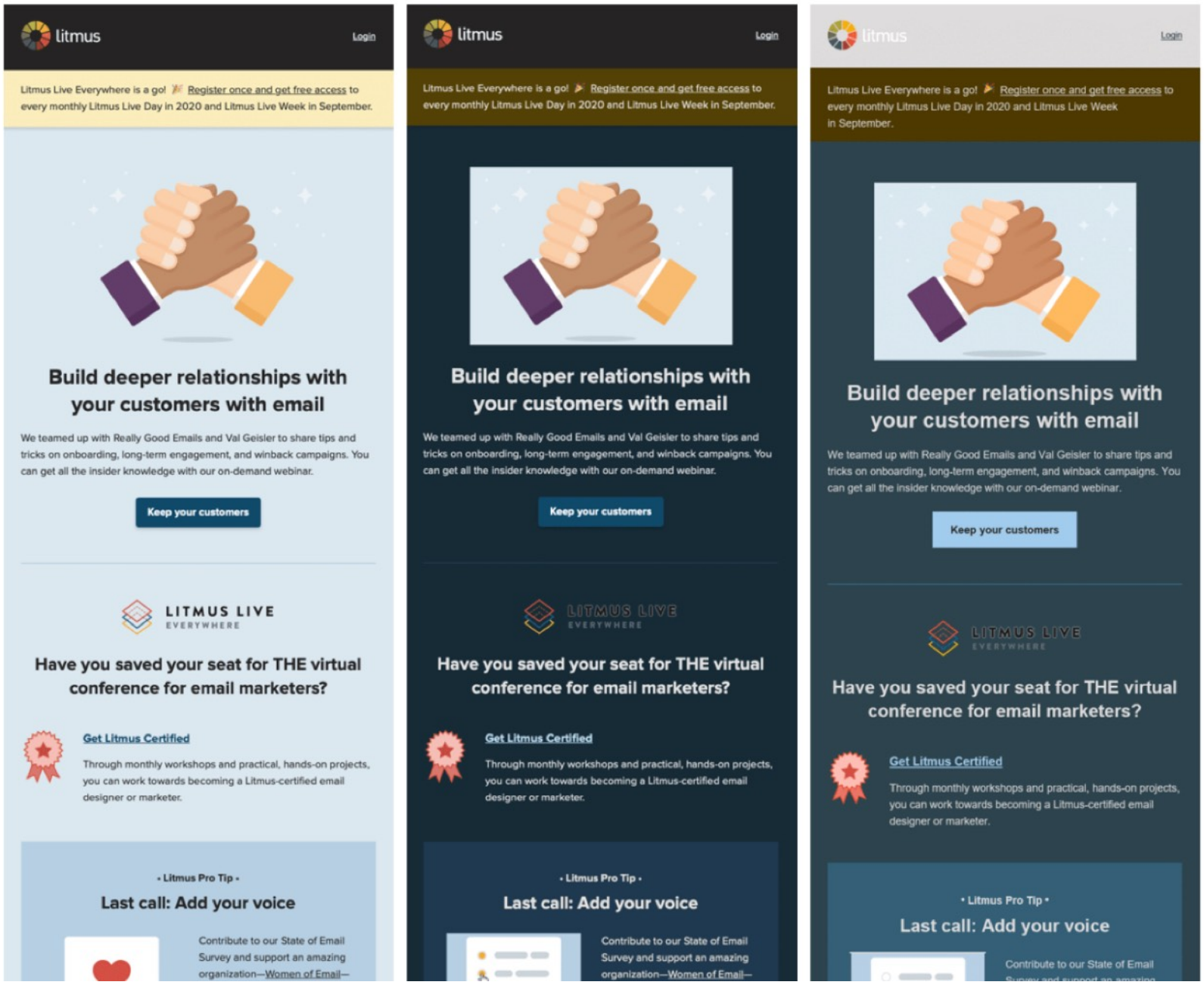
Regardless of the reason dark mode is turned on, it's important to account for it in our designs so that we can respect the user's preferences.

Dark Mode Support in Email

Just like literally everything else in email, dark mode support varies from email client to email client. Assuming you're doing nothing to support dark mode in your email code, one of three things can happen to your email when a user has dark mode enabled:

1. Nothing, your email will be displayed in its normal light version. This is usually a good experience, if jarring when you're using bright background colors.
2. Partial color inversion, where some of the colors are automatically changed to imitate a dark mode theme in your email. Light backgrounds become dark, dark text becomes light, but dark backgrounds with light text remain as designed.
3. Full color inversion, where all of the colors are swapped for their complimentary version.

While both options 2 and 3 are intended to enable dark mode in some capacity, the color choices that are chosen on your behalf are rarely flattering. They're functional, but usually off brand for most senders. Here's a good example taken from Litmus's excellent **Ultimate Guide to Dark Mode for Email Marketers** from **Alice Li**:



On the left is the original email. The middle version is Outlook.com doing a partial color inversion. The version on the right is Outlook 2019 on Windows doing a full color inversion. Note the header of the email, where the white logo is almost impossible to see when the dark header color is inverted.

Our goal is to use code and some defensive design techniques to make the best experience possible users. When we do, we can create better, more fully-controlled dark mode experience in clients that have some support for dark mode, including:

- Apple Mail on Mac and iOS
- Microsoft Outlook on iOS and Android
- Outlook 2019 on Mac
- Superhuman

- Hey.com

The way we do that is by using **1)** toggling dark mode support in our code, **2)** coding dark styles in a media query, and **3)** designing defensively (especially images) for when we can't control the situation.

Toggling Dark Mode

We've actually already seen how to toggle dark mode support. Remember this basic boilerplate email document?

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-
scale=1 user-scalable=yes">
  <meta name="x-apple-disable-message-reformatting">
  <meta name="color-scheme" content="light dark">
  <meta name="supported-color-schemes" content="light dark">
  <title></title>
  <style>
    :root {
      color-scheme: light dark;
      supported-color-schemes: light dark;
    }
  </style>
</head>
<body>
</body>
</html>
```

The **`color-scheme`** and **`supported-color-schemes`** meta tags, along with the embedded **`:root`** styles, are what tell email clients with good dark mode support that we have code that'll work for dark mode users.

Including Dark Mode Styles

The next step is to tell the email client what styles you want applied to your HTML when dark mode is enabled. You do this by including a media query within the embedded CSS styles in the head of the email.

```
<style>
    :root {
        color-scheme: light dark;
        supported-color-schemes: light dark;
    }

    @media (prefers-color-scheme: dark ) {
        /* Dark mode styles go here. */
    }
</style>
```

The **`prefers-color-scheme: dark`** rule acts as a toggle within your CSS. The inline styles, or anything in the embedded style block outside of the media query, will be the default light mode styles. When dark mode is supported and the email client comes across the media query, though, any style rules within that media query will override your default styles.

What you include here depends on what you have in your email. Generally speaking, you'll want to pick inverted versions of background colors and text colors for your elements. When you override them in the media query, be sure to use the **`!important`** declaration so that it takes precedence. For example:

```
<style>
    :root {
        color-scheme: light dark;
        supported-color-schemes: light dark;
    }

    @media (prefers-color-scheme: dark ) {
        .container {
            background-color: #222222 !important;
            color: #fafafa !important;
        }
```

```
    }
</style>
```

Right now, the Outlook for Android app does some funky stuff with styles which means that your dark mode CSS won't show. If you need to support Outlook for Android, you will need to duplicate those dark mode styles and prefix them with the **`[data-ogsc]`** selector, like so:

```
<style>
    :root {
        color-scheme: light dark;
        supported-color-schemes: light dark;
    }

    @media (prefers-color-scheme: dark ) {
        .container {
            background-color: #222222 !important;
            color: #fafafa !important;
        }
    }

    [data-ogsc] .container { background-color: #222222 !
important; color: #fafafa !important; }
</style>
```

Design Defensively

Finally, we need to account for those situations when we don't control the full dark mode experience. The partial and full color inversions take care of things when it comes to the colors of our backgrounds and text, but doesn't do anything to the images in an email. That can lead to situations like this:

A transparent logo that looks great on a light background is almost completely lost when the background colors is inverted. There are two ways to combat this.

The first is to use a graphics application like Figma or Photoshop to add an outline or light shadow to the image that only appears when the background is inverted:

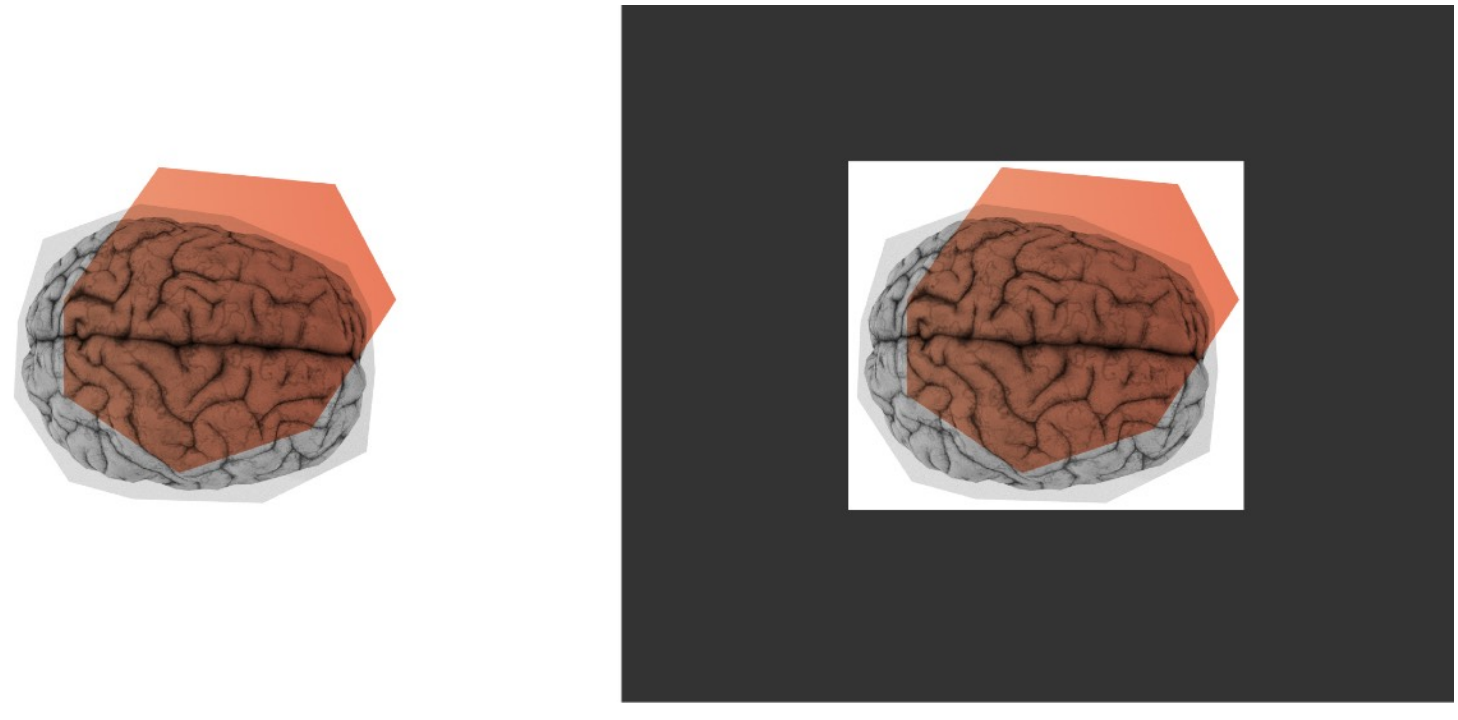
It's a little better, but still doesn't look great. A better option is to switch out the logo for one optimized for dark mode scenarios in the code of your email:

Essentially, there are two versions of the logo in the HTML of the email. By default, the light mode version is displayed and the dark mode version is hidden with CSS. For dark mode email clients, this is reversed in the media query:

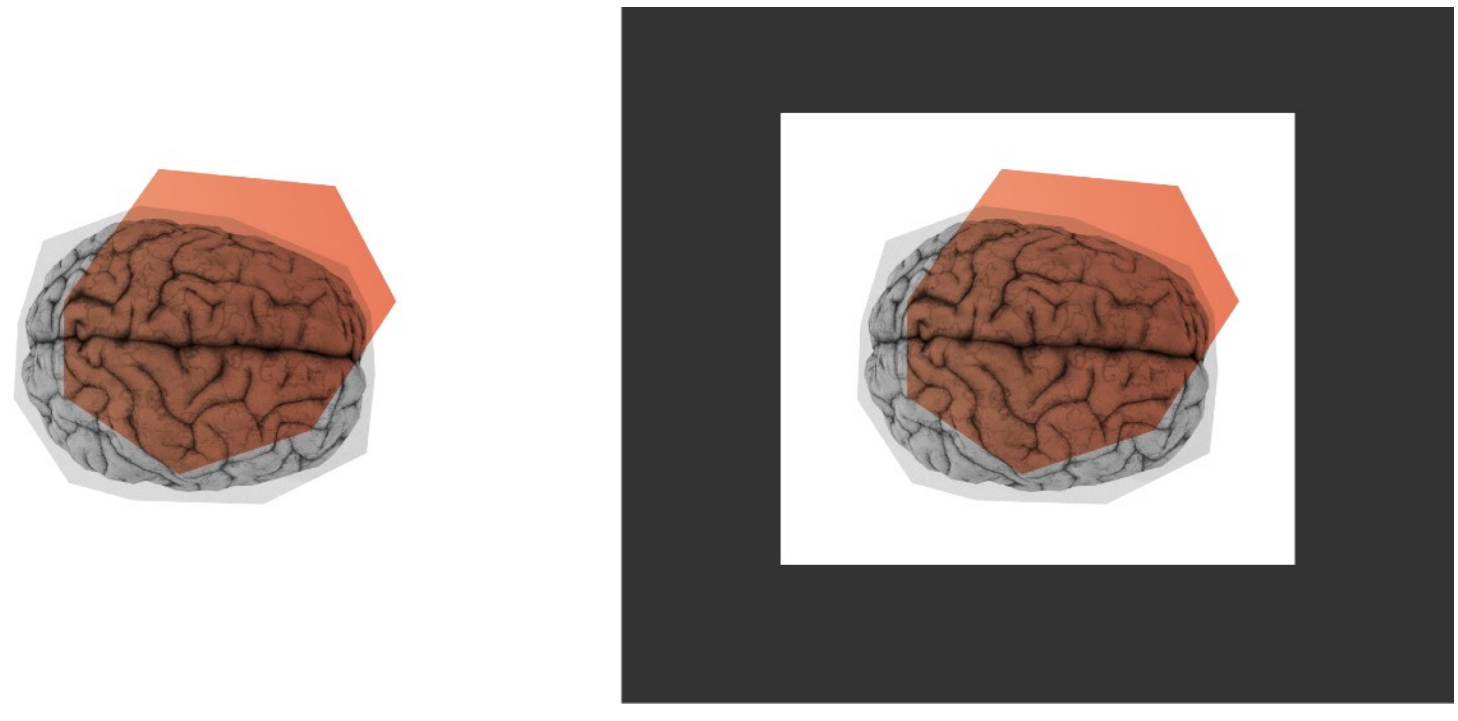
```
<style>
    :root {
        color-scheme: light dark;
        supported-color-schemes: light dark;
    }

    @media (prefers-color-scheme: dark ) {
        .light-mode { display: none !important; }
        .dark-mode {
            display:block !important;
            overflow: visible !important;
            max-height:inherit !important;
        }
    }
</style>

<img src="logo-light.png" alt="" class="light-mode">
<!--[if !mso]><!-->
<div class="dark-mode" style="display: none; max-height: 0px;
overflow: hidden;">
    <img src="logo-dark.png" alt="">
```

