



DO NO HARM GUIDE

CENTERING ACCESSIBILITY IN DATA VISUALIZATION

EDITED BY JONATHAN SCHWABISH, SUE POPKIN, AND ALICE FENG

DECEMBER 2022

TABLE OF CONTENTS

03

PART ONE

Introduction

Chapter One

Centering Accessibility
in Data Visualization

JONATHAN SCHWABISH

SUE POPKIN

ALICE FENG

11

PART TWO

A Framing of Why
Accessibility is Important

Chapter Two

The Right Tools for the Job: Learning and
Building for Data Visualization and Accessibility

FRANK ELAVSKY

Chapter Three

Designing Data for Cognitive Load

DOUG SCHEPERS

31

PART THREE

Alternative (alt) Text and
Screen Readers

Chapter Four

Writing Alt Text to Communicate the
Meaning in Data Visualizations

ELIZABETH HARE

Chapter Five

Coding Accessible Data Visualisations

LÉONIE WATSON

Chapter Six

Creating Better Screen Reader Experiences

SARAH FOSSHEIM

59

PART FOUR

Accessibility Testing and Remediation

Chapter Seven

Practical Accessibility Testing for Data Visualizations

LARENE LE GASSICK

Chapter Eight

Infographic Equity in PDF Documents:
Designing with Accessibility in Mind

DAX CASTRO

84

PART FIVE

Accessibility in Teams
and Organizations

Chapter Nine

Building Accessibility Best Practices Into Your
Organization's Data Visualization Style Guidelines

AMY CESAL

Chapter Ten

Nontechnical Barriers to Data Visualization
Accessibility in Government

MELANIE MAZANEC



PART ONE

Introduction

Centering Accessibility in Data Visualization



JONATHAN SCHWABISH



SUE POPKIN



ALICE FENG

Every day we are inundated with tables, graphs, charts, and maps explaining everything, including the unemployment rate, COVID-19 vaccination rates, baseball home run launch velocities, and our investment portfolios. When made well, data visualizations can help readers and users find insights and make discoveries. When made poorly, they obfuscate, mislead, or make it difficult for people to use them effectively.

At the signing of the Americans with Disabilities Act 32 years ago, President George H. W. Bush remarked that the law would enable “every man, woman, and child with a disability [to] now pass through once-closed doors into a bright new era of equality, independence, and freedom.” That promise remains unrealized. Disabled people still fight for full inclusion and equality when it comes to employment, health care, access to transportation, and more. And in today’s digitized era, people with disabilities do not have full access to many documents, reports, newspapers, data, and data visualizations. This report is a guide to help close those gaps and work toward a more inclusive and equitable world for the more than the estimated 61 million people^[1] with disabilities in the United States today.

For years, the data visualization field has been largely inaccessible to people who cannot process visual content the same way as others. And when researchers and practitioners have focused their efforts on creating accessible content, they have almost exclusively addressed issues around color. Color vision deficiencies—or “color blindness” in the common parlance—have been the primary focus (and often the only focus) when data visualization developers and communicators consider the needs of people who have visual impairments. Creators focused their attention on avoiding red-green color palettes because an estimated [4 percent](#) of people cannot distinguish similar shades of those colors. Less attention—perhaps even no attention—is paid to people with other forms of vision impairments, including blurriness, contrast sensitivity, and blindness.

But disabilities extend far beyond sight impairments. Some people cannot use a keyboard or mouse; some have attention management difficulties; some have disabilities affecting balance or motion; and some have learning disabilities such as sensory processing disorder

or dyslexia. And combinations of these disabilities can affect different systems simultaneously. Moreover, probably all of us have at some point had a temporary disability such as a broken arm, a migraine headache, or a concussion.

The creation and display of digital information usually do not include accessibility practices for people with a wide variety of disabilities. As content creators, we need to be mindful of new tools and guidance for making our work available equitably. By doing so, we can ensure that everyone has equal access to accurate information and data. And when more people can access and process our information, the more likely it is that our ideas, our data, and our analysis will actually be used.

This third volume in the *Do No Harm Guide* series from the Urban Institute seeks to provide in-depth lessons on how to create visualization products that are more accessible to disabled people. As with the other volumes in this series, one of the central themes in creating better, more equitable, and more inclusive content is to center the work around empathy. By thinking carefully about the needs of all people and communities—especially those who have been historically underrepresented and marginalized—we can create better and more accessible content.

As in the second volume, *Do No Harm Guide: Additional Perspectives on Data Equity*, we solicited the input of experts to help build a practical, actionable guide. We identified several authors who have experience creating accessible content, many of whom also have lived experience with disabilities. We solicited proposals from a select group of scholars and practitioners, then narrowed those proposals down to nine essays that would provide a comprehensive (though by no means complete) guide to creating accessible data and data visualization products. We also had an advisory board, consisting of four experts in data, data visualization, and accessibility, who reviewed the essays to ensure we did not omit any major issues or challenges.

As we reviewed and edited the various essays, five clear themes emerged:

Design with accessibility in mind from the beginning.

- From creating static reports and graphs (Castro, Chapter 8), to building tools and platforms (Elavsky, Chapter 2), to procuring and implementing those tools (Mazanec, Chapter 10), data practitioners will create better products by starting the process with accessibility in mind rather than adding accessibility in as a remediation step retrofitted at the end. Taking an accessibility perspective from the outset requires teams and individuals to critically consider what the expected user experience will be; who will use the tool or data; and how visual elements like text, colors, and online navigation will be consumed by the broadest audience possible.
- **Accessibility should not be a specialty.** Anyone working with data or creating digital content should understand and strive to produce accessible products. Although some of the aspects of creating accessible online content clearly require technical experience, the work should not be left to a single person or some subset of the team. Mazanec (Chapter 10) discusses how people at different levels in a management hierarchy have their own roles to play, and Cesal (Chapter 9) lays out strategies for creating data visualization styles and style guides that incorporate accessibility. Elavsky (Chapter 2) calls for more resources to address the knowledge gaps many web developers have.
- **There is no established definition for what makes a data visualization accessible.** Although Web Content Accessibility Guidelines lay out requirements for making accessible websites, no standards have yet been agreed upon for how to make data visualizations accessible. Our authors offer several ideas and approaches, including linking to the underlying dataset and adding screen reader interaction to web-based charts. Le Gassick (Chapter 7) walks through

the specific issues to look for when conducting accessibility tests.

- **People with disabilities should be involved in the design of data visualization products and usability testing.** Many of our authors urge data practitioners to involve people with disabilities in the design and development of data visualizations and data-driven products to ensure their needs are met. Charts, graphs, and other content should also be tested by users with disabilities to identify any usability or accessibility issues.
- **There is not a single right answer for writing alternative (alt) text.** The phrase alt text shows up a lot in this volume. Three chapters are devoted exclusively to helping you write better alt text in your graphs and documents. But there is no single right answer or right strategy for writing effective alt text. The final visual product, the platform on which that product is being published, the target audience, and the technology being used all influence the best approach. The guiding principle is to write alt text that gives disabled readers as close to the same experience as nondisabled readers as possible.

As comprehensive as we believe this volume is, we are still missing key elements of creating truly accessible data and data visualization products and an accessible web more broadly. First, this report omits how people with certain physical disabilities use online content. How do and should data visualization tools, platforms, and websites enable people who cannot use a mouse or keyboard? Are there new technologies, platforms, and best practices that can help these users better use and experience the web?

Second, we do not investigate the (slowly) growing use of sonification to communicate data. Some practitioners and media outlets are now using sound to communicate data, such as piano notes aligned with changes in political polling results in Germany^[2] or entire scores of sounds based on data values.^[3] Interesting joint work between the Sonification Lab at Georgia Tech and the Highcharts data visualization

company to create a free, open-source tool to combine data and sound is also very promising.^[4] This area of data sonification is still in its early stages, but it holds potential for enabling people to interact with data in new ways.

Third, we don't provide "an answer." Perhaps we started this work with the hope of finding a concrete answer on how to write alt text or how to approach different technical challenges. But the actual experience, as usual, is more complex than imagined.

When creating this volume, we tried to follow a core principle of the Disability Justice movement—"nothing about us without us." We worked with and relied upon the involvement of people with disabilities throughout the project, including contributors, advisors, and reviewers.

Our editorial team is also unique. Jonathan Schwabish is a researcher who focuses his data visualization work on static graphs or relatively straightforward dashboards. Alice Feng is a data visualization developer who creates unique and bespoke visualizations using JavaScript and other object-oriented programming languages. Sue Popkin is codirector of Urban's new Disability Equity Policy Initiative and the founder and coleader of Urban's Disability Affinity Group, which seeks to make Urban a more equitable and inclusive workplace for staff with disabilities. Between the three of us, we bring together technical expertise and the experience of living with disabilities. Popkin helped ensure the language and concepts in this volume reflect the needs and perspectives of people with disabilities and ensured our work avoids ableist language and assumptions. As with much of the language that seeks to promote equity, language about disability is constantly evolving and changing, but we have done our best to follow Urban's comprehensive guidance.