```
</div>
<!--<![endif]-->
```

This is useful for images with transparent backgrounds—especially logos and icons—but what about images without transparent backgrounds? Without a little planning, images can end up looking cramped when displayed in dark mode:



To avoid this, add some padding around the image in your graphics program so that it looks more intentional when displayed in dark mode:



Chapter Wrap Up

Technical accessibility is a lot to take in, especially if you're unfamiliar with coding email campaigns. To help provide a foundation on which to build (both your understanding and your own emails), I've put together a full HTML email template which incorporates all of the practices we discussed above. I've used HTML comments to point out key accessibility considerations, as well as a lot of the techniques discussed in **The Better Email On Design**.

Access the template here.

Before moving onto the testing phase, let's take a quick look at what you need to keep in mind when it comes to technical accessibility:

Do	Don't
Include a doctype.	Override user preferences.
Include language and text direction.	Forget language settings.
Keep content scalable for users.	Leave out role="presentation" on tables.
Use semantic markup.	Misuse semantic markup or ARIA.
Apply ARIA roles to enhance accessibility.	Forget to include the alt attribute on images.
Include proper alternative text depending on an images role.	
Support dark mode to the extent you can.	

After you implement all of those optimizations, all that's left is testing your email in the real world. That's the subject of the final chapter.

Chapter 5

Testing Accessibility

OK, we've covered a lot in the hopes of building more accessible HTML email campaigns for our subscribers. But how can we know that we've succeeded?

The answer is simple (at least in theory): **We test our campaigns**.

The goal is simple. We want to test our emails for the three areas which we've focused on:

1. Writing accessible copy
2. Visually designing an accessible experience
3. Coding technically accessible emails

In practice, all three are a bit harder than just saying, "Oh, I tested my email." There are a variety of tools that will help us test each area, but they range in complexity and none present a truly comprehensive view of what a disability is actually like for the person experiencing it. Everything is an approximation.

Still, they are helpful in letting us see and hear our emails in a different way so that we can identify areas for improvement.

While some tools listed in this chapter are automated, the best accessibility testing will always be manual. The more you can do to put yourself in the shoes of the subscriber (regardless of their abilities), the better equipped you'll be to address their needs. Manually testing and experience your email campaign is the best way to do this.

Let's get started.

Testing Copy

The first step in the testing process is to ensure your copy is readable and accessible. This should come before you start designing your email, since changes to copy will force you to update the design once you start creating it. You want to limit those revisions and give your designers (or yourself) the actual copy to work with to help speed up the design process. So this testing phase will come during and immediately after the writing process.

Fortunately, accessible copy is the easiest to test.

If we look back on our recap from the copywriting chapter, you can see that testing will mostly consist of reading through your copy to get a feel for three things:

1. Whether or not your campaign has a clear goal.
2. Whether or not you're using plain language.
3. Whether or not you're using descriptive, actionable links.

All of the other guidelines in the dos and don'ts lists support those three things.

Do	Don't
Have a clear goal.	Try to do too much in any one email.
Use plain language.	Use lots of jargon or acronyms.
Make copy readable.	Use lots of clichés or complex metaphors.
Use shorter words, sentences, and paragraphs.	Write too much!
Translate and localize when needed.	Blindly automate translations.
Write descriptive, actionable link text.	Use "click here" or "more" for your links.
Write descriptive, actionable alternative text	Repeat words like "link" or "image" for screen readers.
Read ALL of your copy out loud.	Forget to read your copy out loud. Seriously.

So, the single best thing you can do is to read your copy out loud.

Seriously, it's the last point in the table above. And it says "seriously" there, too, so you know I mean business.

What you read in your mind is always going to sound different from when those same words are read out loud. Verbalizing them allows you to actually hear how they'll sound and, more importantly, whether or not they sound natural and easy to follow. Your brain is capable of jumping through all kinds of hoops to make sense of words on a screen. You mouth? Not so much.

You want to literally pull up your copy doc (it's in your email brief, remember?) and read it out loud, either to yourself or someone else. If you're shy about it, close your office door or go somewhere by yourself. Even if you are shy, it might be worthwhile to screw up a little courage and read it to someone else, as they'll be able to pick out problems that you might miss.

Ask yourself a few things while you're reading your copy out loud:

- Do the headings and body copy flow together?
- Are any of the sentences too long? Do I have to pause in the middle of them?
- Does the order of the sections in the email make sense?
- Are there any words or acronyms that sound confusing or aren't defined?
- Is it clear what I need to do after reading the email?
- Could a kid or teenager read this and make sense of it?

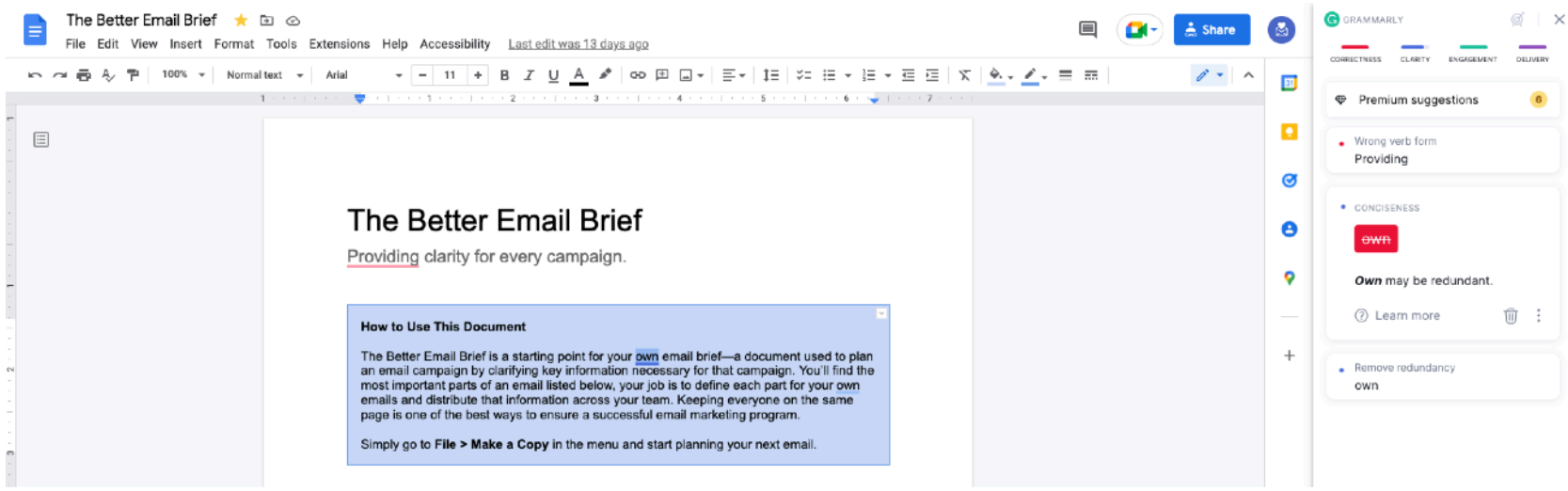
Depending on how you answer, you may need to go through and revise your email copy before moving onto the design phase.

Apart from reading copy out loud, there are some software tools that can help you identify issues—especially in regards to readability, reading level, and using plain language (but not about the clarity of your goal, unfortunately).

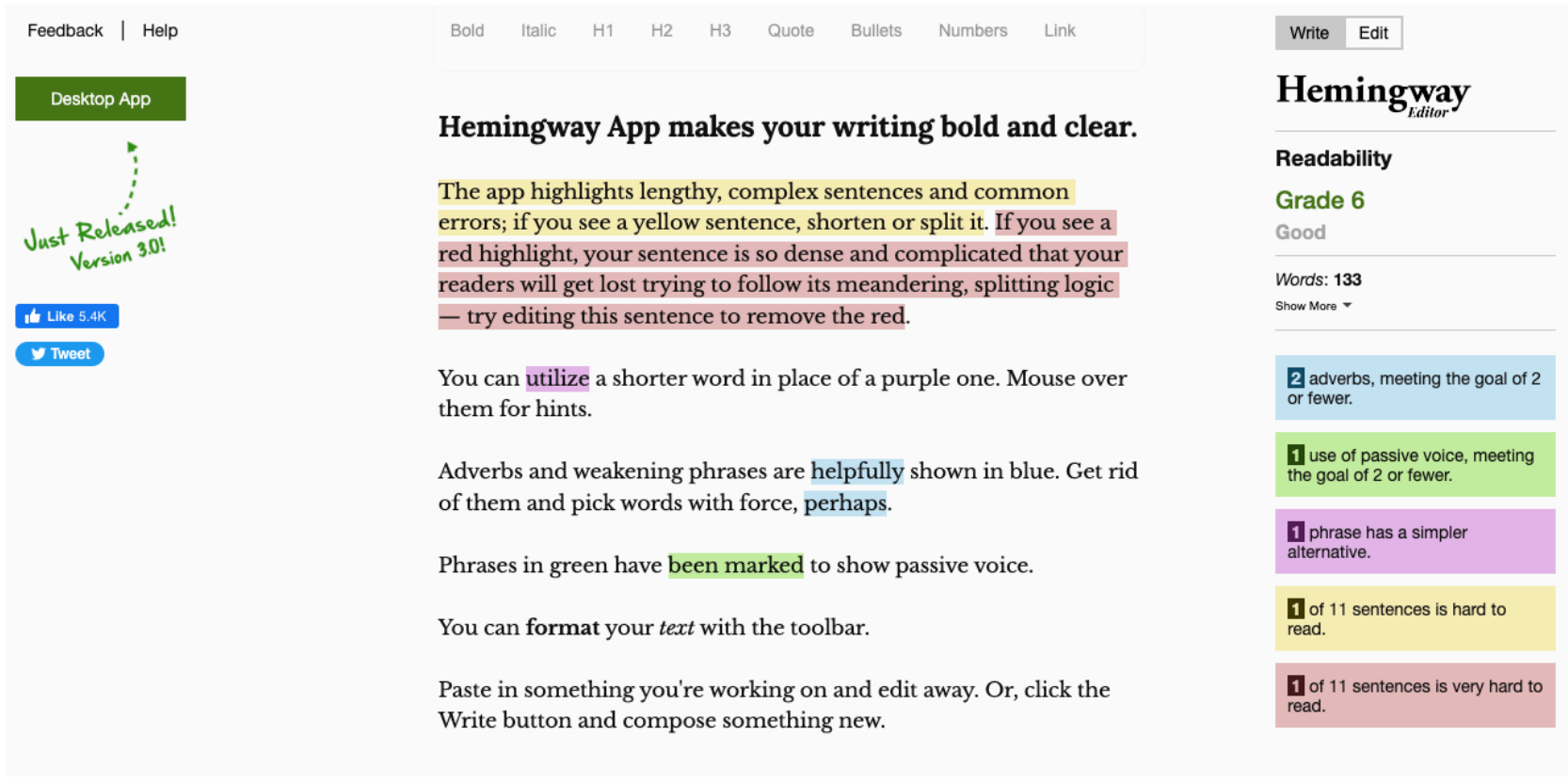
One of the most popular is **Grammarly**, which bills itself as a tool to help you "compose bold, clear, mistake-free writing." That basically describes accessible copywriting, right? Basically, Grammarly checks your copy either while you're writing

it or after the fact and identifies problems not only with spelling, but grammar, overall tone and clarity, formality levels, fluency, politeness, and even plagiarism. It's a comprehensive tool.

Grammarly is available as both a free and paid tool, and has a web interface as well as desktop apps for both Windows and Mac. Perhaps more importantly, it integrates well with a massive number of tools—many of which are commonly used to write email copy—including popular browsers, Microsoft Office, and Google Docs. In Google Docs, for example, Grammarly is available as a sidebar that identifies issues as you work on the doc:



Another tool that takes a similar approach is the [Hemingway App](), named after the author known for his succinct lines. Hemingway is available as both an online editor and desktop apps.



Like Grammarly, it identifies issues with your writing to help you clarify and simplify your writing. One of the things I like about Hemingway is that it makes the readability level of your writing obvious in its interface, along with all of the usual suggestions. This can help you get a general sense of how easy it will be for your subscribers to understand your message.

If you want to dig deeper into readability stats, the **Readability Analyzer** allows you to paste in your copy and see stats like:

- Number of sentences and words per sentence
- Percentage of difficult words (lower the better)
- Flesch Reading Ease score
- Gunning Fog Scale level
- Flesch-Kincaid Grade level
- Dale-Chall Score
- And more

While none of these is meant to be the end-all-be-all of determining your email's readability and accessibility, all of them can be useful for understanding how easy or difficult to read your email is. Combined with reading it out loud yourself, and automated tools like Grammarly or Hemingway, readability analyzers can be helpful in creating more accessible copy.

There are a bunch of variations on all three of the tools above, some free, some cheap, some expensive, but those are the three which I've found to work well for most use cases.

Remember, though, nothing beats reading your email out loud. Or having someone read it out loud to you. Or a combination of the two. Hearing your copy spoken will give you the best experience for testing the accessibility of your copywriting. It's a close analogue to the screen reader experience, which we'll look at when it comes to testing the code of an email campaign.

Testing Visual Designs

The next step of the testing process comes during and immediately after designing an email campaign, which is commonly done in a graphics application like Figma, Photoshop, Sketch, Adobe XD, or Adobe Illustrator. However, some designers create emails entirely in code (that's how I do it), so some of these steps will take place alongside testing the code of a campaign.