We also strive to make this volume as accessible as possible for a nontechnical audience. Much of the technical language is unavoidable—after all, creating online content requires tools or programming languages that not everyone knows or understands—

but our review process helped ensure these essays include clear examples, clear language, and accessible data visualizations and images.

The essays in this volume do not need to be read sequentially, though we have organized them with that approach in mind. We have divided the volume into four main parts. The two essays in Part 1—from Frank Elvasky and Doug Schepers—provide a big-picture view of creating accessible content. Elvasky uses his work on the Chartability tool—a framework for creating accessible data visualizations—to make the case for data tools that enable people to build accessible content. Schepers explains the foundation of how our eyes and brains work to process visual content (“cognitive load”) and how those processes work differently for people with different kinds of disabilities.

Part 2 of the volume focuses on creating alt text and screen readers. Liz Hare provides some basics on how to think about and write alt text. She explores two different models to write effective alt text and shows how users might incorporate those approaches into their work. Sarah Fossheim and Léonie Watson explore best practices around screen readers—tools that are used by blind people, people with low vision, and people with learning disabilities who consume text in audio formats. Fossheim provides an in-depth exploration of how screen readers work and how to write alt text and add it to data visualizations. Watson provides technical guidance on how data visualization developers can use Accessible Rich Internet Applications to create better screen reader experiences in their online work.

In Part 3 of the volume, we turn to testing and remediation. Larene Le Gassick plays the role of accessibility tester and demonstrates how to evaluate a website or data visualization for accessibility. Dax Castro dives into more detail, showing how to make PDF reports and standalone graphics more accessible by incorporating tags, alt text, and layers into your documents.

Finally, Part 4 takes an organizational view of incorporating accessibility into the data and data visualization workflow. Amy Cesal provides practical advice on how to create a data visualization style guide with accessibility at the forefront. And Melanie Mazanec describes how existing rules and regulations in the US federal government are a good start for creating accessible content, although those regulations have a long way to go before they’re ideal.

Together, we hope these nine essays provide a solid foundation for beginning to think about how to create accessible content. You will find specific, detailed information on how to make single graphs, reports, and websites more accessible by considering the accessibility of colors, fonts, motion, and text. You will also find higher-level considerations around teams and organizations, models for accessibility standards, and ways to build more inclusive products.

This third volume of the *Do No Harm Guide* series does not cover everything—there are more avenues to explore, platforms to evaluate, and solutions to uncover. As with other reports in this series, the lessons described here are not fixed rules, but we hope they provide a starting point at which you can begin your journey to create better and more inclusive work. By doing so, you not only make it possible to hear more voices, you also ensure your work is accessible to everyone.

Chapter One Notes

¹ <https://www.cdc.gov/ncbddd/disabilityandhealth/infographic-disability-impacts-all.html>

² <https://interaktiv.morgenpost.de/spd-absturz-sound/>

³ <https://www.loudnumbers.net/>

⁴ <https://sonification.highcharts.com/#/>



PART TWO

A Framing of Why
Accessibility is Important

The Right Tools for the Job: Learning and Building for Data Visualization and Accessibility



FRANK ELAVSKY

Since 2010, the number of websites in existence has grown from less than a quarter billion to nearly 2 billion,^[1] and data visualization's explosive growth has followed across every domain: personal, corporate, government, policy, and research.

We collect far more data than in the past, and as a result, we have started building faster and better tools to make that data useful. Now, we encounter charts and graphs for everything, such as sports, local infection rates, stock prices, and elections.

But these empowering advancements have not been made available to people with disabilities. Those who are not able to access these data visualizations are left with clunky, broken, and ineffective systems.

Who do I mean by “people with disabilities?”

Well-meaning folks often assume that accessibility in visualization focuses on visual disabilities, such as color vision deficiency, blindness, and low vision. But interactive data visualizations in complex tools and in applications can also produce barriers for people with cognitive, vestibular, and motor and dexterity disabilities. In fact, this problem isn't unique to visualization. One study found that academic accessibility research disproportionately focuses on visual disabilities.^[2] When I write “people with disabilities,” I refer to a broad spectrum of people. Not everyone with disabilities will encounter barriers with data visualizations, but I want to encourage you to consider that most people with disabilities might.

Although for decades we have had the tools and standards to build an accessible internet, 97 percent of the top million website home pages still fail basic, automated accessibility tests, according to the latest WebAIM Million report.^[3] Further, automated tests leave out anywhere between 57 to 80 percent of other accessibility considerations,^[4] meaning the state of content on the web for people with disabilities is likely worse than measured.

Even with this terribly small share of accessible home pages, the state of data visualizations is even worse—virtually all are inaccessible in some way or another. In 2021, I created Chartability,

a set of heuristics pulled together from designers, developers, researchers, and practitioners to evaluate data visualizations for visual, motor, vestibular, neurological, and cognitive accessibility.^[5] In all of my professional audits using Chartability—over 80,000 tests performed to date—not a single data visualization scored 100 percent for accessibility, and most consistently scored worse than every equivalent test performed by the WebAIM Million report.

As the data visualization field continues to expand and grow, accessibility considerations need to be central to content creators' analyses and communications. If creators continue to use the same methods and tools without interrogation, the current state of inaccessibility will only worsen. To make the future more accessible, creators first need to recognize what tools and solutions aren't a good fit, become more knowledgeable about accessibility, explore how we can improve the tools we have, and build new tools entirely.

Not Every Technical Solution Is a Good Fit for Accessibility

How can we intervene on human-built problems of accessibility in data visualizations and the usability of the web more generally? Business-minded approaches seek to reduce up-front cost by leaning on quick-fix solutions like web accessibility overlays, which consist of third-party code, programs, or tools that are layered on top of existing websites to make them more accessible.^[6] Researcher-minded approaches strongly incentivize “solving” inaccessibility by using artificial intelligence or machine learning methods on things that have already been built.

Both approaches try to fix accessibility problems using the same tools and systems that created those problems. Although this goal is noble, the results are not good. Overlays and machine-learning solutions suffer from many of the same fundamental issues. They force people with disabilities to rely on machine-generated accessibility, provide a poor overall experience compared with human-authored

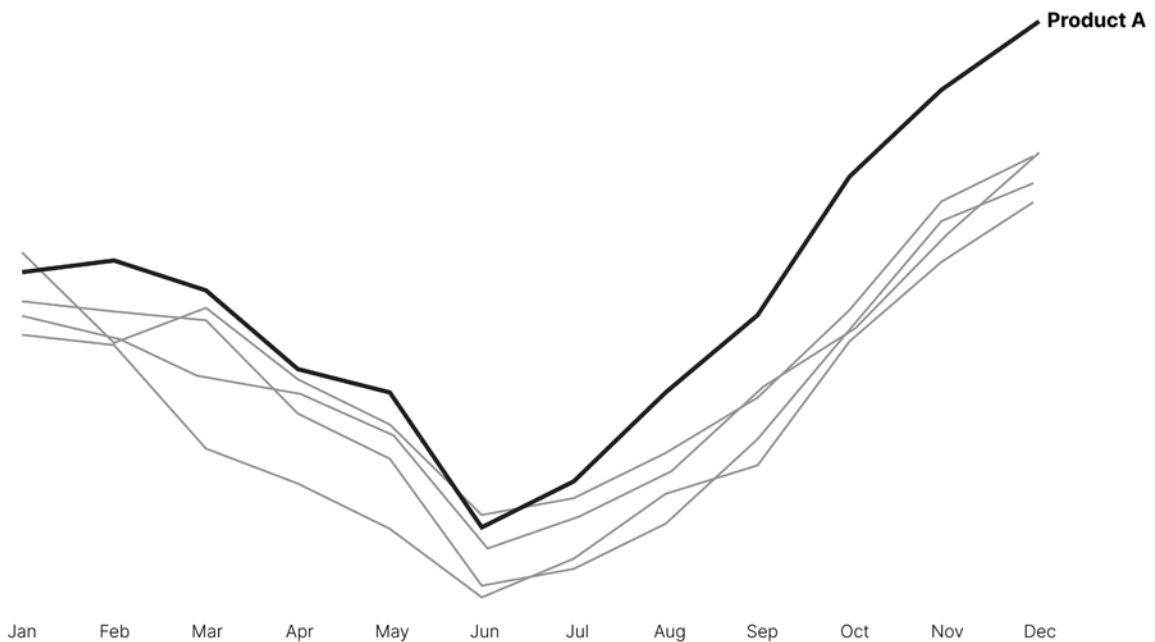
functionality, and fail to contextualize and provide appropriate interpretations of information.