Let's look back at the dos and don'ts of accessible visual design:

Do	Don't
Understand the range of disabilities that affect how you design an email.	Mistake your abilities for everyone else's.
Create a strong email hierarchy to call attention to important information and make scanning easy.	Make every element the same size and color.
Use size, color, position, and spacing to reinforce that hierarchy.	Rely on color alone to convey meaning.
Use high contrast for low vision users.	Forget to test your contrast ratio against WCAG accessibility guidelines.
Use color effectively to convey meaning.	Use choose text alignment for it's aesthetic symmetry.
Write descriptive, actionable link text.	Create overly complex, confusing layouts.
Create readable text by combining font size, line length, line spacing, and text alignment.	
Layout content for to aid accessibility.	

The first "do" is all about understanding the range of abilities different subscribers have, since that will impact what you need to focus on when testing visual designs. Since "visual" translates to sighted users (even ones with visual disabilities), that's what this testing phase will focus on. In particular, you will want to spend this time testing the following:

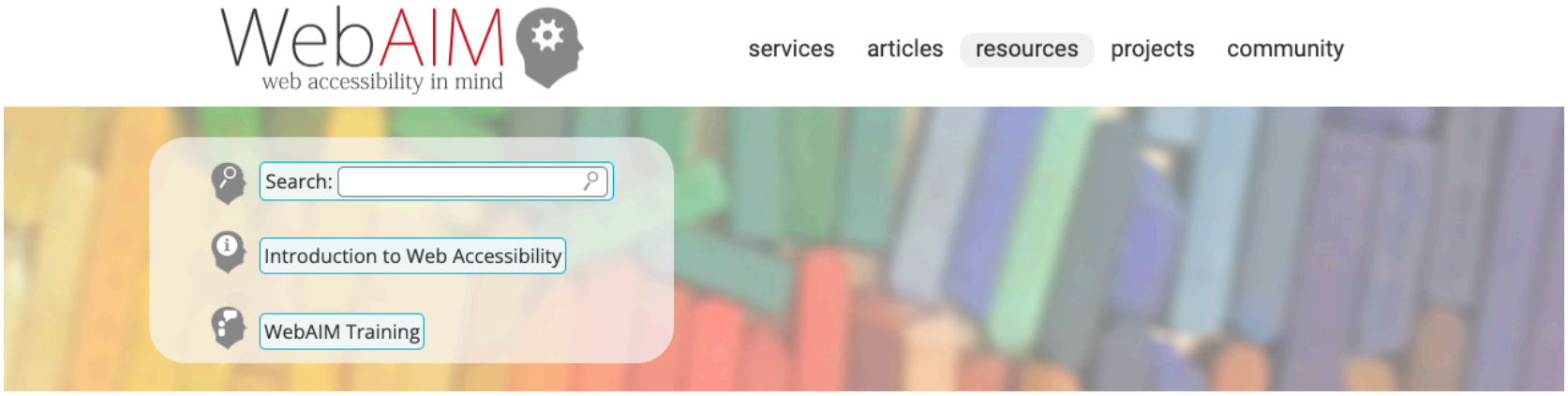
- Colors and contrast
- Hierarchy and spacing for scannability
- Size and spacing for visual disabilities

The tools used in this testing phase essentially do one of two things:

1. They test for contrast using automated math-based formulas
2. They simulate disabilities so that you can see how something looks for affected users

The first category of tool can be found all over the place. There are tons of websites that let you plug in your foreground and background colors to see their contrast ratio.

My favorite comes for the WebAIM group.

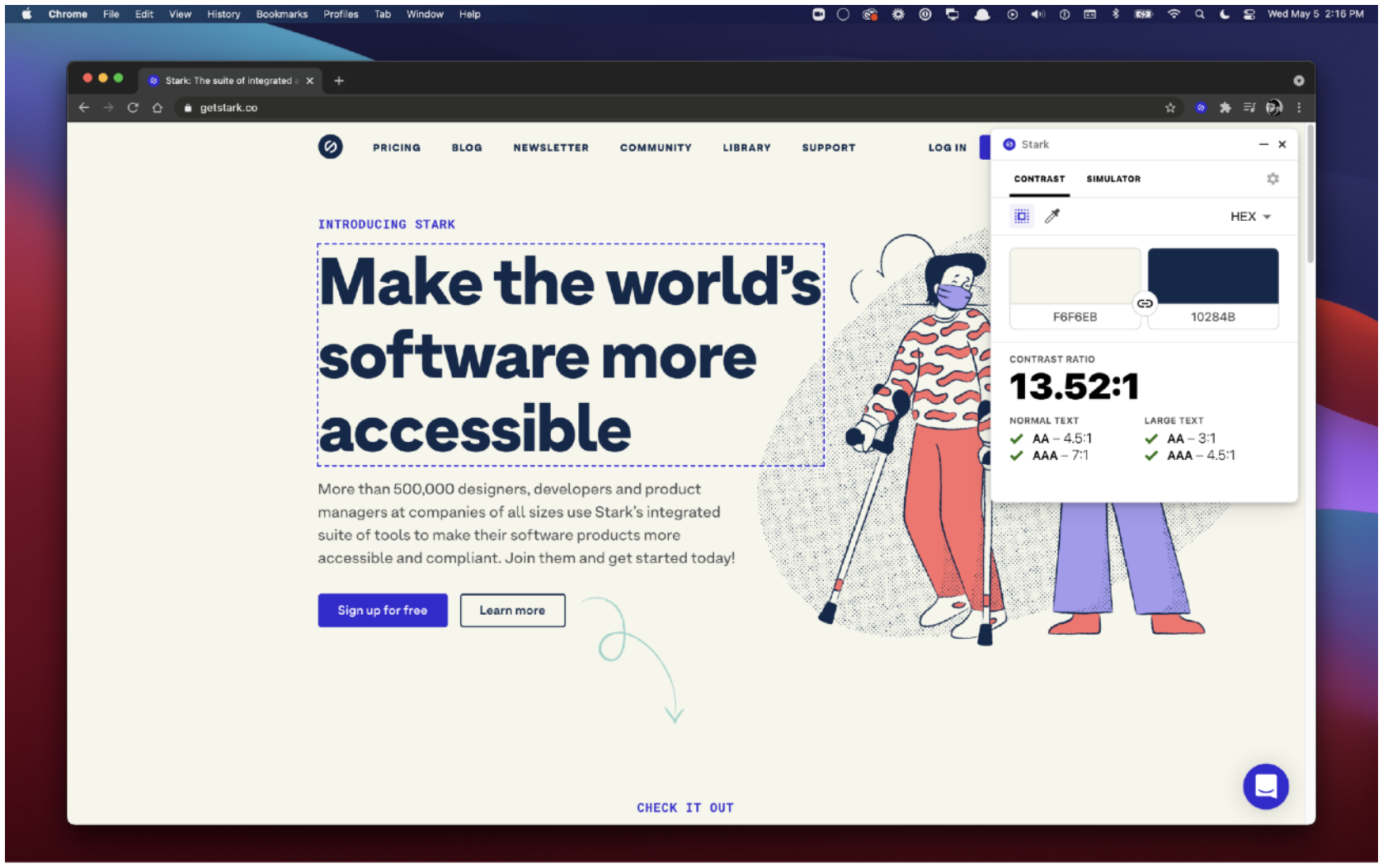


You can see fields for foreground and background colors, which can take a hexadecimal value, or you can use the color picker or sliders to define your colors. When updating, the contrast ratio, examples below, and WCAG ratings will all dynamically update, allowing you to quickly see how accessible your color combinations are.

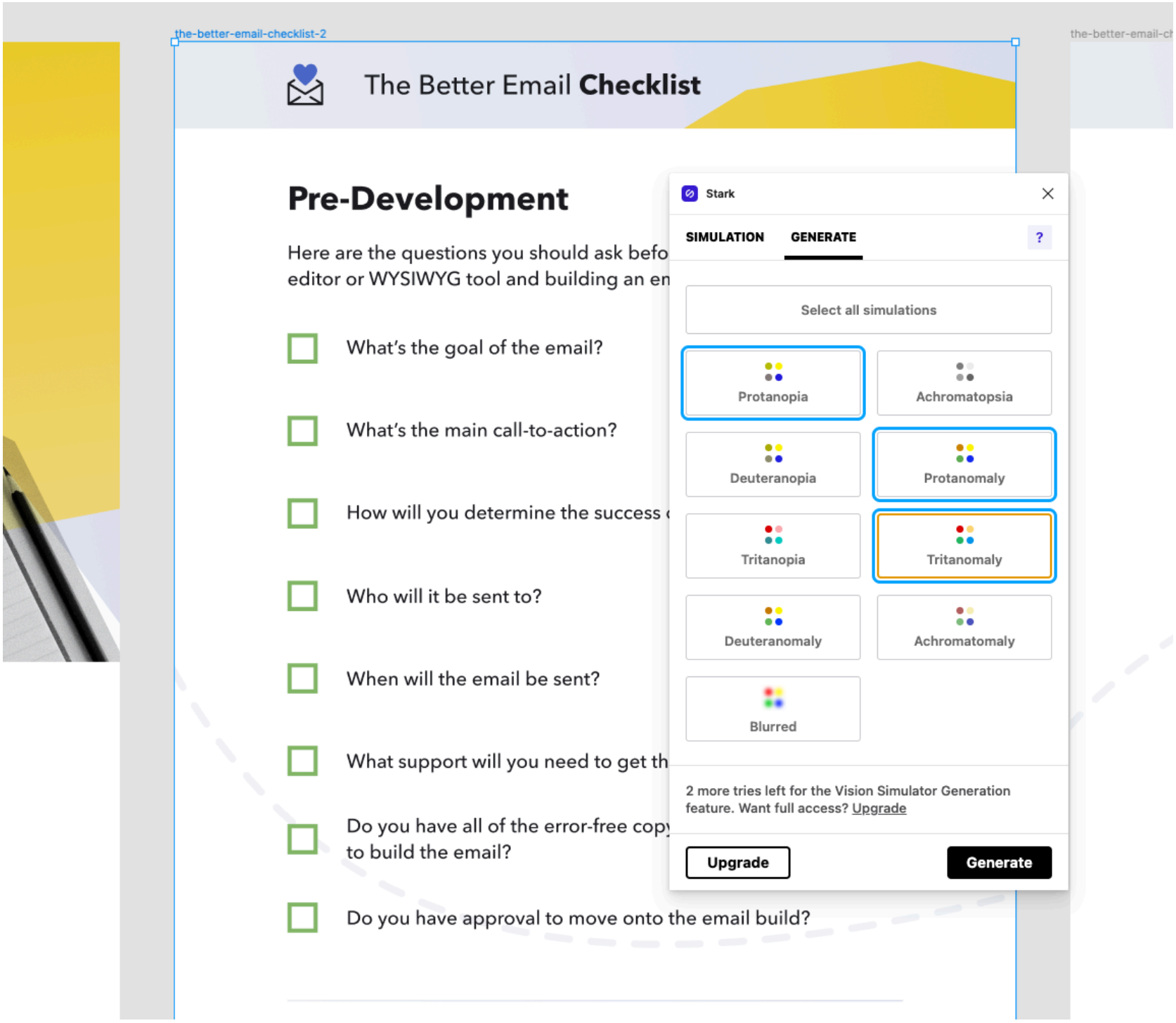
The major downside here is that it takes time to manually input all of the foreground and background colors in your email. And, as a general rule, email marketers, designers, and developers are short on time.

That's where plugins for your graphics and code editors as well as web browsers come into play. These tools are built into the same tools you use to design your emails, so that you can quickly check your colors and contrast ratios while you're setting them.

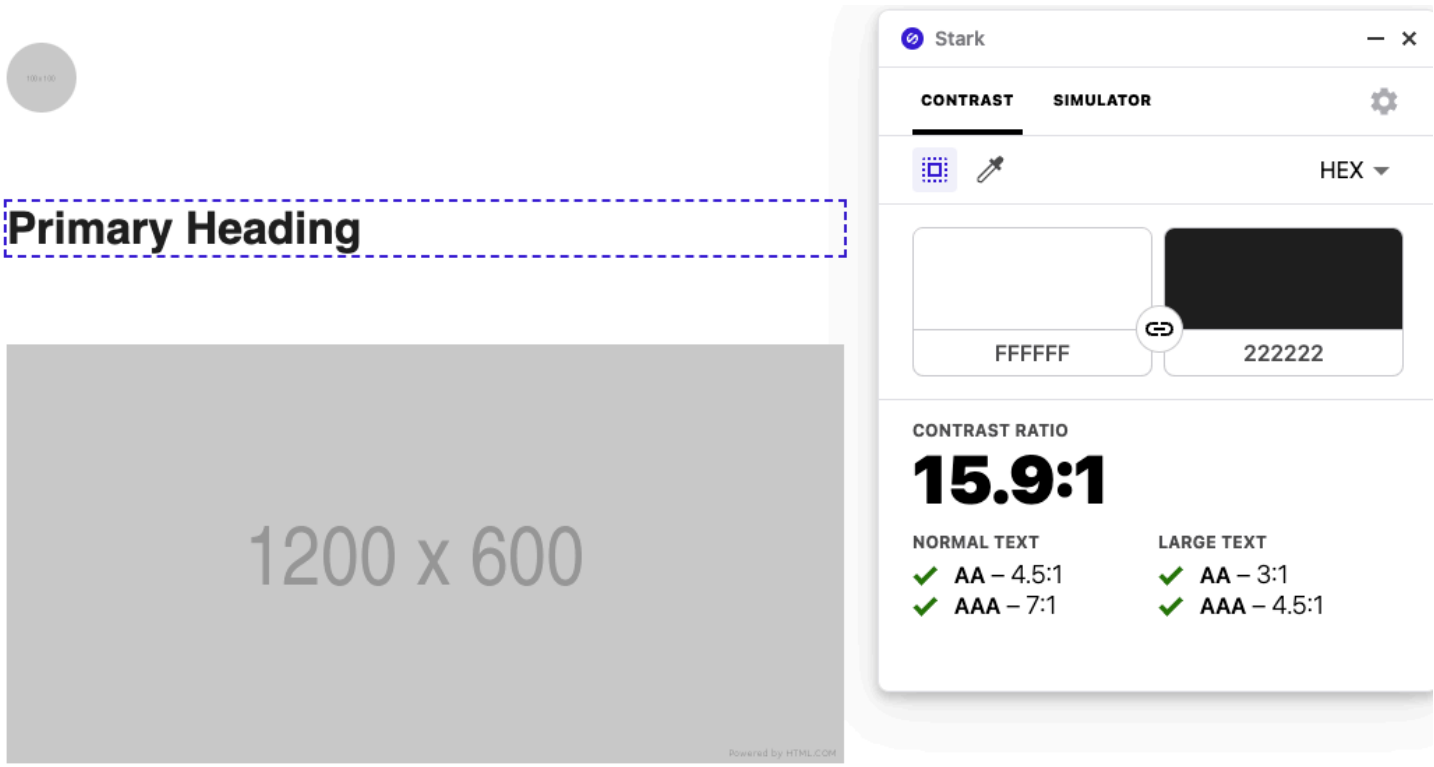
One of my favorites, which I've referenced before in this guide, is **Stark**. Stark is an extension for Figma, Sketch, Adobe XD, and Google Chrome. Once installed, you can open stark in your tool of choice and see accessibility information displayed. Here's an example taken from the Stark website:



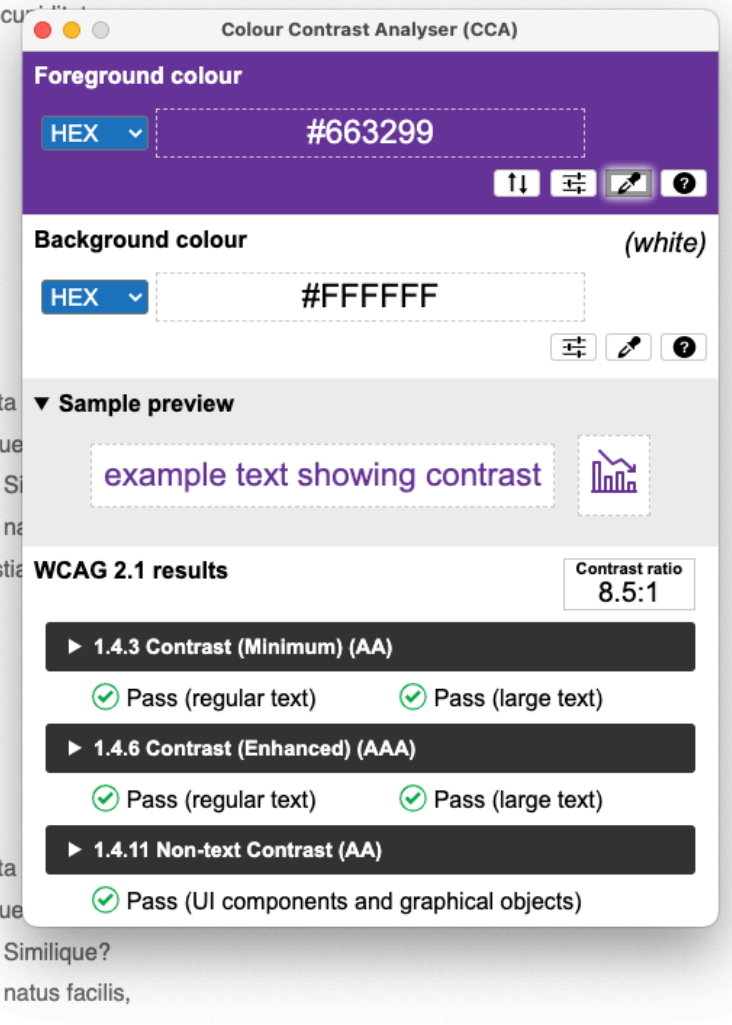
Within your design tool, you can select different elements and test their contrast. You can also simulate different color deficiencies, test the focus order of elements, and even define and test landmark roles.



I'm particularly fond of using Stark as a browser extension in Chrome, since it's easy to select and test different elements of an email to see their contrast ratios, or toggle different color simulations and see them applied to the whole email.



Another tool I like is the [Colour Contrast Analyzer from TPGi]. It's a free downloadable app for Windows and Mac users that allows you to use an eye dropper tool to pick colors on your screen to test contrast.



This is helpful when you're working on an email locally. Most browser extensions or online tools require a URL or hosted version of an email to work. But when you're coding on your own machine, you don't have a hosted version. It's easier to pop this open and pick or input colors to test contrast while you're developing an email than it is to upload that email or send yourself a test campaign to get a hosted version every time you need to check.

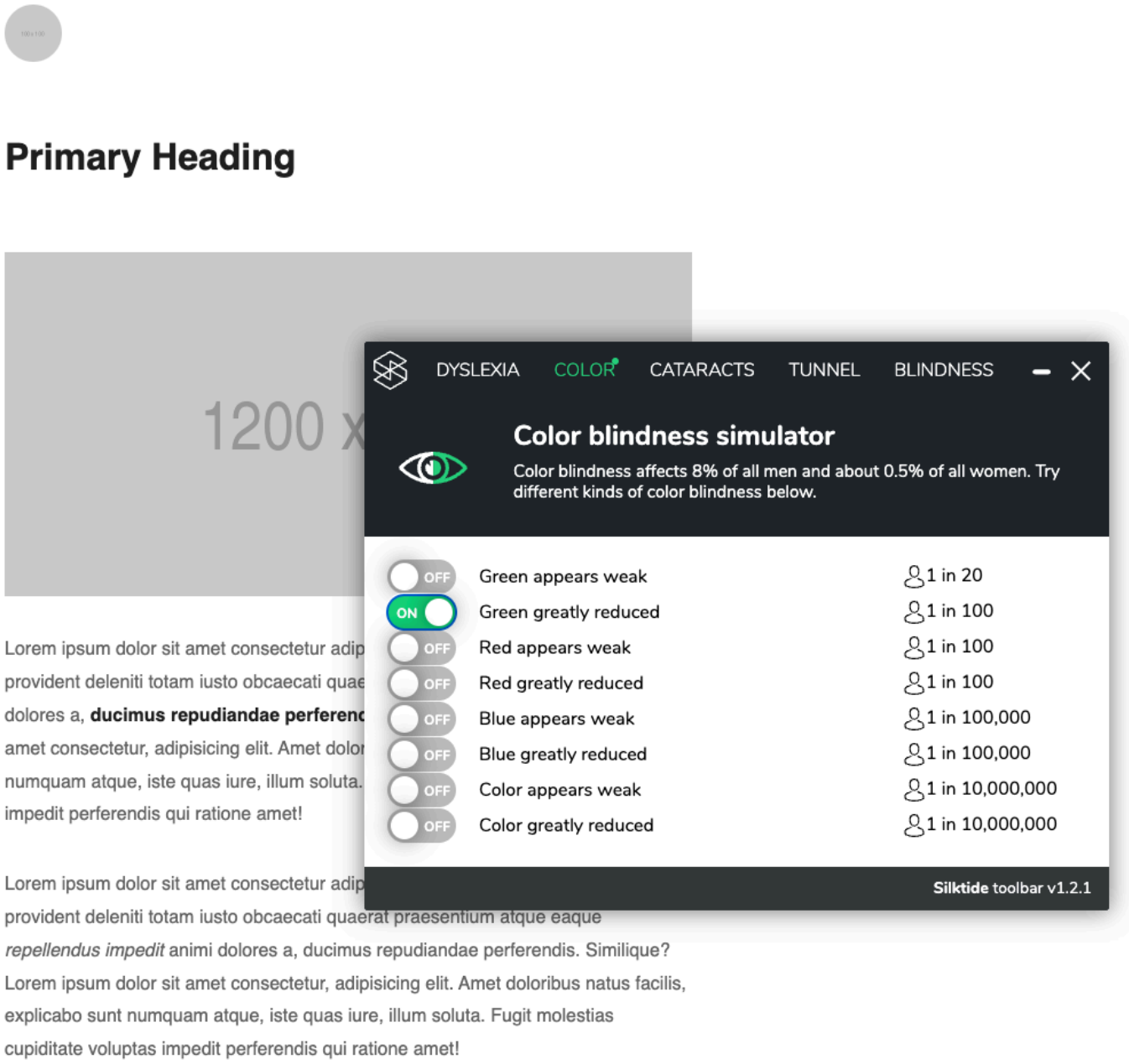
Depending on what tools you use to design emails, there are probably other options available for testing the colors and contrast in your emails. Just do a search online for "accessible color checker" to find the right one for you.

If you're designing from scratch and not adhering to a branding guide, you can also take advantage of resources like the **Accessible Color Palette Builder** or **90 Combinations** to pick colors for your email design.

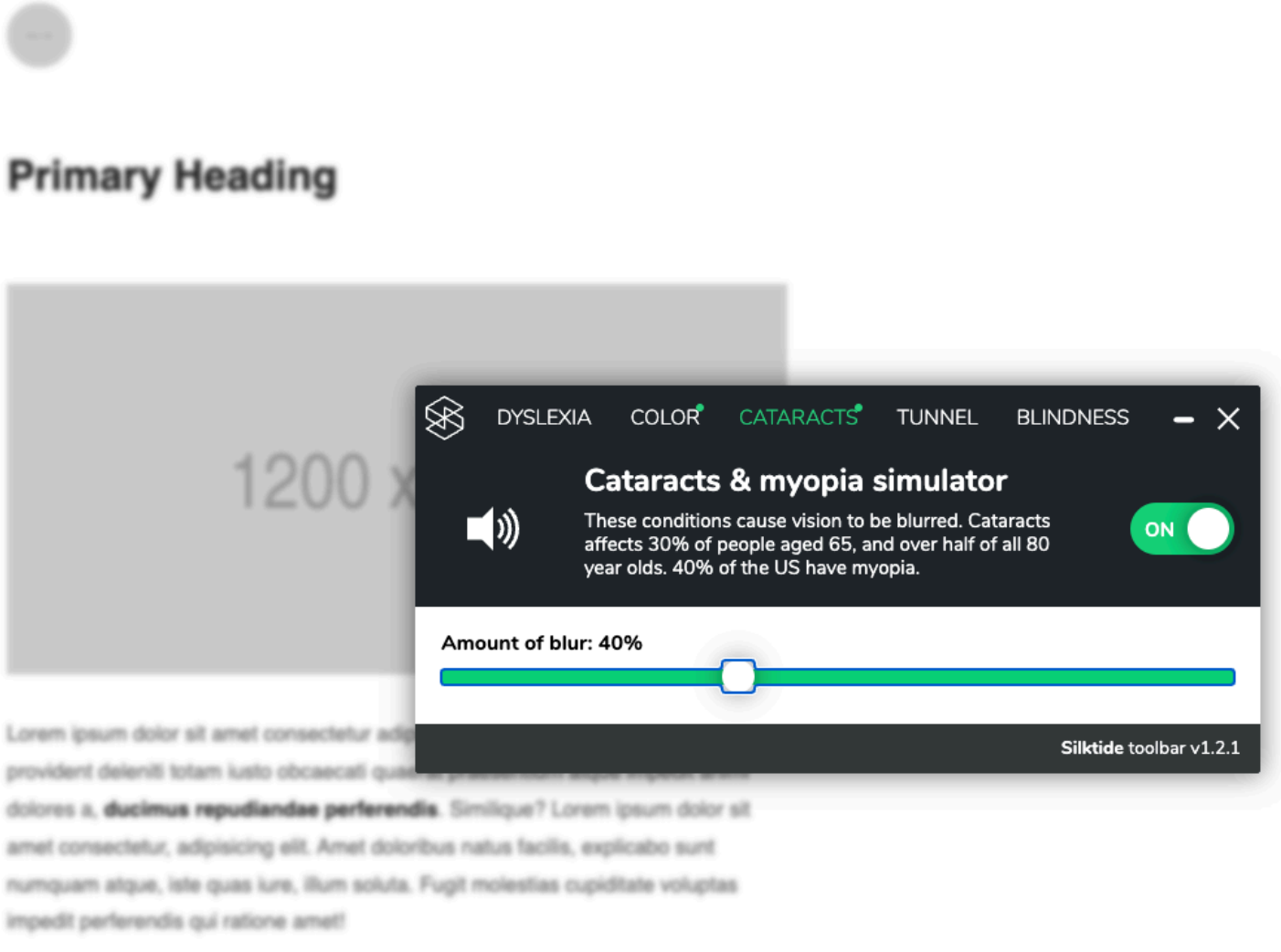
Regardless of what tool you're using, just adhere to accessibility guidelines to keep contrast high enough for users with limited vision to understand your content.

The second category of tools are used to simulate different visual impairments so that you can literally see what other people see (or a close approximation at least). Stark's color simulation modes is an example here, but there's one tool that I prefer to use and recommend to everyone: [The Silktide Toolbar for Google Chrome](). Silktide is an accessibility company that provides a lot of tools, from automated accessibility reviews to content audits. But my favorite tool of theirs is their free browser extension that lets you simulate a ton of different situations. Although it requires a hosted version of your email to work, the testing options are impressive.

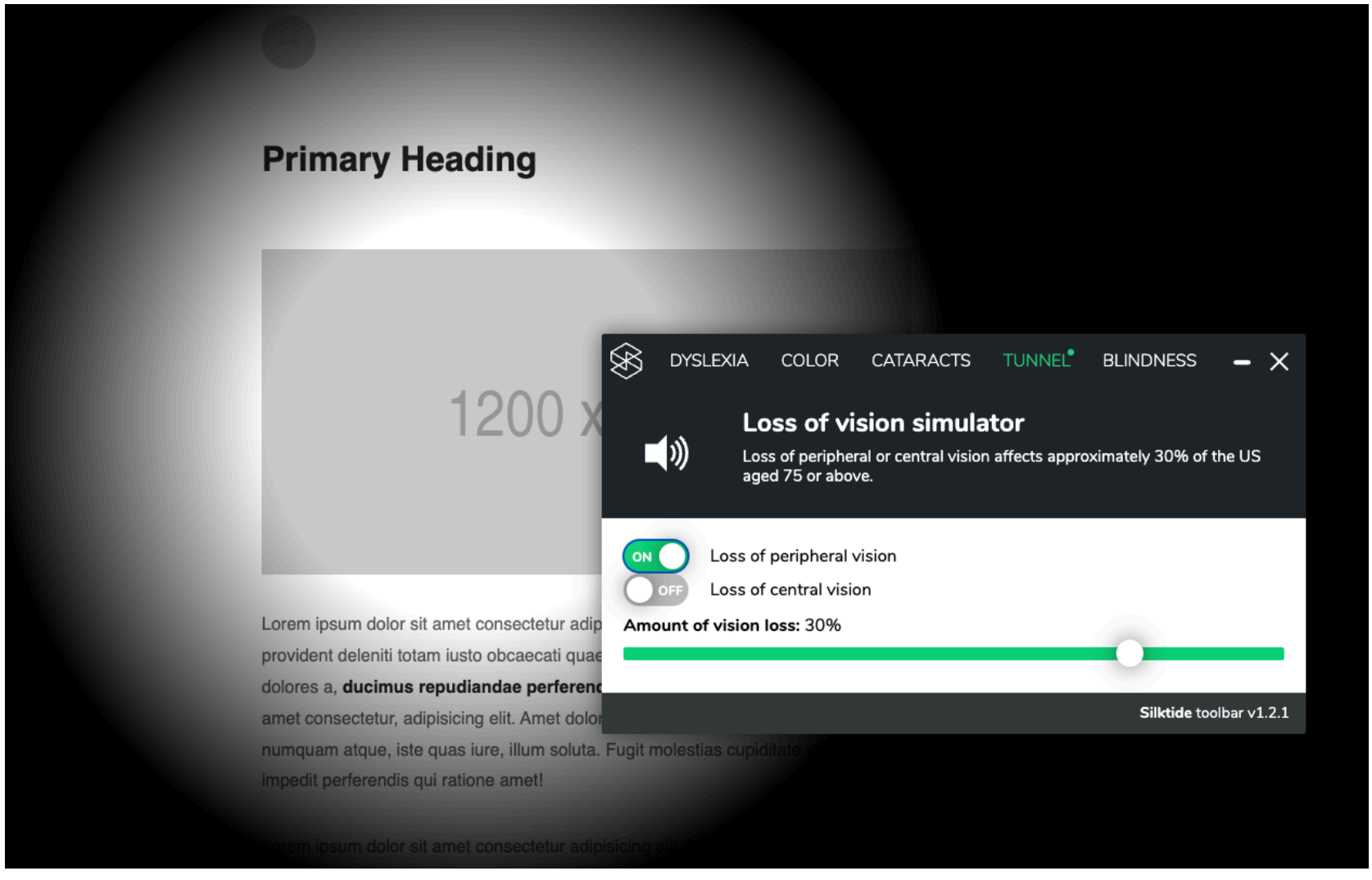
You can test different color deficiencies:



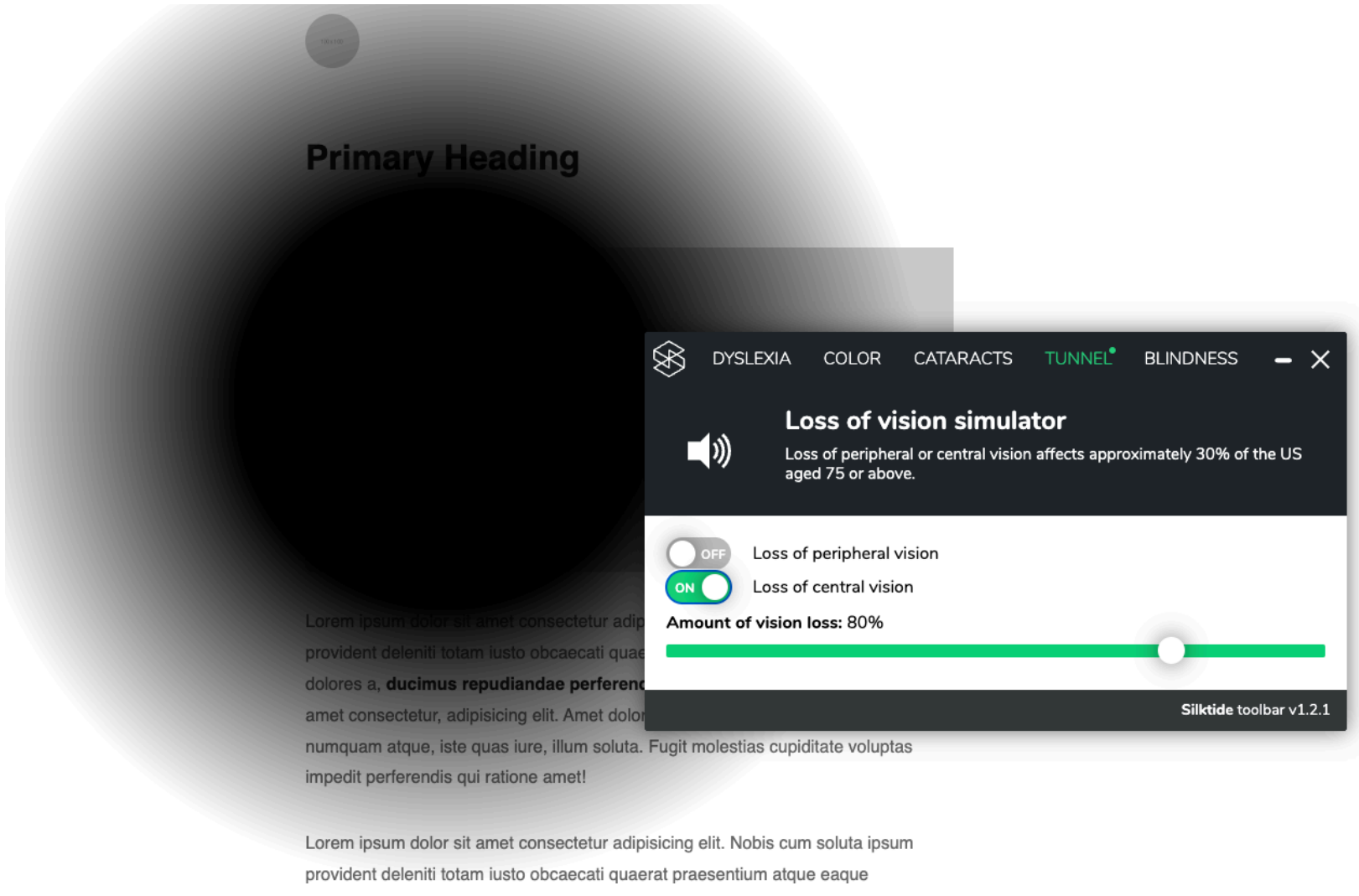
Or simulate cataracts or myopia, which results in blurring for people:



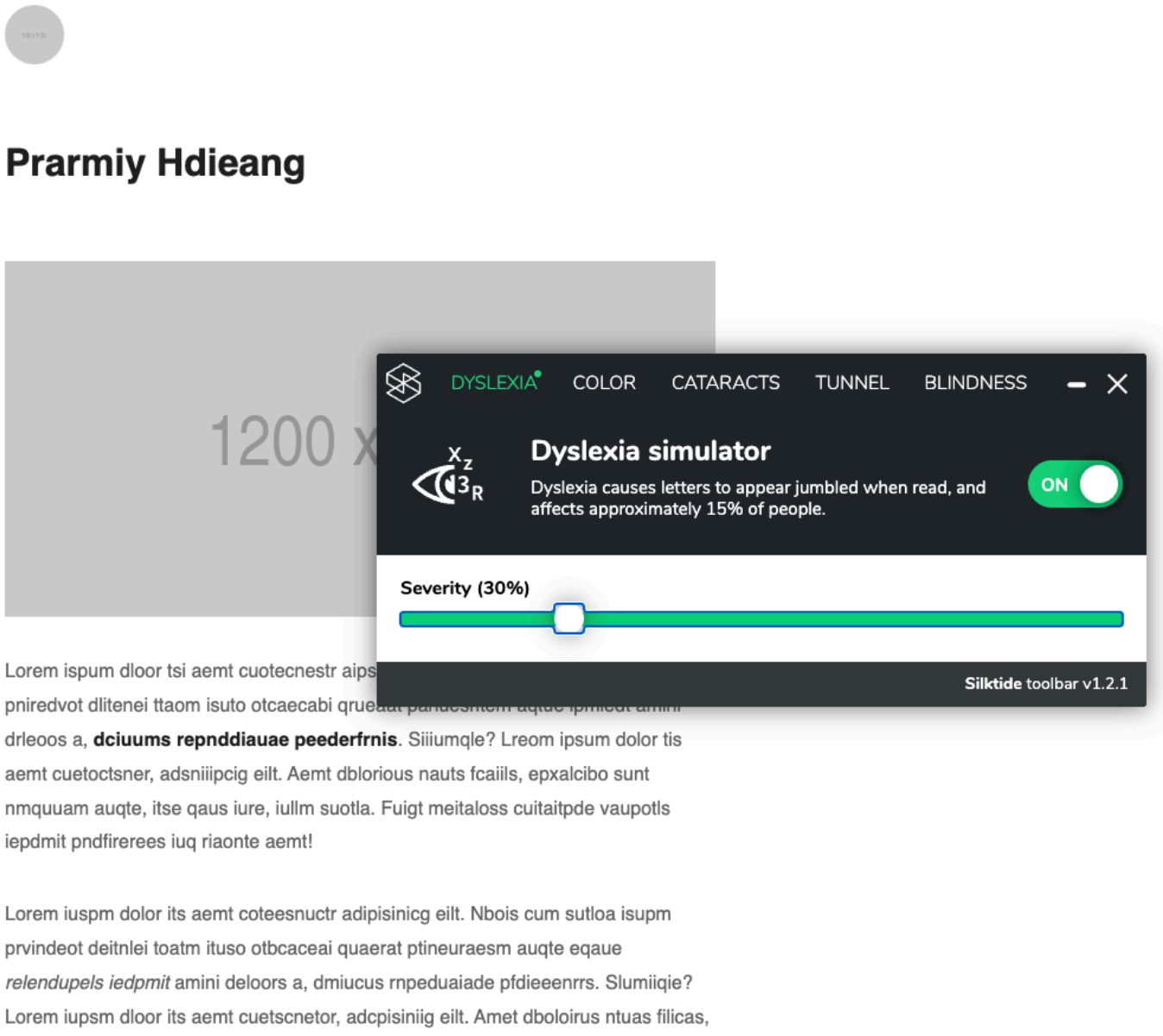
You can also test both loss of peripheral vision:



As well as central vision:



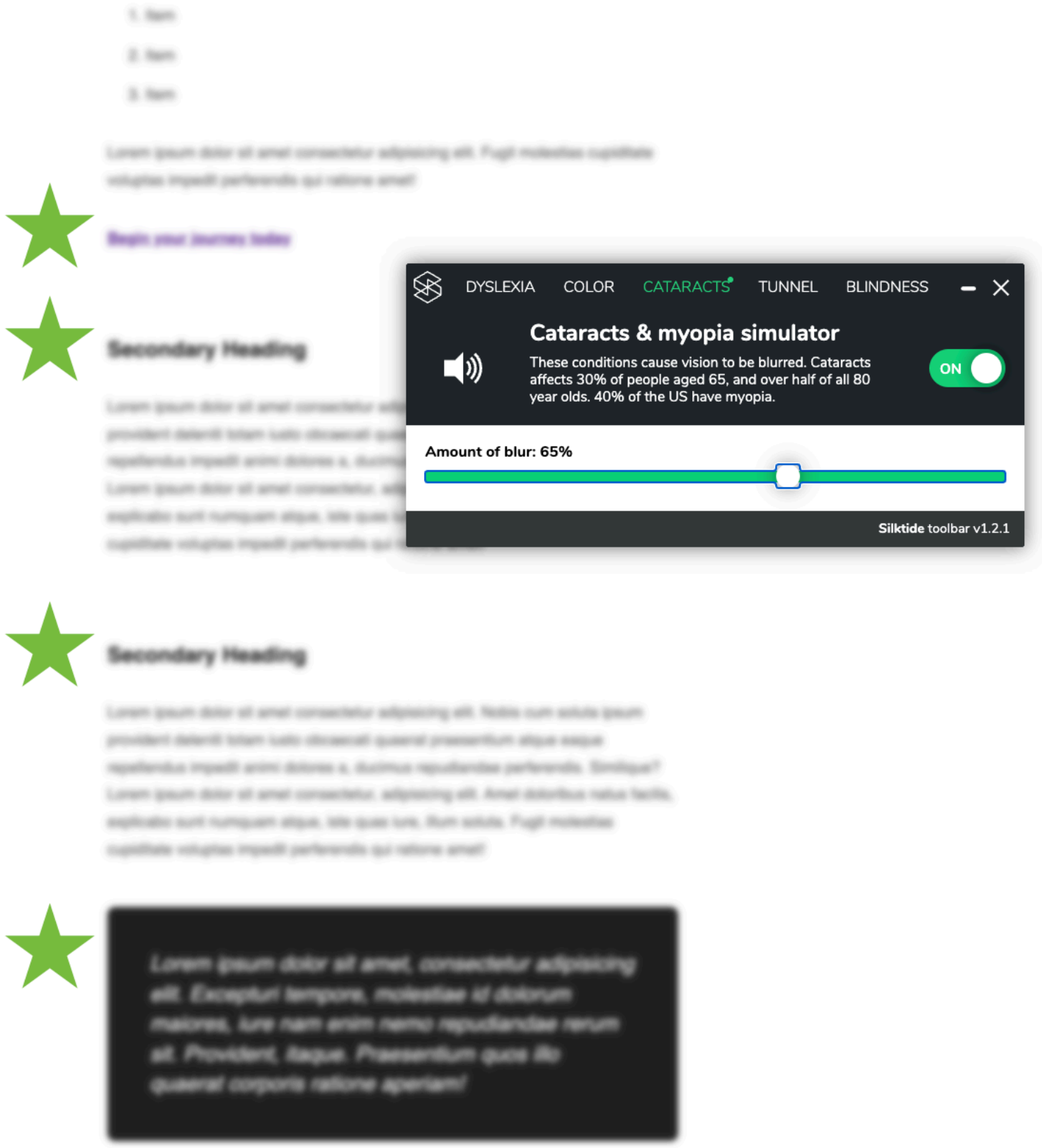
It even allows you to test cognitive disabilities like dyslexia, which affects how the brain interprets visual information. Notice how the letters are mixed up? It does that dynamically to your emails, allowing you to better understand the challenges faced by dyslexic subscribers:



We'll get to the "Blindness tab" in a bit…

After you test the contrast and simulate visual disabilities, you should ensure that a clear hierarchy and plenty of spacing between elements exist so that users can easily scan your content. This is especially important for people browsing and quickly scrolling on mobile devices, as well as users with issues like cataracts.

For testing cataracts, the Silktide extensions is my preferred method. Crank up the amount of blur and scan your email to make sure there are clear waypoints for the user's eyes, as well as that text (especially headings) are large enough and have high contrast so that they're readable even for low-vision users.



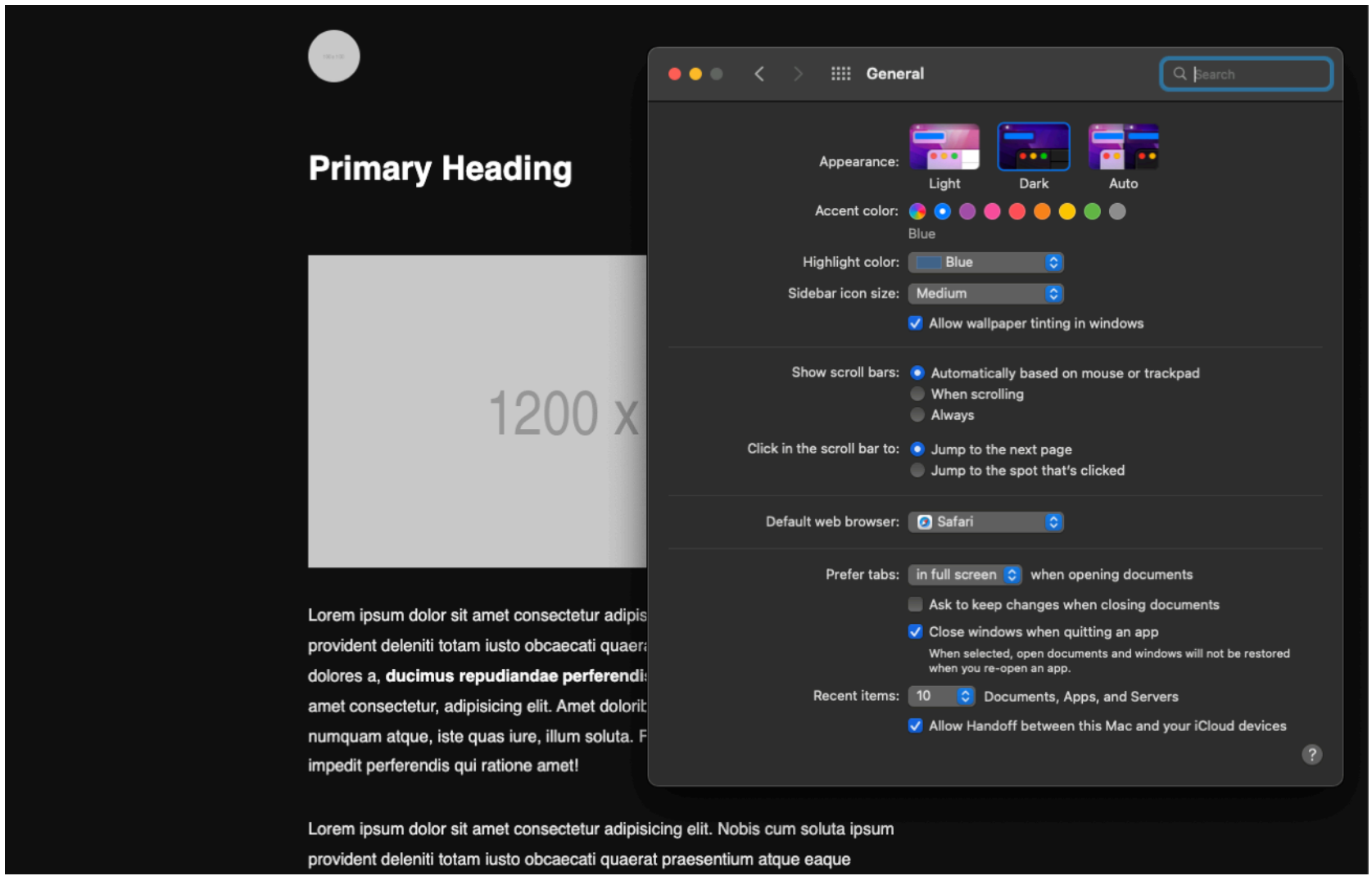
In the example above, I've starred the key elements in the hierarchy of the email. As you can see, they can quickly be distinguished even when vision is impaired.

For mobile users, the best way to test is by sending yourself a test email and scrolling around in it on your own device. You'll be able to tell whether or not elements are too small, too big, too cramped, or lack the contrast needed to keep the email scannable. As a bonus test, you can turn down the brightness on your own device to see how the email fairs in low light situations.

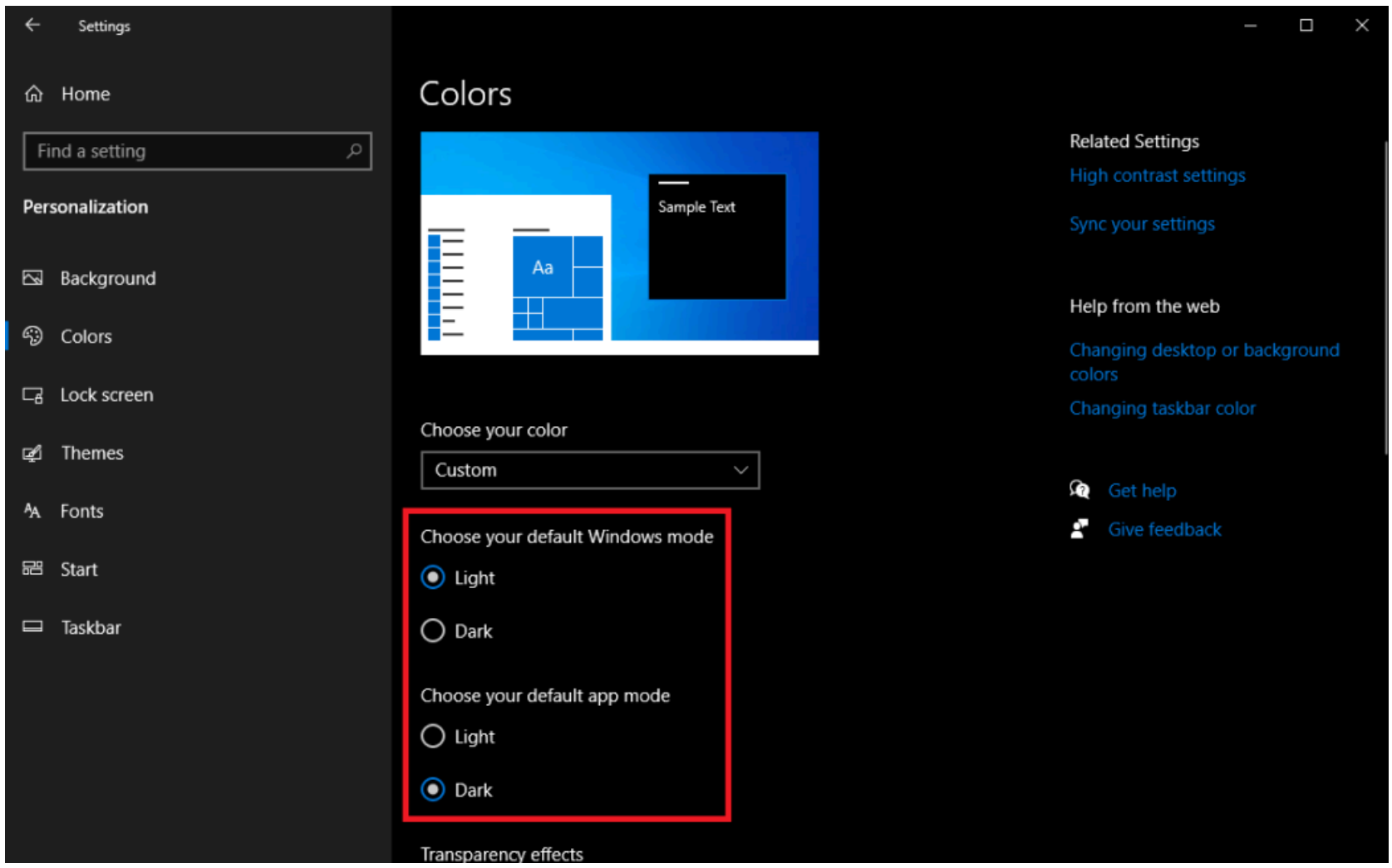
Similar to testing in low light conditions, we need to test emails in dark mode, too. Although we discussed dark mode as a technical consideration, it should be part of the visual accessibility testing phase.

Dark mode can be complex to test, depending on your email audience. While testing on your own device is relatively easy, subscribers use a variety of email clients that all support dark mode in different ways. Remember the partial and full color inversions? That's the stuff that's harder to test depending on the resources you have available.

The quickest way to test your dark mode email is by toggling dark mode on and off at the operating system level. On a Mac, go into **System Preferences > General** and then select the dark option:

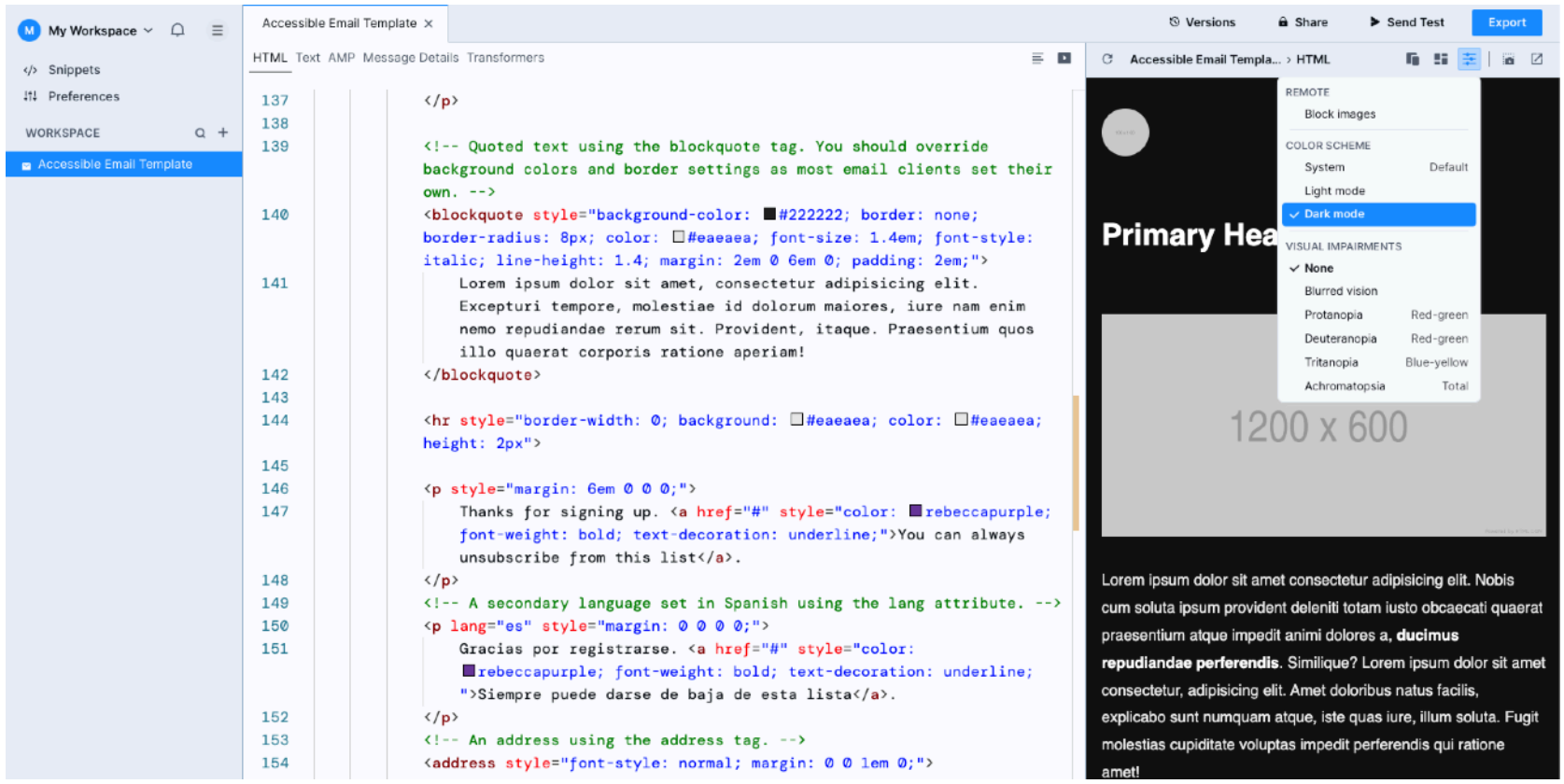


If you're on Windows, go into **Settings > Colors** and choose the dark option for both the Windows mode and app mode:



Now, open your email campaign in a browser and check whether or not your dark mode support is actually working. You can do something similar on mobile devices if you send yourself a test email.

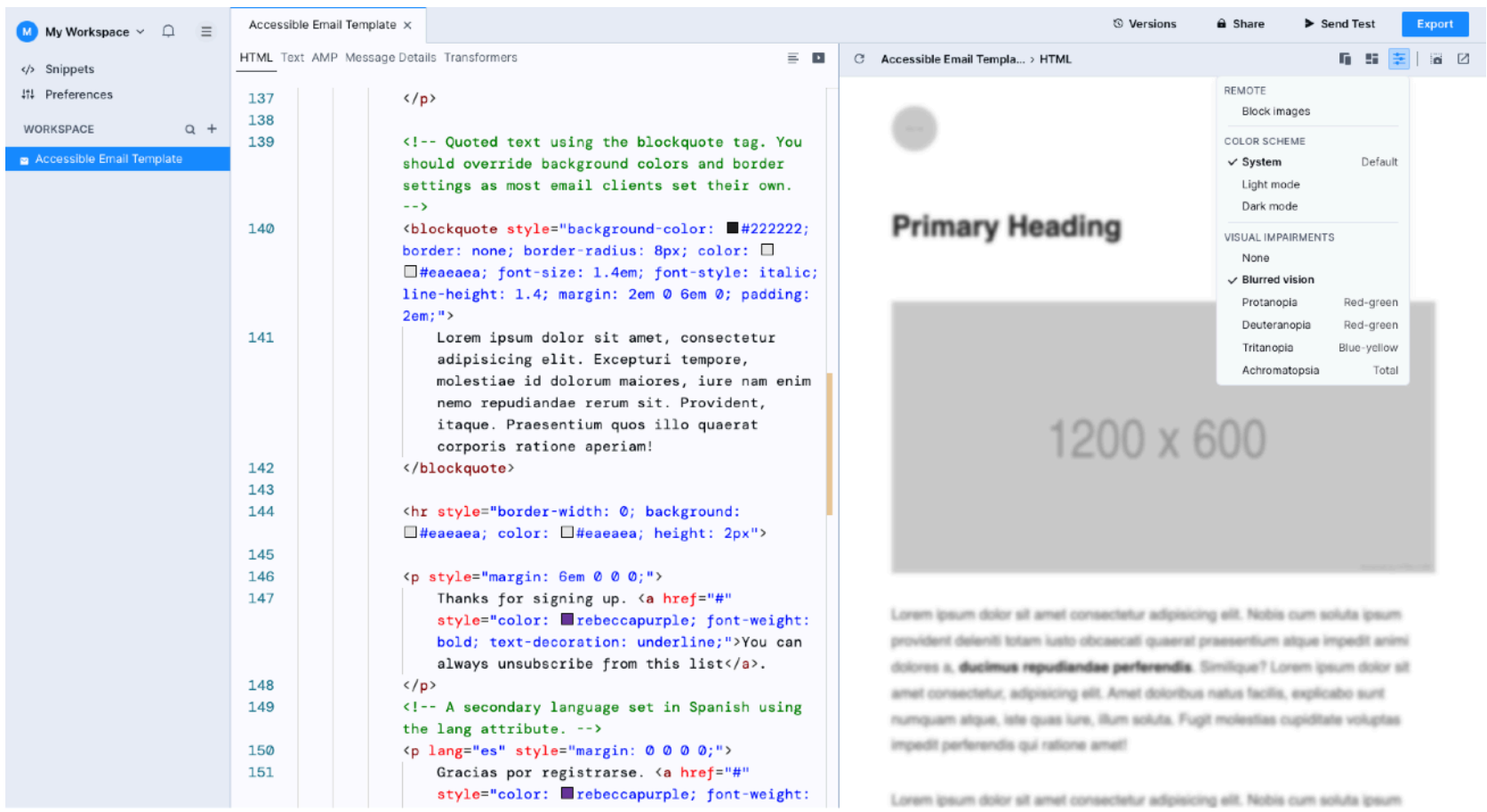
Another option is to use a code editor like **Parcel**, which has some amazing features built in. It's free to get started (and worth the money if you're coding a lot of emails) and includes an option for setting dark mode when you're editing an email:




```
137        </p>
138
139        <!-- Quoted text using the blockquote tag. You should override
           background colors and border settings as most email clients set their
           own. -->
140        <blockquote style="background-color: #222222; border: none;
           border-radius: 8px; color: #eaeaea; font-size: 1.4em; font-style:
           italic; line-height: 1.4; margin: 2em 0 6em 0; padding: 2em;">
141            Lorem ipsum dolor sit amet, consectetur adipisicing elit.
               Excepturi tempore, molestiae id dolorum maiores, iure nam enim
               nemo repudiandae rerum sit. Provident, itaque. Praesentium quos
               illo quaerat corporis ratione aperiam!
142        </blockquote>
143
144        <hr style="border-width: 0; background: #eaeaea; color: #eaeaea;
           height: 2px">
145
146        <p style="margin: 6em 0 0 0;">
147            Thanks for signing up. <a href="#" style="color: rebeccapurple;
               font-weight: bold; text-decoration: underline;">You can always
               unsubscribe from this list</a>.
148        </p>
149        <!-- A secondary language set in Spanish using the lang attribute. -->
150        <p lang="es" style="margin: 0 0 0 0;">
151            Gracias por registrarse. <a href="#" style="color:
               rebeccapurple; font-weight: bold; text-decoration: underline;
               ">Siempre puede darse de baja de esta lista</a>.
152        </p>
153        <!-- An address using the address tag. -->
154        <address style="font-style: normal; margin: 0 0 1em 0;">
```

This is handing if you don't want to constantly go into your system settings and update your display mode.

Parcel actually has a few accessibility simulations available, too, which allow you to simulate blurring (which isn't adjustable like the Silktide tool) as well as different color deficiencies:

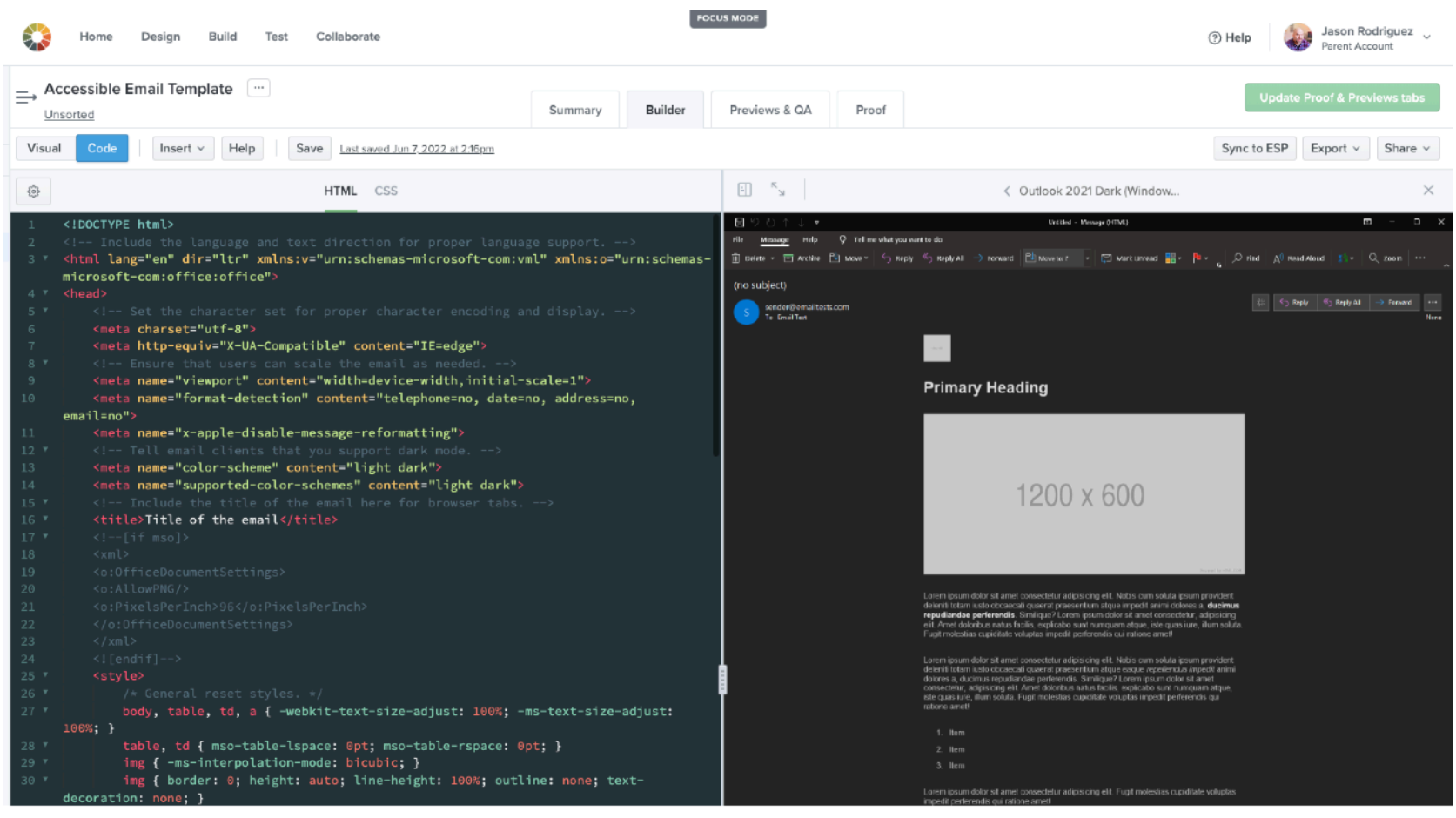