Data visualization practitioners continue to rely on machine learning models that magnify existing inequalities. As an example, practitioners have tried build machine learning models that attempt to automatically interpret chart descriptions and insights without any guidance from the author.

Writing alternative (alt) text to describe visual content on the web takes some serious consideration (see Chapters 4, 5, and 6). In a 2020 blog post, disability and accessibility expert Sheri Byrne-Haber argued that no single answer exists to the question of “what is the right alt-text for an image?”^[7] “Context is the most critical aspect of alt text everyone seems to miss,” she writes. As an example, she shows eight different possible descriptions of an image of a Jack Russell Terrier wearing sunglasses. All of the descriptions could be valid depending on the author's intent and the context that surrounds the image. Here I provide my own version of this test. Do you think a machine learning algorithm could know which of the following descriptions of this line chart is the right fit?

1. **“Line chart.”** If this image appeared in the chart-picker interface of a tool like Excel or Tableau, the chart type is the only piece of information that matters.
2. **“A line chart with five lines, titled Product Performance in 2021.”** If the image appeared instead in a mockup made by a designer whose work is in progress, only the chart type and title would matter, because the data are made up and the design is likely to change.
3. **“Line chart. Product Performance in 2021. Product A is outperforming all other products.”** In a presentation to a decisionmaker who wants to know which product is doing the best, both the data and design matter.

Product Performance in 2021



4. **“Line chart. Product Performance in 2021. All products trended down sharply from January until June and slowly stabilized back into positive territory by November.”** In a report to analysts who want to know about larger trends, the takeaways from the data are what matter most.

5. **“A minimalist, greyscale line chart with five lines titled Product Performance in 2021 using a full-width, bold-weight, sans serif typeface. Product A’s line is emphasized with a near-black charcoal color, thickened stroke, and direct label at its end. All other lines in the chart are shown with reduced importance and are thinner, unlabeled, and colored with a softened grey. There are no data labels, grid lines, or y-axis on the chart, but an x-axis shows abbreviated months of the year from January to December in a small, light, sans serif typeface.”** In a design portfolio or article that explains minimalist chart designs, the design details are what matters.

6. **“(The alt text is set to NULL, and the image is marked as decorative/presentation-only).”**

The graph is simply decorative and not intended to convey content. It might be a background image or meaningless filler. (It’s worth noting that in practice, you can’t just leave an alt text box empty. The image must be purposely set as decorative or “NULL”; see Chapter 5 for more details.)

Just like Byrne-Haber’s example, any of these (or none of these) descriptions could be appropriate depending on the context. Will a machine learning algorithm know what other text, subtext, functionality, and content surrounds the chart or what the chart is intended to be used for? Only a human author really understands why they’ve made something and what they believe their audience should know. But as Louise Hickman and Alexa Hagerty argue in a 2021 blog post, solutions that fit people’s needs are often in tension with solutions that scale, such as machine learning or artificial intelligence.^[8] Algorithmic solutions tend to avoid nuance and focus on a one-size-fits-all design, which can create problems for people with disabilities.

This nuance is why it’s imperative that the people who create charts and graphs (with the assumption

that they have some understanding and intent for that visualization’s use) also understand how to make their work accessible for people with disabilities.

Gaining Practical Knowledge about Accessibility Is the Prerequisite to Using Tools Effectively

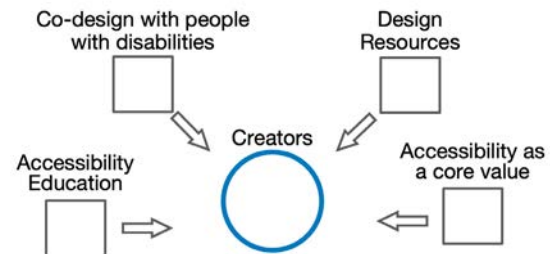
“The classroom remains the most radical space of possibility in the academy,” wrote author and activist bell hooks.^[9] She argues that teaching people how to think about the problems in our world can cultivate the most radical opportunities to enact change.

If we educate designers, developers, researchers, and authors about the importance of accessibility and how to implement accessibility into their work, perhaps they will be the ones to make their work more accessible and imagine new futures. Before we imagine what the right tools are for the job, we need to have the collective knowledge of what accessible experiences should be. Many tools are capable of being accessible, but practitioners need to think about accessibility and their design goals before they can use those tools effectively.

Providing a floor of practical knowledge was my goal for Chartability. In my collaboration with Dominik Moritz and Cynthia Bennet, we explained that Chartability would take existing accessibility knowledge and synthesize it for the difficult domain of data visualization.^[10] I operated on the assumption that we need a level of self-sufficiency and capability to recognize quality or challenges as we do our work. Chartability exists as a design resource for practitioners to evaluate their own visualizations. As a field, we need more resources like it. People need patterns, guides, and reusable materials.

But a tool like Chartability doesn’t set a ceiling or push the boundaries of what we already know. It serves as a record of the bare minimum of what we should be doing. Working alongside people with disabilities is the single most important thing we can do to go beyond the minimum and work toward accessibility. People with disabilities are the actual experts who can tell us what is or is not accessible to them. We need them as

coworkers, codesigners, leaders, and experts who help build things with us. But this guidance must always include a caveat: be wary of asking for free labor as you embark on a path toward learning. Do everything you can to find what has already been documented in standards, blogs, research, and articles, and compensate experts for their expertise when you work together.



Improving practical knowledge also entails teaching accessibility as a foundational skill. Although people new to accessibility work often find it difficult, this barrier largely arises from missing skills. Self-teaching accessibility frequently leads to a mismatch of skills: practitioners are highly advanced in one area of design or development by the time they recognize the significant gaps in their understanding. I’ve seen this gap time and time again with web engineers who don’t know how to properly use programming languages like HTML and CSS. Unfortunately, these engineers get so deep into bad practices that the basic skills necessary for accessibility become significantly expensive to relearn.

Knowledge alone can’t fix everything. We need to be able to use what we know. If our tools and materials aren’t designed to make accessibility work easy, we will continue to take the path of least resistance.

Builders Need Improved Tools

I’m a maker for makers: I build tools that others use to create. And I firmly believe that improving the tools we have or simply selecting the best tools out there are two of the easiest ways we can stop



Installation view: Tales of Our Time, November 4, 2016-March 10, 2017, Solomon R. Guggenheim Museum, New York. Photograph by David Heald © Solomon R. Guggenheim Foundation, New York.

building inaccessible data visualizations. Think of this process as slowing down or tightening the flow on a firehose. Right now, the overlays and machine learning algorithms that eternally try to clean up the mess we create are like Sun Yuan and Peng Yu's haunting art installation of a robot forever mopping its own fluid.^[11] We are making messes with our tools faster than we can clean things up.

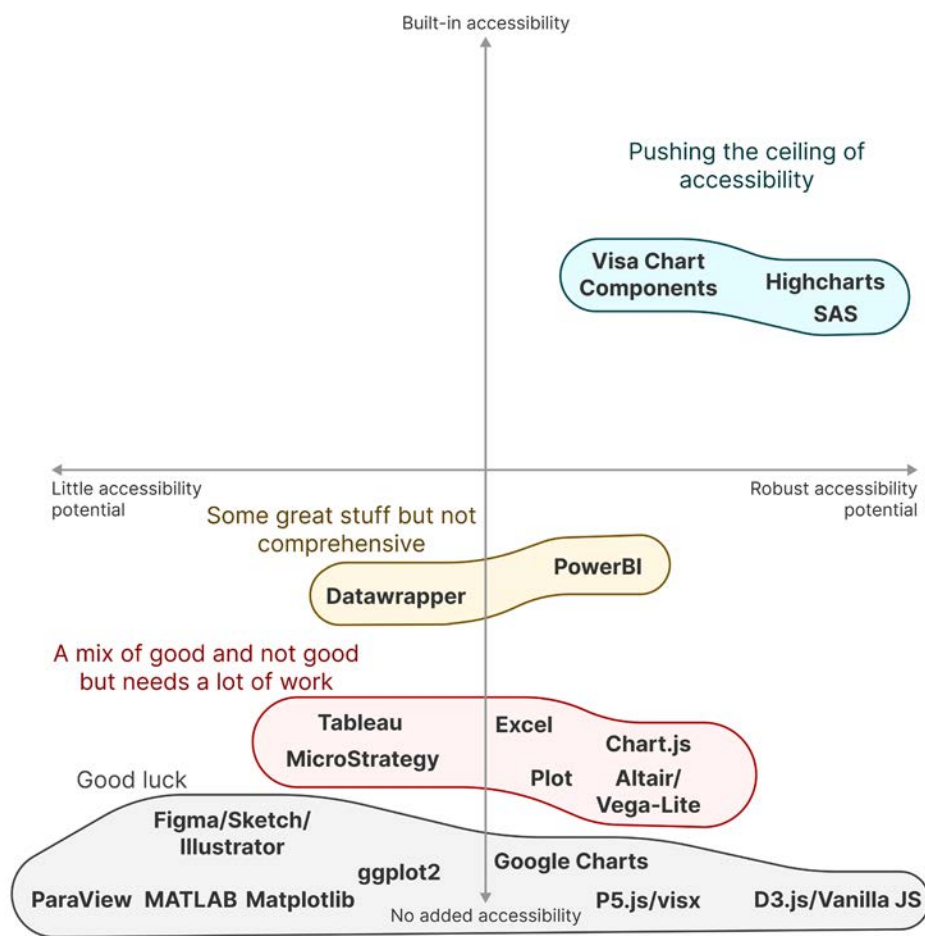
In my previous work for Visa Chart Components—a design system toolkit for Visa,—we wanted to empower creators at Visa to easily make accessible charts and graphs.^[12] We wanted to be opinionated about our design, which is the mission of any good design system (Chapter 8).

This work revealed to me that there are four high-level variables worth comparing when selecting visualization tools for the job:

1. Ease of making a visualization from data
2. Expressiveness of the visuals
3. Out-of-the-box (ready-to-go) accessibility
4. Robust/flexible potential of the accessibility

Significant time is spent considering the first two when selecting the right tools for the job. But have we paid the same attention to accessibility? To assess that question, I provide my opinions on various visualization tools' accessibility abilities based on my experience auditing and working with these tools. (Keep in mind this assessment is not in any way legal or compliance advice.)

Currently, most of our visualization tools aren't accessible out of the box, and nearly half don't have strong potential, either. Most accessibility design and development relies on the visual's creator to know about and find a way to implement it. For example, someone has to know that alt text is important and figure out how to add it to a chart or graph. But some tools help designers and developers do the right thing while using the tool. In the same way that Excel automatically sets defaults for spacing and color schemes, some tools have defaults and out-of-the box functionality for accessibility. Visa Chart Components, for example, provides a warning and instructions to developers when they've forgotten to provide expected inputs.



At a high level, the accessibility capabilities of a tool can be improved by ensuring the following:

- The important details within the visualization can be accessed by screen readers, keyboards, and other assistive technologies. Most static images, like JPG or PNG files, can only be described by alt text, meaning more complex features are often lost.
- Users can download the data underlying the chart to their own computer in an accessible format (such as a CSV file).
- The data in the chart can be represented with sonification (as tones that can be heard) and in a tactile format (that can be touched and felt).
- Creators can annotate the visualization and guide readers through it.

Tools can also offer the following functionality out of the box:

- Automatic contrast adjustment
- Redundant encodings
- Text spacing
- Interactive highlighting and dimming
- Tooltips
- Semantic structure
- Safe color palettes
- Accessible examples in core documentation

Easy-to-follow core documentation of accessible practice is vital as a first step. Highcharts, a JavaScript visualization library, provides a stellar example of how tools can provide such documentation.^[13] Often when it comes to tools that make charts and graphs

using code, practitioners learn how to use the tools from their documentation. And the more that these core documentations can integrate accessibility considerations into every step of the tool's use, the more that community members will follow suit and do the right thing. Datawrapper, another data visualization tool, has guidance built in and includes considerations for alt text and color vision deficiency.^[14] The tools themselves are the first place where correct, accessible best practices should be communicated.

However, when core documentation is absent, practitioners often learn how to use a tool from community-contributed examples. Very few, if any, community examples on blogs or websites include considerations for accessibility. This absence leads us to a larger problem: when someone makes something with an envious functionality, other practitioners copy it directly or otherwise emulate it. If the first product is inaccessible, that inaccessibility ends up being reproduced every time it is copied. The Tableau community is especially bad at this: expert users often “hack” Tableau to build extraordinary application-like interactivity or produce advanced visualization types, but these examples almost always generate nightmarish tangles of inaccessibility. Similarly, it is difficult to intervene when inaccessible visualization recipes are shared by creators on Observable, company websites, or social media. If we don't have a community that can recognize inaccessibility and catch these bad examples, we will continue to reinforce bad practices.

Another way to counter the lack of documentation, beyond catching bad practices and building good examples, is to highlight builders and makers doing the best with the tools they have. The data team for the City of San Francisco is a good example. Not only did they carefully document how to make public dashboards of COVID-19 case data accessible to the public using PowerBI, they also created a set of guidelines for others embarking on similar work.^[15] Chris DeMartini has documented his accessibility journey with Tableau, including how he

went to great lengths to add keyboard interactivity within charts, which is currently not something that Tableau offers.^[16] DeMartini used the hackability of Tableau for accessibility rather than just for extraordinary visual effects. These members of their respective communities are showing what their tools are capable of, pushing limits, and inspiring others. It is important that we look for and encourage this work when we encounter it.

But tools shouldn't rely on hacks to make accessible data visualizations, they should create accessible products by default. Practitioners can pressure toolmakers to make changes to their tools. Pressure works. Advocates in every major visualization community already do this work: the teams developing data visualization tools and platforms like PowerBI, Datawrapper, the Jupyter Project, Vega-Lite, Tableau, and Observable's Plot have all made strides for accessibility following their initial releases because of internal and volunteer efforts. PowerBI and Jupyter, in particular, have invested significant effort in the past few years. PowerBI has worked to ensure that tool itself, not just the output from the tool, is accessible. The community of practitioners around Project Jupyter have organized an accessibility working group to improve their tools, including Jupyter Notebook, which is one of the most common collaborative data science notebooks used today.^[17]

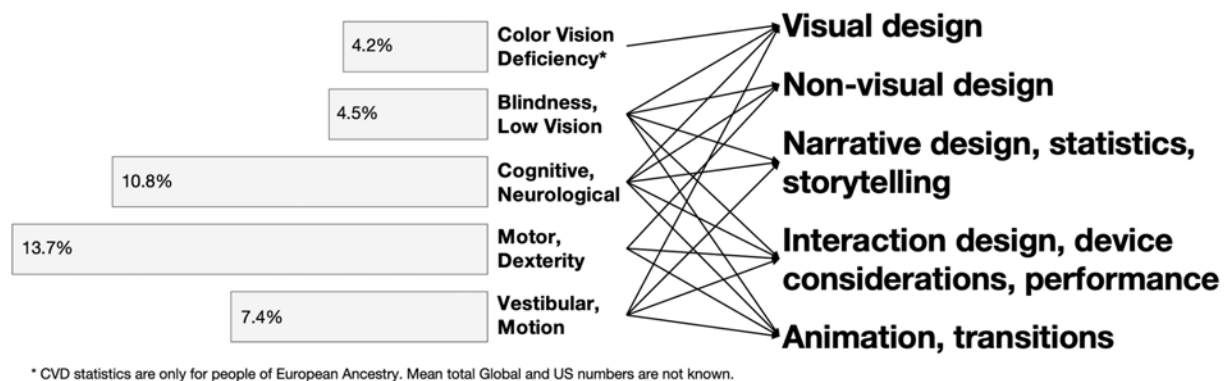
They've built a coalition of funded and volunteer work to shape the future of accessible data science but still have a long road in front of them.

Many of our tools are salvageable as long as we get involved. If more practitioners demand better, more accessible tools, companies might begin making accessibility a priority.

It Isn't Too Late for Visualization

Data visualization is in a hopeful place as a field: we've experienced explosive growth in tooling and practice in the past two decades, and nothing is so entrenched that it cannot change. Two hundred years ago, we had tactile maps and graphs made for people who are blind or have low vision. Nothing is stopping us today

Considering Disability In Design Affects Every Area of Visualization



from also making information experiences broadly accessible in old and new ways.

Our mission as a community is to surface insights out of data. Our field can grow beyond centering visuals and remain true to this practice. In the Nightingale data visualization blog, Doug Schepers writes that “accessibility is at the heart of data visualization.”^[17] Making complex information understandable doesn’t have to be limited to just visual representations. I want to inspire everyone reading this essay to imagine more expansive data experiences, interactive stories told in written form that can involve sound and touch as well.

Disabilities can have many dimensions that affect every part of data representation in design. In my talks, I use the figure above to demonstrate how considering the distributions of disability among people living in the US could affect every area of our practice. Everything on the right side of the figure we already do. Considering people with disabilities encourages us to become better at each of these things.

Data visualization is at a watershed moment. We have an opportunity to grow and mature in ways that could inspire many other fields. We have a beautiful community of researchers, designers, engineers, scientists, analysts, reporters, and storytellers. Accessibility is an opportunity for all of us to hone our craft and become excellent at what we do.