```
137        </p>
138
139        <!-- Quoted text using the blockquote tag. You
           should override background colors and border
           settings as most email clients set their own.
           -->
140        <blockquote style="background-color: #222222;
           border: none; border-radius: 8px; color:
           #eaeaea; font-size: 1.4em; font-style: italic;
           line-height: 1.4; margin: 2em 0 6em 0; padding:
           2em;">
141            Lorem ipsum dolor sit amet, consectetur
               adipisicing elit. Excepturi tempore,
               molestiae id dolorum maiores, iure nam enim
               nemo repudiandae rerum sit. Provident,
               itaque. Praesentium quos illo quaerat
               corporis ratione aperiam!
142        </blockquote>
143
144        <hr style="border-width: 0; background:
           #eaeaea; color: #eaeaea; height: 2px">
145
146        <p style="margin: 6em 0 0 0;">
147            Thanks for signing up. <a href="#"
               style="color: rebeccapurple; font-weight:
               bold; text-decoration: underline;">You can
               always unsubscribe from this list</a>.
148        </p>
149        <!-- A secondary language set in Spanish using
           the lang attribute. -->
150        <p lang="es" style="margin: 0 0 0 0;">
151            Gracias por registrarse. <a href="#"
               style="color: rebeccapurple; font-weight:
```

The harder part of testing dark mode is testing it in actual email clients. While it's relatively easy to set up free email accounts with the major webmail providers like Gmail, Yahoo! Mail, AOL Mail, and Outlook.com, it's still time-consuming to send test emails, enable dark mode, log in, and manually test your email in each. That's where services like Litmus come into play.

Litmus allows you to test your emails in dozens of different email clients—including dark mode options—without maintaining individual accounts. It can be expensive if you're working alone, but most companies are willing to pay for a subscription to ensure their emails look and work well across email clients.

Within Litmus, you can code inside of the Builder tab, which is a full featured code editor similar to Parcel, then see actual screenshots of your email in all of the different clients. Here's our accessible email template in Outlook 2021 on Windows with dark mode enabled:

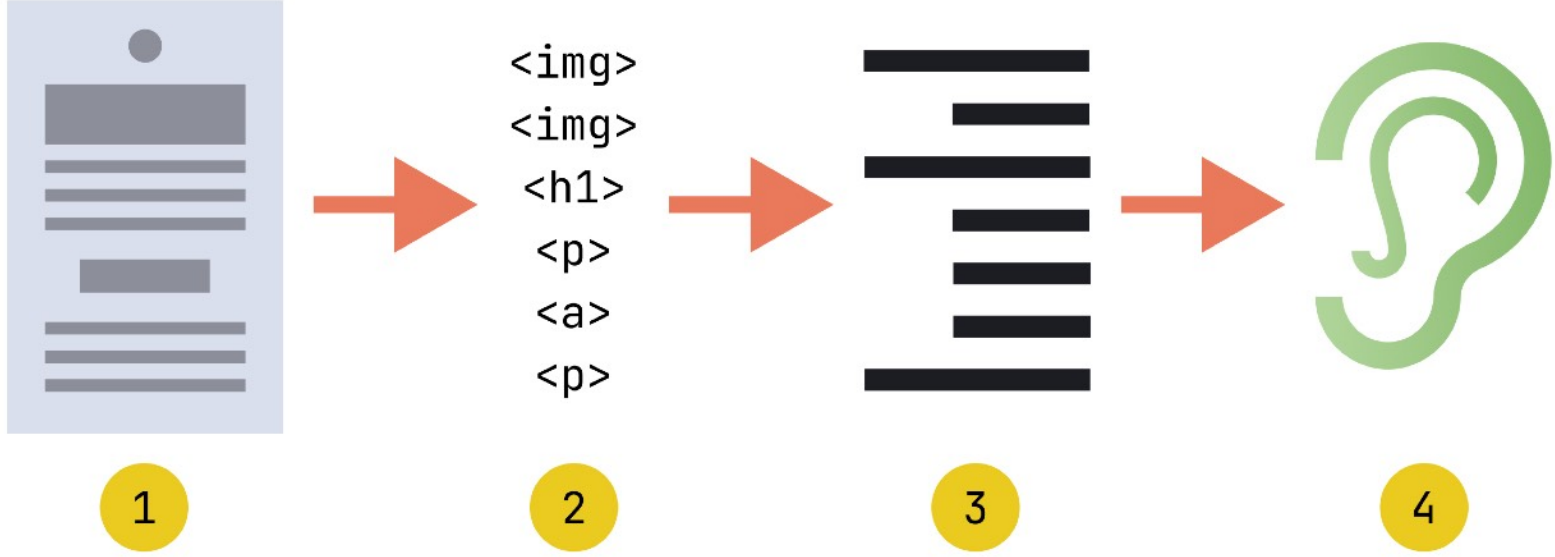


Using a tool like Litmus (or Email on Acid, another option) allows you to see exactly what your email will look like in specific email clients and in dark mode scenarios. Litmus also has some accessibility-related tools built in, which we'll see in action in the next section.

Testing Code

The final phase of accessibility testing comes once your email is actually coded. It involves testing the underlying code that powers an email to ensure that it's accessible for people using assistive technology, especially screen reader software.

As we saw in the last chapter, screen reader software works by analyzing the HTML in an email, using it to assemble an accessibility tree which can be navigated by the user, and then reading the contents within that tree out loud to the user, who can take actions like activating links via a keyboard or other input device.



All of the technical accessibility considerations are designed to create a good accessibility tree and properly expose information to a subscriber. Revisiting the dos and don'ts from the last chapter, we need to:

Do	Don't
Include a doctype.	Override user preferences.
Include language and text direction.	Forget language settings.
Keep content scalable for users.	Leave out role="presentation" on tables.
Use semantic markup.	Misuse semantic markup or ARIA.
Apply ARIA roles to enhance accessibility.	Forget to include the alt attribute on images.

Include proper alternative text depending on an images role.	
Support dark mode to the extent you can.	

The nice thing about technical accessibility is that, since it's powered by code with specific rules and roles, some of the testing can be automated, which reduced the amount of time we need to spend on testing. Still, knowing how to manually test your email with a screen reader will give you a better picture of what your subscriber hears. So, we'll start there.

Using a Screen Reader

Screen readers come in a few different flavors. Most operating systems—both desktop and mobile—include screen reader software by default. Windows has

Narrator, Android has Talkback, and Mac and iOS share a common screen reader called VoiceOver. Most browsers have screen reader extensions you can download and use for web-based content. And a variety of free and commercial standalone software packages exist, the most popular being NVDA and JAWS.

While there are some differences between how each option works and interprets content, they are mostly similar, so the best option for testing is usually whatever is built into your operating system. For me, as a Mac user, it's Apple's VoiceOver, which I'll use here and in the tutorial videos.

VoiceOver, and other screen readers, work by reading through content on the screen and allowing the user to hear that content through speakers and then interact with that content via an input device, most commonly a keyboard.

Keyboard shortcuts are used to interact with the screen reader, and these are always proprietary to the individual screen reader software so that they don't clash with other keyboard shortcuts in other applications. Because of this, there's definitely a learning curve for using and testing with screen reader software. However, you can do adequate email testing using a few simple commands.

First, you need to enable VoiceOver. You can do so by either going into **System Preferences > Accessibility > VoiceOver** and ticking the "Enable VoiceOver" checkbox or by simply pressing **⌘+F5** on your keyboard. You'll see a welcome screen

that will give you the option to learn more about using VoiceOver, start using VoiceOver immediately, or turn it off.

Tapping the “Use VoiceOver” button will enable the screen reader. You’ll start hearing options and content read out loud from here on out.

Once VoiceOver is activated, open your email in your browser. When you do, you’ll hear VoiceOver begin reading the content in your email, starting with the **`title`** that’s coded into the HTML. It will proceed to work its way through the email, reading out each element and piece of content in a linear fashion until the email is complete.

If you want, you can listen straight through to hear how your email sounds. Or, you can take over navigation from VoiceOver and focus on individual elements. You do this by using the keyboard combination of **CTRL+OPTION and the right or left arrows**. If you want VoiceOver to stop reading, you can use **CTRL+OPTION+DOWN**, which will trigger a short beeping noise to let the user know that VoiceOver is paused.

If you want to navigate solely through the links or interactive bits of your email campaign, you can use the **TAB** key. This allows you to quickly cycle through links, which is how a lot of users consume content until they find what they need. It’s also the best way to hear how your links sound and whether or not your descriptive text works for a user. Be aware, though, once you run out of links in the email, continuing to hit TAB will take you out of the email and into the controls of your browser.

A few things to take note of:

- You'll be able to hear how VoiceOver identifies key elements like "Heading Level 1" and "Heading Level 2"
- You'll hear that your alternative text is read first followed by the identifier "Image"
- You'll hear VoiceOver identify correlating elements, like lists and list items. It will pause at the list level before allowing a user to skip down into individual elements, identifying each one
- You'll hear VoiceOver identify links as "Link" followed by the text of the link
- You'll hear how VoiceOver pauses at certain text elements like the `em` tag

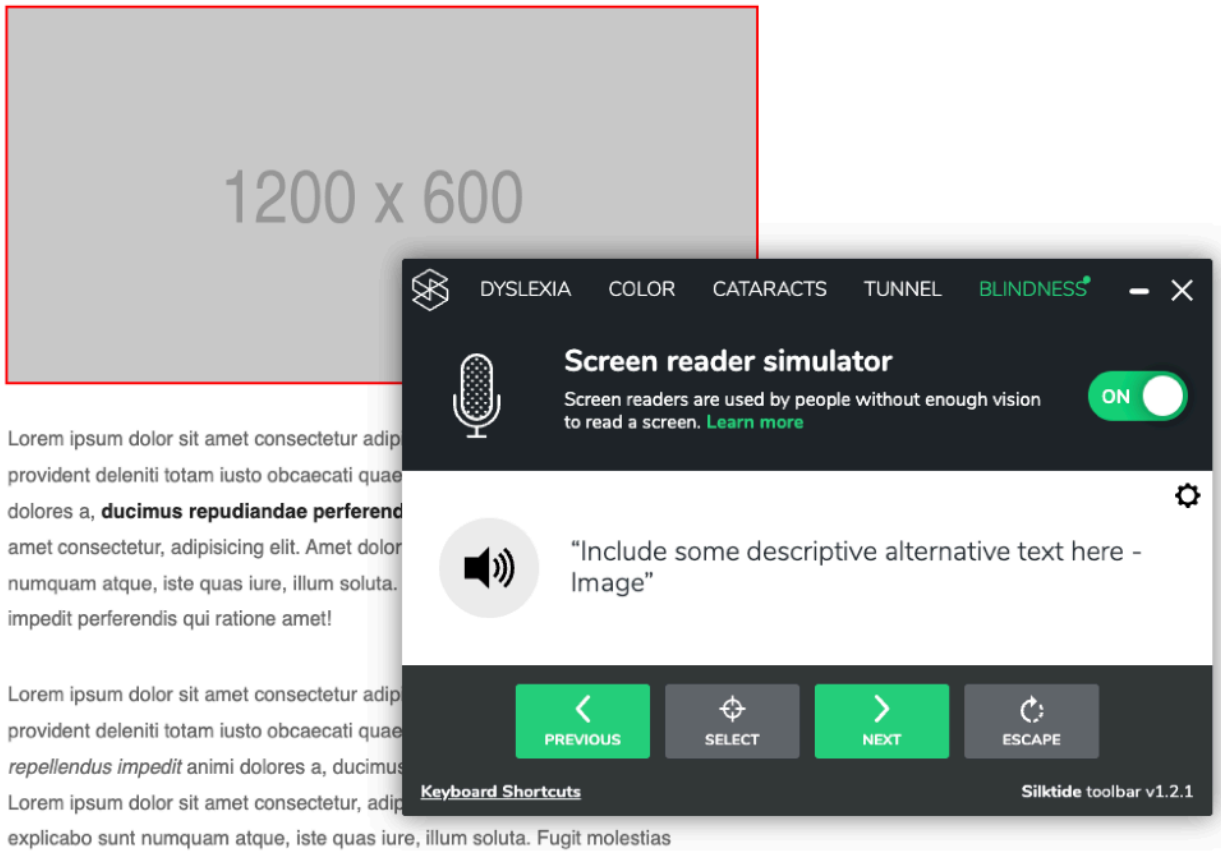
It's a great experience to have with your emails, as you'll be able to literally hear how your technical choices impact accessibility.

The goal is to ensure that everything sounds natural and understandable when VoiceOver is working. If anything feels off or confusing, then it's worth revisiting either the underlying code or the content itself to see where you can improve things.

Once you feel good about how your email sounds with VoiceOver, you can turn it off by pressing **⌘+F5** again.

If you're not comfortable using a traditional screen reader, you can rely on a tool like Silktide again to help you understand how content is read to users. The same browser extension we say before has a "Blindness" tab that allows you to hear a simplified screen reader experience and use a buttons to cycle through content. While it's not as realistic as using a traditional screen reader, it's still a good approximation.

Primary Heading



Lorem ipsum dolor sit amet consectetur adip
provident deleniti totam iusto obcaecati quae
dolores a, **ducimus repudiandae perferend**
amet consectetur, adipisicing elit. Amet dolor
numquam atque, iste quas iure, illum soluta.
impedit perferendis qui ratione amet!

Lorem ipsum dolor sit amet consectetur adip
provident deleniti totam iusto obcaecati quae
repellendus impedit animi dolores a, ducimus
Lorem ipsum dolor sit amet consectetur, adip
explicabo sunt numquam atque, iste quas iure, illum soluta. Fugit molestias
cupiditate voluptas impedit perferendis qui ratione amet!

1. Item
2. Item
3. Item

Automated Code Testing

I firmly believe that everyone should experience using a screen reader at least a few times so that they get a real sense of how a lot of users experience the web and email. But, I understand that time is precious in the email world, so you can't spend a ton of time manually testing with screen readers for every email campaign. That's where automated testing tools come in.

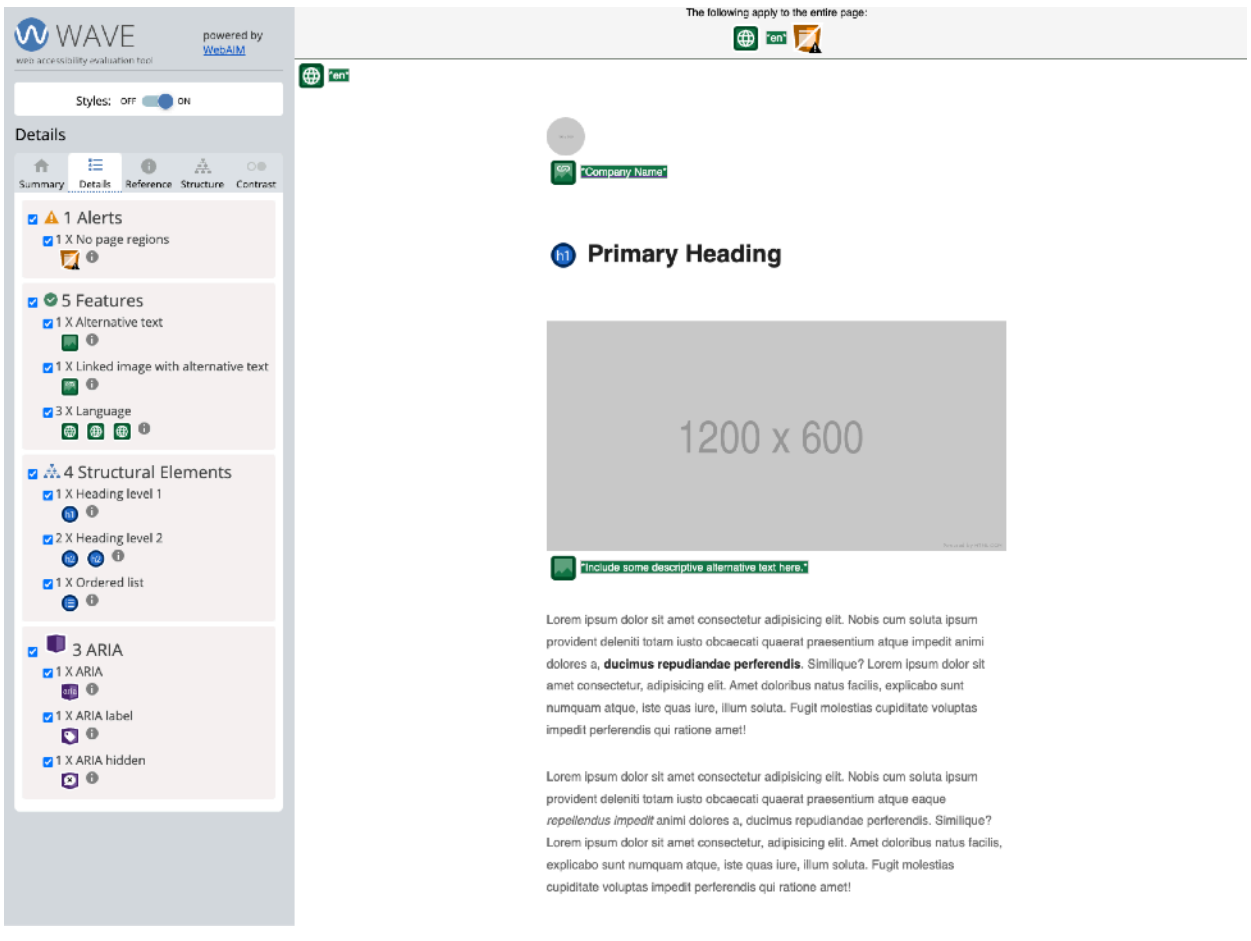
Automated tools look at your code and identify and flag issues that you can then fix. I like to use actual screen reader software when setting up a reusable email template, then run automated checks before individual campaign sends built on those templates. If you do the up front testing on the main template and ensure it's accessible, the automated checks should turn out OK or flag any minor issues that occur as a result of typos or updating the template.

Automated testing tools come in three major varieties: Browser extensions, standalone websites, and built-in accessibility checkers for code editors.

On the browser side, my preferred extension is the **Web Accessibility Evaluation Tool (WAVE)**. You can install if for Chrome, Firefox, and Microsoft Edge. Once installed, you can open the web version of an email and click on the WAVE icon in your toolbar to pop open the WAVE sidebar. Here, you'll see the results of the automated tests that WAVE runs. In particular, it looks for:

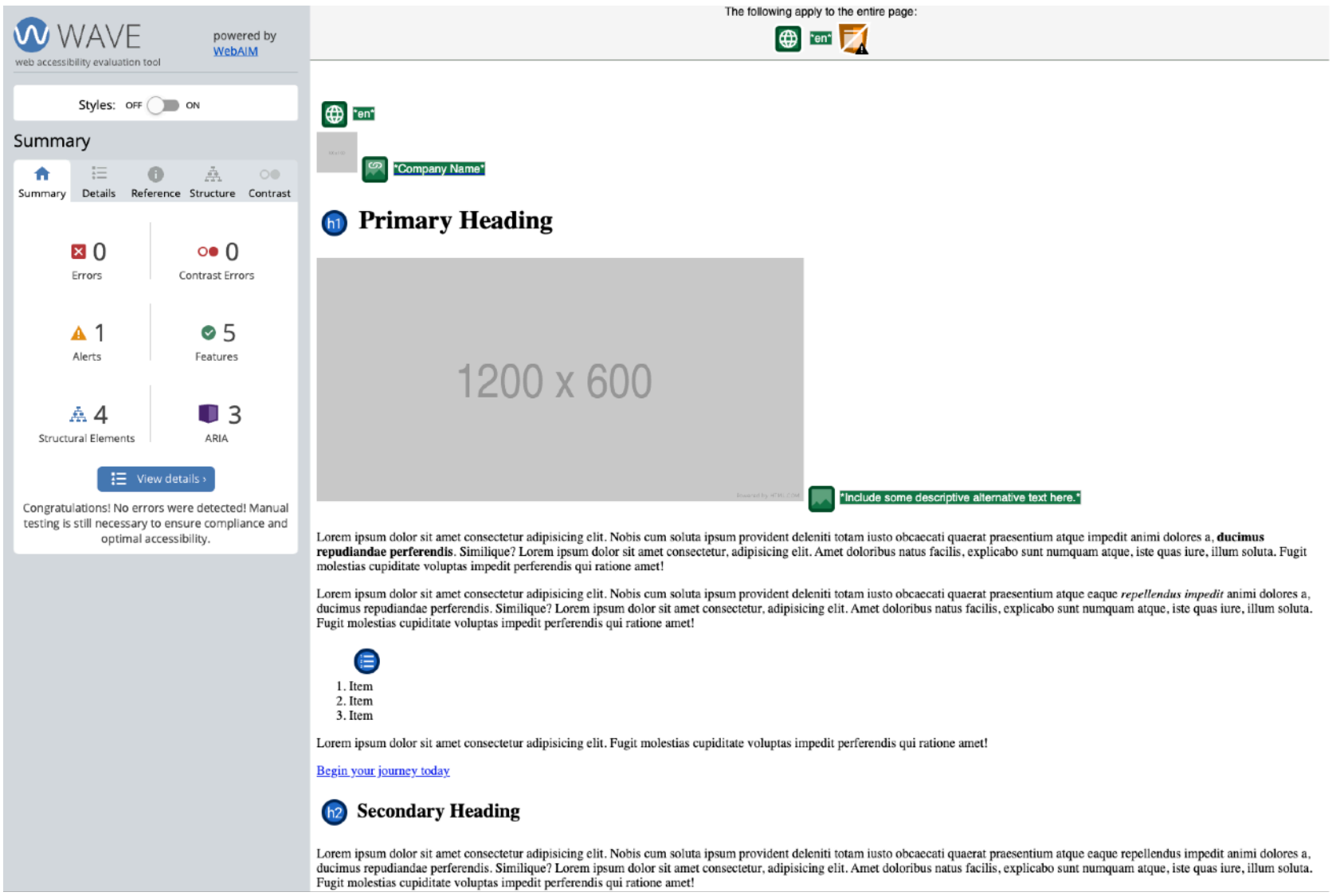