We can’t rely on overlays and machine learning models to solve access problems for us. We are the ones designing and building data visualizations, so the onus falls to us. Although I wish my takeaway could be as simple as telling you to begin learning accessibility and working to improve your tools (which of course you must do), I also want to stress that we have structural and cultural priorities to change as well. Our employers need to make accessible workplaces for our coworkers with disabilities. Our budgets need to include accessibility as a top priority—and we need to compensate disabled people for their advice and expertise in making our products better. Our immediate roadmaps need to include accessibility as part of our core products. Our educational programs need to teach about access and disability as foundational topics. Every data visualization contest or celebration needs to include accessibility as a measurement of excellence.

Until we see changes like these, however, individuals must do what we can, and it won’t be easy. So get learning and building, and remember: pressure works. Let’s take our relationship with our tools to the next level and never compromise on accessibility.

Chapter Two Notes

- ¹ “Total Number of Websites,” InternetLiveStats, accessed October 14, 2022, <https://www.internetlivestats.com/total-number-of-websites/>.
- ² Kelly Mack, Emma McDonnell, Dhruv Jain, Lucy Lu Wang, Jon E. Froelich, and Leah Findlater, “What Do We Mean by ‘Accessibility Research?’” In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, paper 371 (New York: Association for Computing Machinery, 2021), <https://doi.org/10.1145/3411764.3445412>.
- ³ “The WebAIM Million,” WebAIM, last updated March 31, 2022, <https://webaim.org/projects/million>
- ⁴ “The Automated Accessibility Coverage Report,” Deque, accessed October 14, 2022, <https://www.deque.com/automated-accessibility-testing-coverage/>.
- ⁵ Frank Elavsky, Cynthia Bennett, and Dominik Moritz, “How Accessible Is My Visualization? Evaluating Visualization Accessibility with Chartability,” *Computer Graphics Forum* 41, no. 3 (2022): 57–70.
- ⁶ See Overlay Fact Sheet at <https://overlayfactsheet.com/>.
- ⁷ Sheri Byrne-Haber, “Context Is the Most Critical Aspect of Alt-Text Everyone Seems to Miss,” UX Collective (Medium blog), October 10, 2020, <https://uxdesign.cc/context-is-the-most-critical-aspect-of-alt-text-everyone-seems-to-miss-e18803a79212>.
- ⁸ Louise Hickman and Alexa Hagerty, “Standardised Access: The Tension between Scale and Fit,” Ada Lovelace Institute blog, May 24, 2021, <https://www.adalovelaceinstitute.org/blog/standardised-access-tension-scale-fit/>.
- ⁹ bell hooks, *Teaching to Transgress Education as the Practice of Freedom* (London: Routledge, 2021).
- ¹⁰ Frank Elavsky, Cynthia Bennett, and Dominik Moritz, “How Accessible Is My Visualization? Evaluating Visualization Accessibility with Chartability,” *Computer Graphics Forum* 41, no. 3 (2022): 57–70.
- ¹¹ See Sun Yuan and Peng Yu’s 2016 art installation, “Can’t Help Myself,” at the Solomon R. Guggenheim Museum in New York, <https://www.guggenheim.org/teaching-materials/teaching-modern-and-contemporary-asian-art/sun-yuan-%E5%AD%99-%E5%8E%9F-and-peng-yu-%E5%BD%AD-%E7%A6%B9>.
- ¹² “Visa Chart Components,” Visa Developer Center, accessed October 19, 2022, <https://developer.visa.com/pages/chart-components>.
- ¹³ “Highcharts for Accessibility,” Highcharts, accessed October 19, 2022, <https://www.highcharts.com/blog/accessibility/>.
- ¹⁴ See Datawrapper’s website at <https://www.datawrapper.de/>.
- ¹⁵ Lauren Jong, Emily Vontsolos, and Blake Valenta, “A Template for Accessible Data Visualizations.” Medium, San Francisco Digital Services, 13 May 2022, <https://medium.com/san-francisco-digital-services/a-template-for-accessible-data-visualizations-ca2ed52f945b>.
- ¹⁶ Chris DeMartini, “A Tableau Accessibility Journey - Part IV - Keyboard Accessibility.” DataBlick blog, August 13, 2021, <https://www.datablick.com/blog/2021/8/10/a-tableau-accessibility-journey-part-iv-keyboard-accessibility>.
- ¹⁷ Jupyter Team, “Jupyter Accessibility,” Jupyter Accessibility Working Group, accessed October 19, 2022, <https://jupyter-accessibility.readthedocs.io/en/restructure/README.html>.
- ¹⁸ Doug Schepers, “Why Accessibility Is at the Heart of Data Visualization.” *Nightingale Journal of the Data Visualization Society*, May 21, 2020, <https://medium.com/nightingale/accessibility-is-at-the-heart-of-data-visualization-64a38d6c505b>.

Designing Data for Cognitive Load



DOUG SCHEPERS

Today, we are inundated with so much information—good and bad, true and false, and everything in between—through the web, social media, television, radio, and print. Data visualizations, such as charts, diagrams, and infographics, can offer an oasis of simplicity, distilling information to shapes, colors, patterns, and words. That is data visualization at its best.

But at its worst, data visualization confuses, decontextualizes, obfuscates, deceives, or overwhelms. Worse yet, it often blocks access to the information entirely, such as a chart lacking equivalent accessibility for readers with disabilities, including blindness, low vision, or cognitive disabilities.^[1]

We owe it to ourselves and to our readers to present information in the clearest way possible. As with the plain language movement, there are human rights and social justice aspects of clear, accessible data representations. Decreasing the cognitive processing that it takes to understand a data visualization can not only make our work better but also lead to more inclusive, ethical data visualization for people with disabilities.^[2]

Improving charts for people with disabilities normally improves their usability for everyone. Cognitive Load Theory (or CLT) is one tool that helps us consider mental processing when creating material. In this chapter, we'll apply CLT to charts by looking at its different aspects through the lens of specific disabilities; we'll also look at CLT in chart animation.^[3]

Cognitive Load Theory

CLT is a teaching methodology that originated in the 1980s to examine how students learn. Researchers approached learning from a neurological perspective, considering how the brain's constraints could reduce effective learning. In general, CLT does not focus on data visualization or people with disabilities, but we can apply many of its concepts.

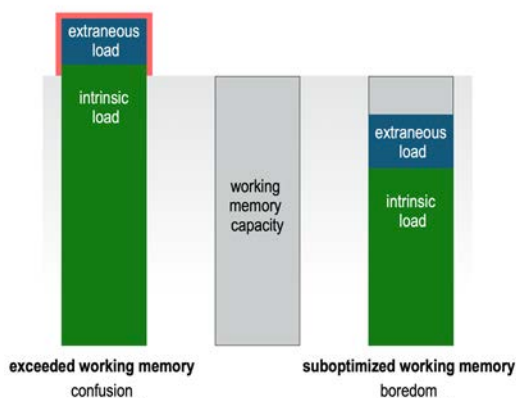
CLT is based on the Atkinson-Shiffrin model of memory,^[4] which describes three types of memory: sensory registers, working memory, and long-term memory. Whereas sensory registers are the external inputs we observe, and long-term memory is the fragments of those experiences we hold on to, working memory

is the pathway and the bottleneck between the two. Tasks like learning new material or interpreting a chart use up the limited capacity of working memory. CLT seeks to understand our memory's capacity by categorizing a task into its intrinsic and extraneous load.^[5]

Intrinsic load is the inherent nature of the information, refined to its bare minimum.

Extraneous load is how the information is structured and presented.

Simplifying is not always enough

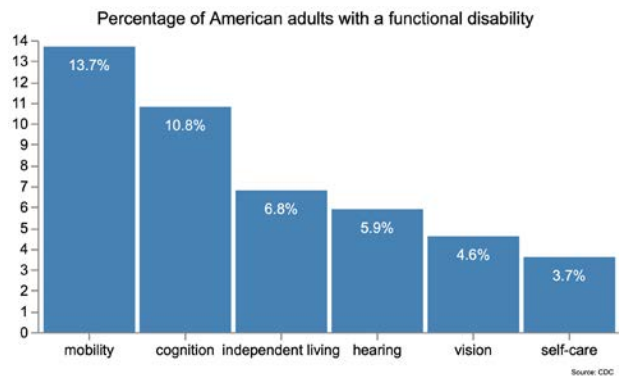


How intrinsic and extraneous loads fit the capacity of our working memory.

Both carry related elements that our working memory must process and our long-term memory seeks to store away. As content creators, our goal is to optimize intrinsic load and reduce extraneous load so the product fits the capacity of our working memory. This process aligns with Edward Tufte's popular (though often criticized) contemporary advice in the data visualization world to increase the data-to-ink ratio and reduce chart junk.

Information flows both ways between short-term memory and long-term memory. Concepts familiar to the reader (that is, already present in their long-term memory) can decrease this load, as new information is bundled into existing mental models in a process called chunking and automating.^[6]

Picture a bar chart, like this one showing the share of American adults with a functional disability. At some point in your life, you didn't know how to read a bar chart, or even know that its different parts had



A bar chart. Because most of us are so familiar with this data structure, the information it conveys is very apparent to us.

meaning. Later, you learned about the different axes, how values were represented as the height of the bars, and how to compare the bars to understand the relationships in the data. Each of these elements occupied space in your working memory. As you became more familiar with the bar chart, you no longer needed to expend effort to read it. You had chunked the different parts of the chart into their individual roles, and now you automatically read and apply new bar-chart data to this schema. The concept of the bar chart no longer occupies your working memory, and you can concentrate just on the novel data.

Now that we understand CLT, we can begin to explore its many principles and applications. I find the following are particularly relevant to data visualizations:^[7]

- **Multimedia principle:** Use words and pictures because both are more effective than words alone. A reader is more likely to remember and recognize data if they are presented as a bar chart rather than as a dense table.^[8]
- **Contiguity principle:** Present words and pictures simultaneously rather than successively. We describe this strategy below in our discussion of direct labeling.
- **Coherence principle:** Exclude extraneous words, sounds, or pictures. Adding elements to a chart, including unnecessary labels or visual distractors, reduces the speed with which a reader comprehends and retains the core information presented.

- **Interactivity principle:** Allow learners to control the presentation rate. This principle is especially relevant in the discussion of animation, which we turn to later.
- **Signaling principle:** Emphasize key steps in the representation. We describe this concept below in our discussion on highlighting and annotation.

Simplifying Is Too Simplified

It's natural to conclude that optimizing intrinsic load and reducing extraneous load is just a fancy way of saying we should simplify our charts, diagrams, and maps. That's good advice, but it doesn't always work out that way. We need to consider our context, our audience, and their attention and engagement.

Simplifying works well for a reader who is unfamiliar with the topic of a chart or who is under stress (such as someone taking a timed test). You can reduce the amount of information or break it down into manageable steps with additional context (extraneous load). You can also use common chart types your audience is familiar with.

But for a reader who has already developed a mental model of our topic with time for exploration, we should make our chart dense with salient information. A more information-dense chart might use familiar symbols as shorthand, include more data, or use novel chart types that highlight relationships an expert user can spot.

Both novices or experts have an upper limit to effective information density. Yoghoudjian and colleagues explored brain activity among 22 study participants as they viewed complex node-edge graphs (i.e., network diagrams) and found that cognitive load is readily apparent in their subjects when they're asked to complete basic comprehension tasks.^[9] The more complex the chart, the more mental effort was recorded, up to a threshold where the mental strain reduced because the reader gave up and disengaged from the task. Yoghoudjian's study demonstrates that when creating our charts, we need to keep in mind the audience's capacity to engage with the material.

CLT and Disabilities

Cognitive load affects everyone, but the challenge can be amplified for people with disabilities. A small distraction can derail a person with Attention Deficit Hyperactivity Disorder (ADHD) or a disability affecting memory. Too much unstructured textual detail can bewilder a reader who is blind or has low vision. Separating related visual elements, like a line and its label, can disorient a person with low vision. Dense information or large blocks of text can be difficult to comprehend and absorb because of fatigue, a complication of many neurological conditions such as traumatic brain injuries and Multiple Sclerosis.

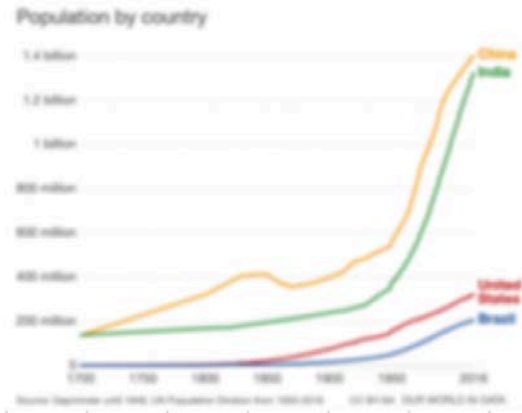
Understanding how CLT can affect people with different disabilities allows us to make more accessible, inclusive, and generally usable data visualizations. Although concepts can apply differently depending on a user's specific disability, our goal is to provide a brief tour of some of the more readily apparent instances.

As we examine different principles of CLT through the lens of specific disabilities, we need to be mindful that each principle may have broader relevance to other conditions. A person may have multiple impairments, and disabilities are often intersectional with other social factors, such as low income or educational disparities, which can increase overall cognitive load. Merely addressing one type of disability may not produce an equivalent and equitable experience for all users.

Low Vision

People with low vision can experience loss of central vision, decreased peripheral vision, reduced contrast sensitivity, or decreased ability to see small details, all of which can make reading text or finding patterns in numeric tables much harder than recognizing patterns in shapes, lines, or blocks of color. Compare the amount of information available in a blurry spreadsheet with a blurry line chart in the two images on the next page. Ascertaining the basics of the patterns is much easier for the line chart than for the table, which requires detailed examination of the numbers in the cells.

		Brazil	China	India	United States
1	2000	175,287,852	1,283,198,376	1,053,092,882	281,862,764
2	2001	177,752,872	1,282,827,820	1,071,477,888	284,852,416
3	2002	181,151,228	1,288,848,828	1,089,827,124	287,338,848
4	2003	182,482,180	1,308,243,828	1,108,227,864	289,827,816
5	2004	184,738,448	1,314,227,852	1,128,138,888	292,328,828
6	2005	186,917,360	1,321,823,852	1,148,118,888	294,828,864
7	2006	189,212,400	1,329,228,888	1,167,277,728	297,327,828
8	2007	191,328,840	1,338,823,812	1,178,881,280	299,828,220
9	2008	192,878,224	1,344,418,232	1,187,148,888	301,378,248
10	2009	194,888,220	1,352,288,288	1,214,272,888	308,378,288
11	2010	196,788,272	1,358,728,128	1,232,888,728	308,841,408
12	2011	198,888,888	1,367,482,152	1,247,228,888	311,281,280
13	2012	201,288,882	1,375,188,882	1,263,288,888	313,228,424
14	2013	202,488,824	1,382,728,216	1,278,882,288	315,828,872
15	2014	204,212,128	1,388,112,228	1,283,888,228	317,718,784
16	2015	205,882,112	1,387,228,480	1,288,228,852	318,828,152
17	2016	207,882,888	1,403,828,888	1,324,171,284	322,178,888



Because these images are blurred, we can much more easily see the overall trends and relationships in the line chart than in the table.

For people with low vision, charts often convey information with less effort than tables or text, which decreases cognitive load. Reducing these elements from several individual numbers in a column to a single line enables a person who uses a screen magnifier to track value changes while zoomed in or to see the whole trend while zoomed out. This data presentation significantly benefits readers with low vision. Tactile graphics can benefit blind readers and readers with low vision because visually impaired people can process many types of graphics much better by touching a texture than by listening to a series of numbers.

Another technique that aids users with low vision is directly labeling each line in a chart rather than using a separate legend. With a legend, readers have to locate it, zoom in, memorize the symbols and labels, zoom out to find those symbols, then identify and recall them. With direct labels, they can zoom in directly on the line of interest and pan to the end to

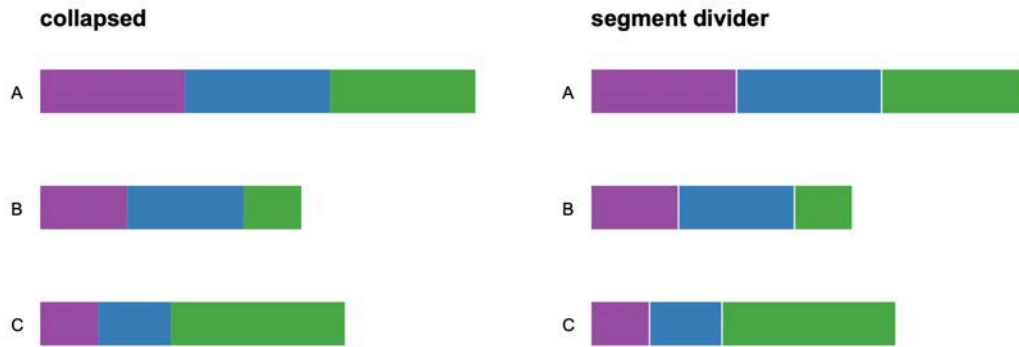
find its label (or zoom in on the label and pan back along the line). Direct labeling also enables magnifier users to know the final relative values of each line while reading the labels, which helps them construct a mental model of the data. (For an example of direct labeling, see the section below on divided-attention tasks.)