- Any errors in the code
- Contrast problems
- Accessibility features like alternative text
- Structural elements like headings
- And ARIA information like roles and attributes

If there are errors or issues with any, it will note them and allow you to dig into each in the "Details" view.

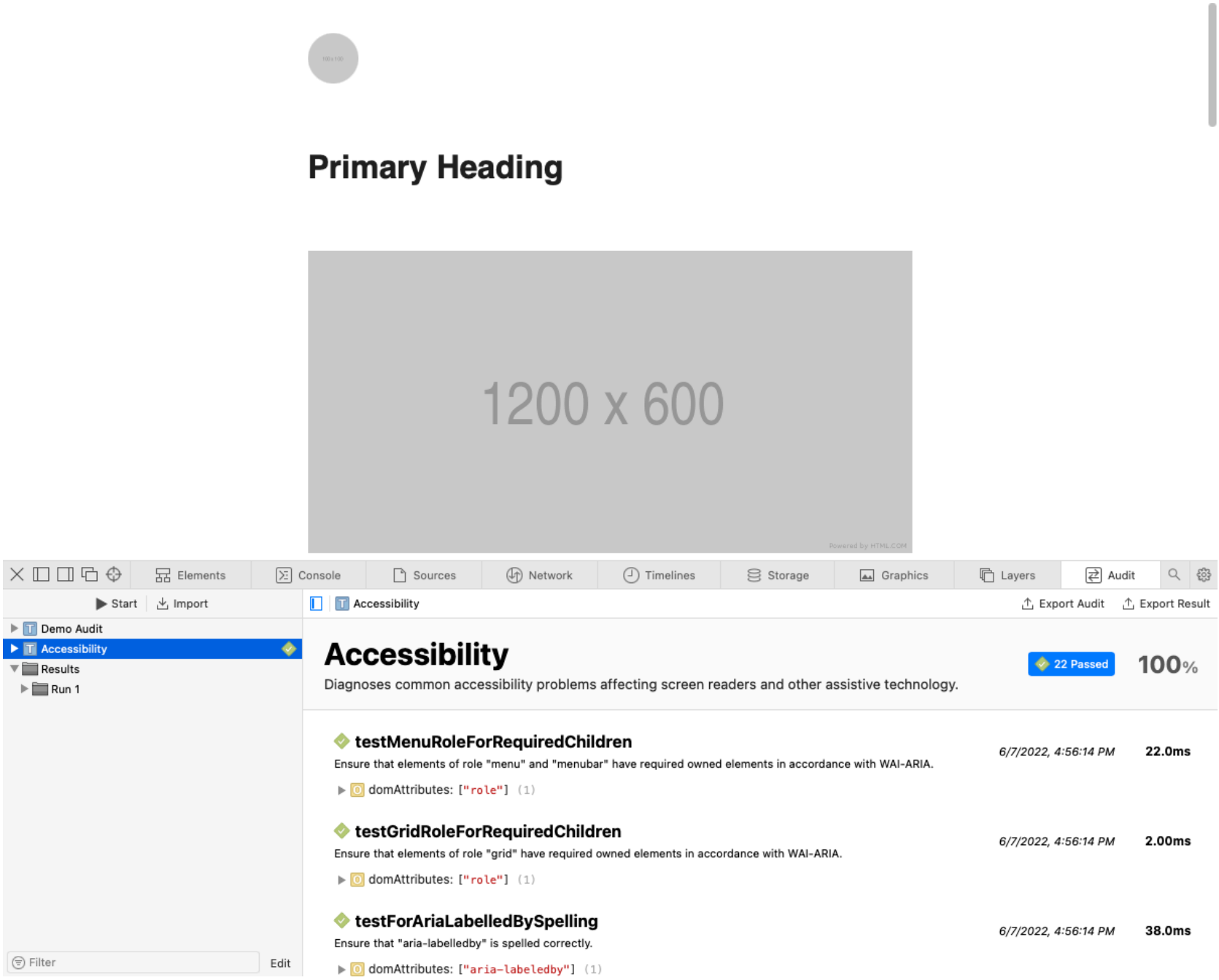


It will also identify those different elements on the email itself, allowing you to quickly scan through and match the issues to the appropriate section of the email. One of my

favorite features is the ability to toggle CSS styles on and off. While it's uncommon for email clients to disable all CSS styles, an unstyled document provides the clearest example of HTML's built in hierarchy as defined by the semantic markup you should be using. If the content is scannable and understandable without styling, you'll know that you have a good basic document coded.



Most browsers have developer tools built into them, as well. And these have accessibility tools included. For example, in Safari, you can press **⌘+OPTION+I** to open the Inspector, then select the go to the Audit panel to run an accessibility audit.

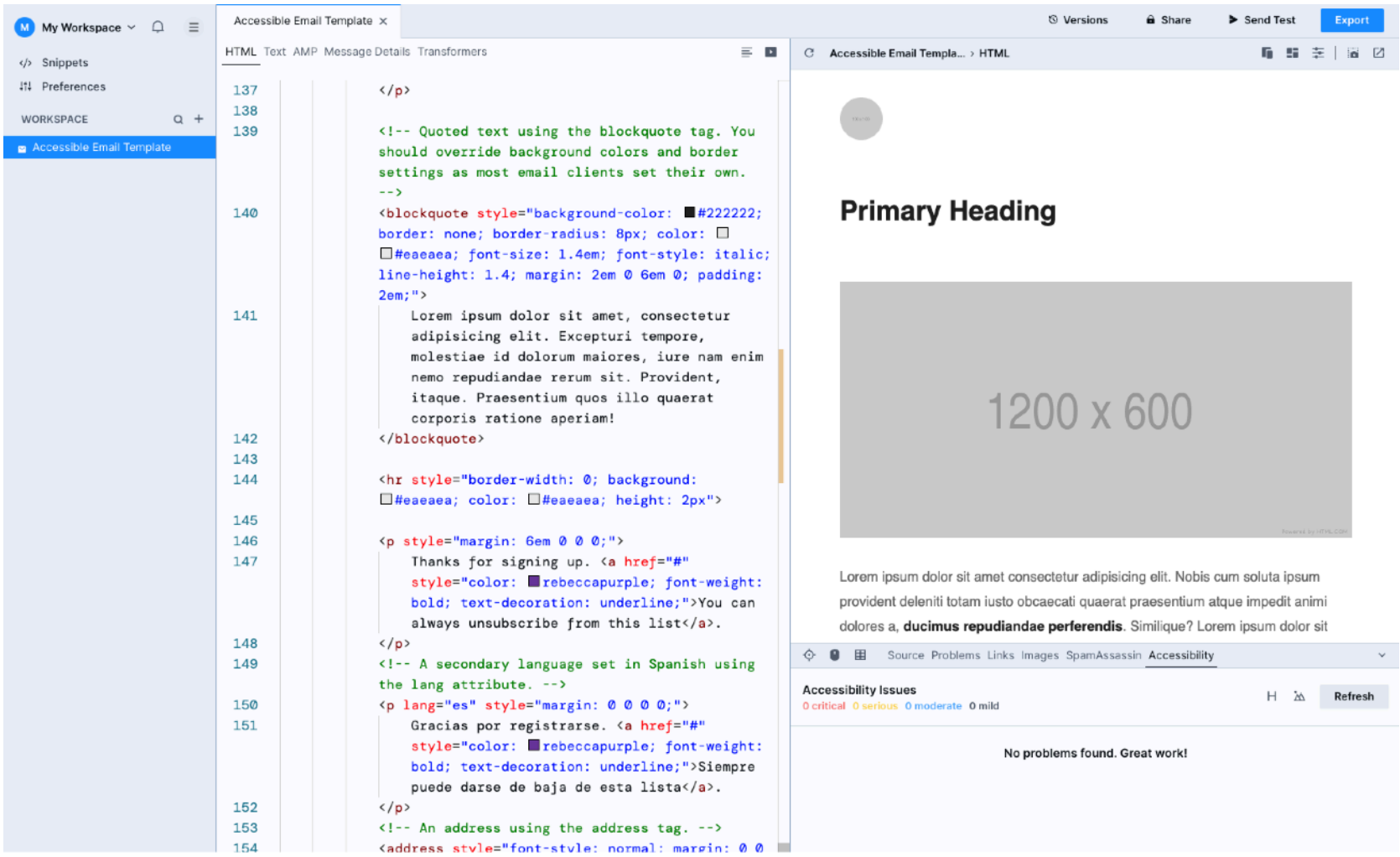


Other browser extensions exist, too, including [axe DevTools from Deque] and [IBM Equal Access Toolkit]. All can be used to inspect your code and flag any potential accessibility issues before you send your campaign.

Similar to browser extensions are standalone accessibility checker websites. These require a hosted version of the email (which most extensions do, too) and inputting a URL to the website in order to run the tests. [WAVE has a version on their homepage], which runs the same tests as the extension version. There are a bunch of these out there, but I usually defer to the WAVE tool since it's free and powered by the WebAIM group.

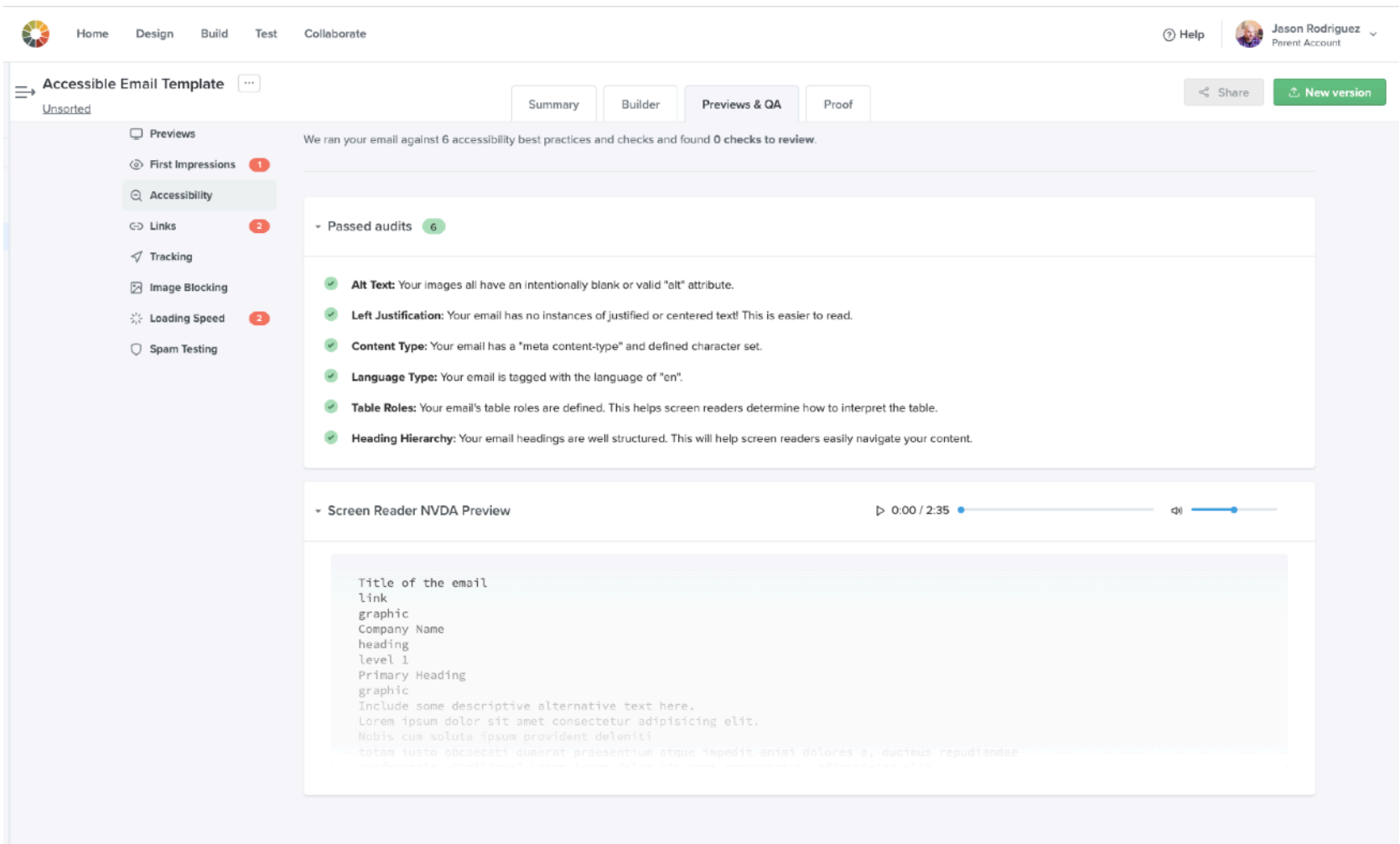
But what about when you're coding an email and you don't have a hosted version yet? This is where code editor tools come into play. And the two best options I've seen are geared towards email developers themselves: Parcel and Litmus.

Parcel has an accessibility checker built in that's powered by [Deque's accessibility rules](), the same ones that power their axe DevTools extension. It's the simple tap of a button within Parcel's workspace to run an accessibility check that will flag any issues. Fortunately, our accessible email template doesn't have any:



Litmus has a similar checker built in with a few cool additional features. Within the Previews & QA tab, there's an Accessibility section that runs your email against 6 audits:

- Alt text
- Left justification
- Content type
- Language type
- Table roles
- Heading hierarchy



While not as comprehensive as the Parcel checker or other tools, the Litmus checker tends to be a bit more descriptive when things go wrong and helps provide good, email-specific information on correcting mistakes. What's especially cool about the Litmus checker, though, is the screen reader preview below the audit summary.

You can see that a transcript of the email is included, which is generated by the popular NVDA screen reader. What's even better is that Litmus includes an audio recording of NVDA reading your email, which you can play to hear how your email sounds. For anyone that's not comfortable using a screen reader themselves, this is a great option for experience the audio version of your email campaign.

Chapter Wrap Up

Regardless of which tools you use to test the accessibility of your campaigns, you need to spend the time to make sure that assistive technology users can consume and benefit from your emails. If you encounter errors throughout the testing process, it's on you to identify and fix them so that none of your subscribers are left with a confusing, broken experience.

It can be a lot to take in so, as always, we'll end with a wrap-up of the dos and don'ts of testing email campaigns for accessibility.

Do	Don't
Use tools to write clear, concise copy.	Build a campaign without reading the copy out loud first.
Read your copy out loud to identify issues.	Send emails with low contrast or no hierarchy.
Test your emails for contrast and hierarchy.	Ignore dark mode users.
Test your emails for dark mode compatibility .	Forget to test the underlying code for accessibility issues.
Get comfortable with screen readers to manually test emails.	Ignore technical issues that create bad experiences.
Utilize automated code testing tools to ensure the HTML is accessible for users relying on assistive technology.	
Fix any errors that pop up during testing.	

Conclusion

We're here. We made it to the end of our current discussion on accessibility and email. We've covered a lot of topics, from copywriting and clear language to visual design principles, coding emails for accessibility, and an overview of testing methods.

If you haven't already, I encourage you to check out the tutorial videos that accompany this text. While I go over most of the same information, I explore different examples (especially in the coding and testing portions) to better illustrate all of these concepts in a more interactive way.

Hopefully you've got a better understanding of why accessibility in email matters, what it means to create an accessible email campaign, and the techniques used from copy to code to send better email experiences to your subscribers. If you walk away from this guide with one lesson, let it be this:

While we all have different abilities, we all deserve respectful and inclusive experiences in the real world, on the web, and in email.

Accessibility helps us achieve that goal. While it's impossible to make a perfect, consistent experience for everyone, we can get pretty close by taking into considerations the many different abilities, contexts, and situations that people experience when interacting with the digital world. It's our job as email professionals to put in the work to make our campaigns as accessible as possible.

We'll wrap up with one more section full of additional resources to help you out on your accessibility journey. If you have any questions or comments, email me at jason@thebetter.email.

Cheers!

Additional Resources

In this section, you'll find links to dozens of resources and tools I've found helpful when learning about accessibility. A lot of them are focused on the web or software applications, but most of the principles apply to email, too.

Happy learning!

Websites and Blogs

Web Content Accessibility Guidelines Overview

W3C Accessibility Overview

Mozilla Developer Network Accessibility Reference

Deque Accessibility Blog

The WebAIM Website

The Digital.gov Accessibility Section

PlainLanguage.gov

Good Email Code

The Accessibility Project

Accessible Email Documentation GitHub Repo

Essential Accessibility Resources Page

Harvard University's Digital Accessibility Hub

24 Accessibility Advent Calendar

Adrian Roselli's Blog

AlastairC's Blog

Bruce Lawson's Blog

The Microassist Digital Accessibility Digest

The Equal Entry Blog

The Hassell Inclusion Insights Blog

The Law Office of Lainey Feingold Blog

Léonie Watson's Blog

The Level Access Blog

Marco Zehe's Blog

The Microsoft Accessibility Blog

A List Apart

Smashing Magazine

Scott O'Hara's Blog

Accessibility is Political Blog

Deborah Edwards-Oñoro's Blog

Microsoft Inclusive Design Toolkit

Campaign Monitor Blog

Litmus Blog

Really Good Emails Blog

Inclusive Components

Specific Articles

Accessibility in HTML Email Presentation, Mark Robbins

The Ultimate Guide to Accessible Emails, Litmus

HTML Email and Accessibility, CSS-Tricks

Accessibility and Inclusion with the World's Most Popular Communication Tool: Email, 24 Accessibility

Stop Designing For Only 85% Of Users: Nailing Accessibility In Design, Smashing Magazine

Accessibility in Email Marketing: 7 Simple Tricks to Make Your Code More Accessible, Litmus

Accessibility vs. Inclusion: What it Takes to Create More Inclusive Email Marketing Experiences, Litmus

We're Just Temporarily Abled, UX Magazine

Alt-Texts: The Ultimate Guide, Axess Lab

Small Tweaks That Can Make a Huge Impact on Your Website's Accessibility, CSS-Tricks

Email Accessibility in Action, Rémi Parmentier

Designing for Accessibility and Inclusion, Smashing Magazine

Inclusive Design: 12 Ways to Design for Everyone, Shopify

Why Digital Accessibility Matters, InVision

Writing Great Alt Text: Emotion Matters, Jake Archibald

Writing Alternative Text That Matters, A11y with Lindsey

The Business Case for Accessibility, Tetra Logical

Everything I Know About Alt Text, Max Kohler

The Surprising Truth About Pixels and Accessibility, Josh W. Comeau

Books

A Web for Everyone, Sarah Horton and Whitney Quesenbery

Accessibility for Everyone, Laura Kalbag

Design for Real Life, Eric Meyer and Sara Wachter-Boettcher

Color Accessibility Workflows, Geri Coady

Inclusive Design Patterns, Heydon Pickering

Don't Make Me Think, Steve Krug

A Type of Email, Paul Airy

Email Marketing Rules, Chad S. White

Design Meets Disability, Graham Pullin

Mismatch: How Inclusion Shapes Design, Kat Holmes

Demystifying Disability, Emily Ladau

Communities and Advocates

EmailGeeks Slack Group

Women of Email

Mark Robbins

Alice Li

Paul Airy

Laura Kalbag

Rémi Parmentier

Wilbert Heinen

Deborah Edwards-Oñoro

Anna E. Cook

Adrian Roselli

Emily Ladua

Carie Fisher

Ethan Marcotte

Jeremy Keith

Chad S. White

Anne Tomlin

Marcy Sutton-Todd

Charles Hall

Rachel Andrew

Glenda Sims

Nick DeNardis

Andy Bell

Heydon Pickering

Cosmin Popovici

Nicolas Steenhout

Steve Faulkner

Molly E. Holzschlag

Rachele DiTullio

Josh W. Comeau

Hillel Berg

Matt May

Todd Libby

Scott O'Hara

Avi Goldman

Stéphanie Walter

Cat Noone

Andrew Hayward

Léonie Watson

Claire Benedikt

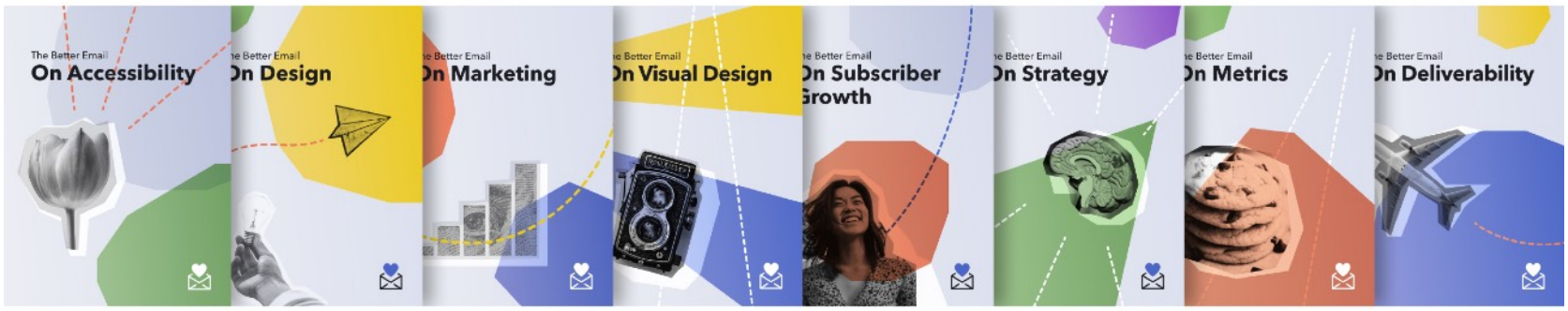
Eric Eggert

Preety Kumar

Ben Myers

Have any suggestions? Email them to me at jason@thebetter.email.

About The Better Email



The Better Email is a collection of professional guides to email marketing, design, and development based on years of research, testing, and conversations with an amazing community of email marketing folks. Each guide aims to answer the question:

How can I send better, more valuable emails?

Learn about email marketing strategy, design, and development at **thebetter.email** and access additional resources at **thebetter.email/resources**. Or sign up for weekly email links and tips at **thebetter.email/newsletter** (and get The Better Email Checklist in the process).

About Jason Rodriguez

Jason Rodriguez is an email industry veteran with a background in design and development and nearly a decade of experience in planning, writing, designing, and coding email marketing campaigns.

There's a good chance you know him from his work at industry leader Litmus. Over 7+ years, he helped build the community around email design and marketing and educated industry professionals on webinars, at conferences, and with more articles and ebooks than he'd care to remember.

Follow him on Twitter or via his email newsletter, **Regular Communication**.