Tactile Graphics

Tactile graphics use touch to convey information. You may have experienced these as 3D exhibits in a museum or as a touch-screen device that vibrates when you move a finger over a particular spot. They're commonly used in education and science for people with visual disabilities and have proven very effective.^[10] A variety of techniques are available for making tactile graphics. Static physical graphics can be created as shapes with a 3D printer or as raised symbols on paper with a specialized embossing printer or special capsule paper that swells when heated. Digital tactile images can be created to use the vibration motors in mobile devices, or content can be designed to work with assistive technology like refreshable pin displays or braille displays.^[11]

Low Contrast Vision

A common low-vision characteristic is decreased contrast sensitivity. Colors may appear muted, and the borders between different blocks of color may appear ambiguous. An easy way to help people differentiate between items while reducing cognitive load is to use separation. We can do this by breaking tasks into bite-sized chunks conceptually and visually with a thin border between the slices in a stacked bar chart (see pair of charts on the next page). This approach works best for discrete data rather than continuous data, but with a little creativity, the principle can be applied to both.



The chart on the right is much easier to process visually because of the thin breaks in the bars. Our brains need to work a bit harder to find the breaks in the left chart.

Blindness

When visually impaired readers hear a data table through a screen reader, each number in the row or column is read to them sequentially. The reader not only has to remember the previous number, they must also use their working memory to perform calculations on the magnitude and direction of change, repeating the process for each new number.

A chart can also expose raw numbers in a structured way (e.g., with ARIA markup to make it accessible to a screen reader; see Chapter 5 on ARIA labels), but this has the same problem as a data table: it's a flood of numbers. To address this problem, you should provide meaningful alternative (alt) text that describes the most salient information, preferably with a detailed summary as the extended description. The raw numbers are still important, because a blind reader may want to delve into details and examine specific values, but you should let people decide whether they want to spend the time walking through the data after first hearing the summary. Respect your reader's time and attention.

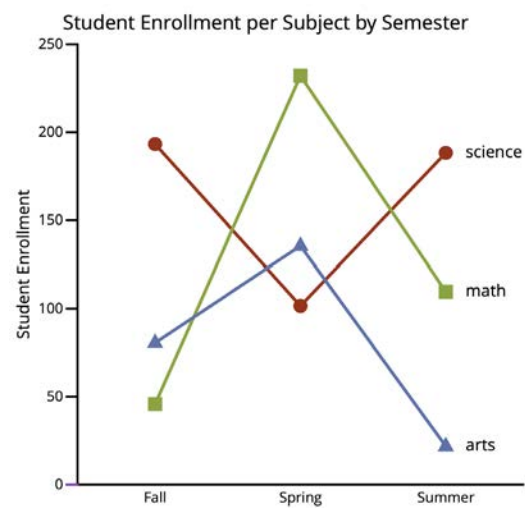
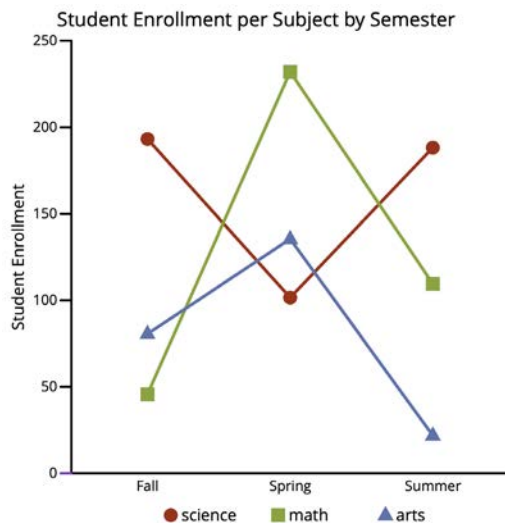
Dyslexia and Dyscalculia

People with dyslexia and dyscalculia can often understand data visualizations more easily than numeric data, such as tables or lists of numbers. But with both conditions, text labels may not be as useful as for most people, and problems with directionality and orientation may arise.

Dyslexia is a reading disorder that often includes difficulties with visually processing symbols and sometimes with reading or interpreting charts or diagrams. For people with dyslexia, flow charts are often ideal for explaining procedures, and icons or pictograms can help more than words alone.

Dyscalculia is a disability relating to numbers and mathematics. It may include difficulty with symbols, quantities, mathematical reasoning, memory tasks, or sometimes visual-spatial interpretation. For people with dyscalculia who don't have problems with visual-spatial interpretation, charts can be very helpful in processing mathematical or numeric concepts and content, whereas data tables might prove more challenging. Many people with dyscalculia can struggle with graphical timelines, so time-series data may be more challenging than other types of data.

In general, if your chart is explanatory rather than exploratory, you should consider writing a summary that reinforces its conclusion, with explicit projections or recommended courses of action based on the data. Symbols in line charts, scatterplots, and maps should be unique and not rely on orientation alone to distinguish between them. Where possible, avoid close-packed graphical elements—space them out to make them distinct and reduce visual confusion. Grid lines and guiding lines are helpful, especially if they are interactive, but don't let them visually overwhelm the core data representation.



On the left, a line chart with labels in a legend/key. On the right, a line chart with direct labeling near the lines.

Attention Management Disabilities

ADHD, or Attention Deficit Hyperactivity Disorder, is misleadingly named. Someone with ADHD does not have a decrease in attention itself (because they commonly exhibit hyperfocus); rather, they have a decreased ability to direct their attention to the most important task at hand.

People with ADHD are not the only readers affected by attention management disabilities—typical aging introduces several deficits in memory management as well.^[12] People with attention management disabilities constitute a large demographic audience.

As you work with data and charts, consider your reader's capacity for selective and sustained attention, and decrease tasks that require divided attention. These changes might include reducing the amount of detailed information in a chart or diagram, reducing the number of annotations and labels, and lowering the complexity of the visualization itself. To better understand how we can tailor our visualizations for readers affected by attention management disabilities, we need to understand the different types of attention that visualizations require.

Divided Attention

Divided-attention tasks require the reader to process more than one source of information or to perform more than one task at the same time. Notice the difference in the two graphs at the top of this page—

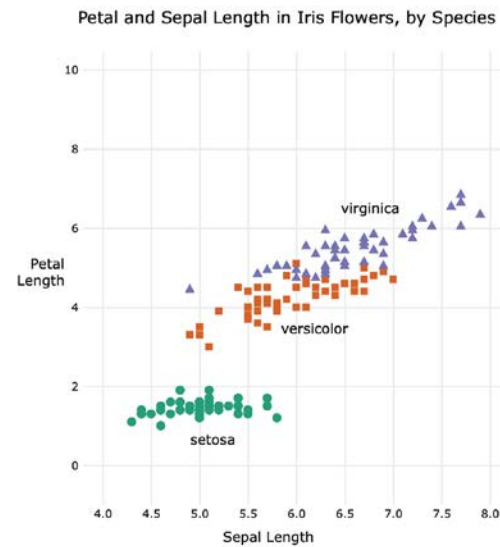
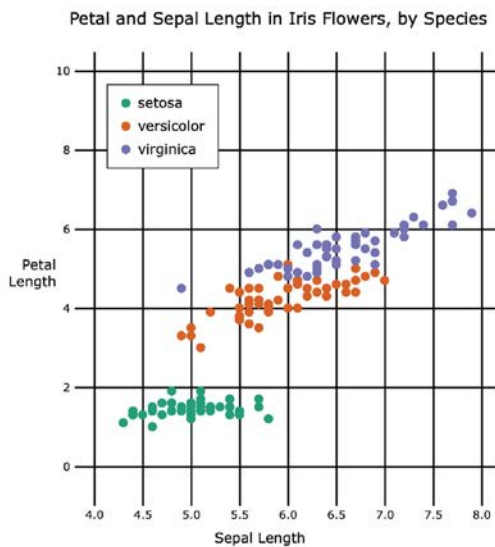
one has a legend, separated from the content; the other has labels directly on the chart next to the ends of the lines. Splitting the reader's attention between the data representation and the legend means extra mental processing. They need to commit each symbol and color to their working memory, then shift their view over to the line to interpret the data while also applying the memorized legend. This effort may seem trivial, but it imposes an unnecessary memory tax that has been demonstrated to decrease the speed and accuracy of comprehension and retention of information.^[13]

The direct labeling is also significantly easier for readers with low vision to process, and it benefits people with color vision deficiency (CVD, or "color blindness"), who can use label proximity rather than color to identify and distinguish the lines.

Selective Attention

Selective attention is the reader's ability to focus on stimuli that are relevant to the task at hand while disregarding irrelevant stimuli. You can improve selective attention with a few simple techniques, all of which are demonstrated in the pair of images on the next page:

1. Decrease, remove, or deemphasize unneeded visual elements using lower contrast, muted colors, or thinner line width.



On the left, a scatterplot with bold black grid lines and several symbols of the same shape with only color differences. On the right, an improved scatterplot with thin gray gridlines and several symbols of different shapes and colors.

2. Make relevant data marks of different categories more easily distinguishable.
3. Move labels from a dedicated legend to direct labels on data points or clusters of points.

As with direct labeling, using data symbols with different shapes and different colors benefits people with CVD as well.

In addition to reducing extraneous load, you can optimize intrinsic load by directing the reader's attention to the most salient parts of the data, especially to reflect the intent of an explanatory chart. You can do this with text descriptions (like alt text) or with highlighting and annotation, which is also particularly useful for people with low vision. This selective emphasis draws on the signaling principle of CLT.

Sustained Attention

Sustained attention is the reader's ability to focus their concentration on a task over an extended period. Charts that require the reader to perform multiple steps often necessitate sustained attention.

If you're making such a chart, consider what tasks you expect the reader to perform and whether they

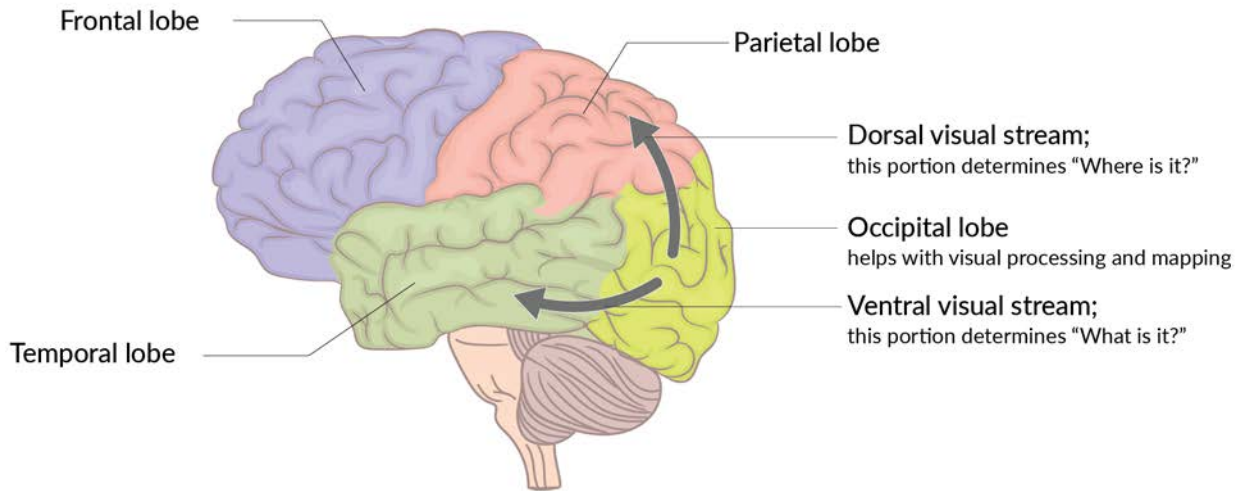
require all the data to be in a single chart or if the chart could be split into multiple charts, each focusing on a specific task. If the set of tasks cannot be broken up efficiently, you may also consider adding a layer of interactivity to help the reader concentrate on a subset of the tasks or offer staged states with feedback on progress.

Meaningful animation may also help the reader sustain attention during prolonged or tedious tasks, such as monitoring a chart for status changes. Animation is especially effective when it draws attention to significant changes, dramatic shifts, or triggered thresholds.

How Animation Can Decrease (or Increase) Cognitive Load

Meaningful animation—through the use of videos, animated GIFs, or other means—can be a powerful aid to cognitive accessibility. Animation is one of the most literal representations of change over time and can help engage readers.

Neurologically, our brains process and interpret movement differently, prioritizing it over other visual stimuli. Because movement captures the reader's eye so effectively, animation can be overused, which is



How information proceeds from the occipital lobe to the dorsal visual stream or ventral visual stream.

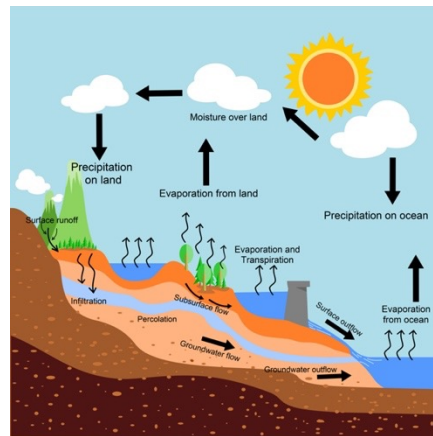
why it should be considered carefully and thoughtfully. The brain has two pathways for processing visual information: the "what" and the "where" pathways. The "what" pathway leads to identification and recognition, and the "where" pathway processes the object's spatial location. The "where" pathway is faster and more immediate, which animation triggers by grabbing and keeping the reader's attention. We need to make sure the content we're animating is worthy of this special consideration by the reader.

Effective animation aids focus rather than disrupting it. Chart animations should be functional and brief. Animation can also engage the reader's emotions, either by creating fun or by imparting causality or agency to the animated objects.^[14] The advantages of animation come with a caveat, however: it can take longer to interpret, which can cause the reader to make mistakes in analysis of the data.^[15]

Good use cases for animation in data visualizations can be grouped into three broad categories.

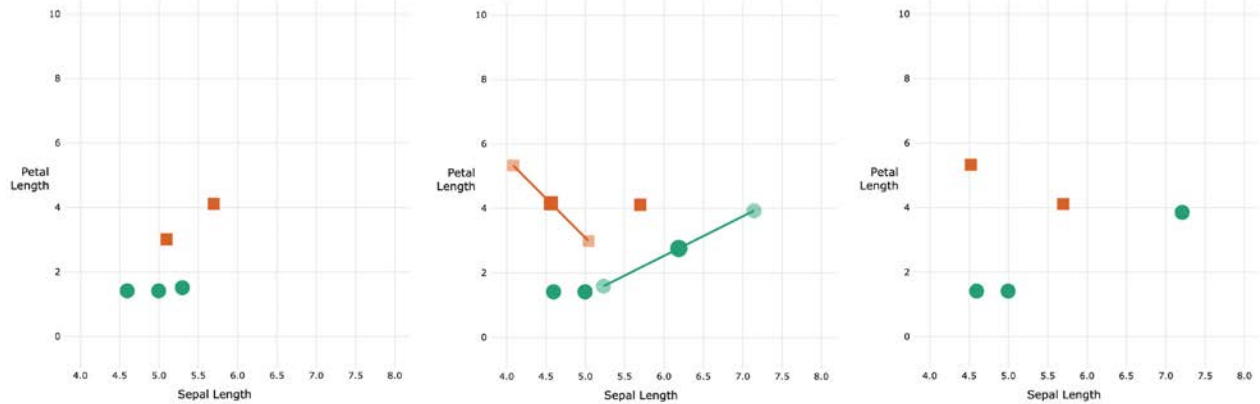
Demonstrating progression. In more literal diagrams, where figures represent real objects like the water cycle, animation can show the changing relative locations of those objects over time. In more abstract line charts, it can reveal the progression of a trend, which can also create emotional engagement. In some

cases, it can even illustrate the actual pace of change, either in real time or scaled.



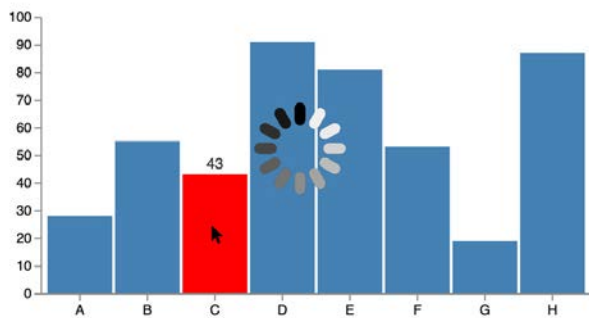
An example of how animation can be used to show real changes in location.

State changes. In interactive charts, visual elements often transition from one state to another, such as when the data on a chart are changed, zoomed, or filtered. Showing this transition as literal movement helps the reader stay oriented and track trends. Without animation, the data changes can be hard to recognize.



Three versions of a scatterplot, where some of the data points move from the start in the first scatterplot to the end in the third scatterplot, with a transition between the two states in the second scatterplot. The movement path of each point is represented by an arrow.

Feedback. Sometimes a reader needs to know immediately that their actions have had an effect in interactive charts. If the computing task takes longer than a few milliseconds, such as with an asynchronous data fetch or a processor-intensive operation, the user may try to trigger the action again or think it's not working. Providing an animation can assure the reader their action has registered.



An example of providing feedback to a user.

Best Practices for Using Animation in Visualizations

Animation is safest when the user has control over it. Data practitioners should respect the user's preferences for when to play animations, the speed of animations, and the ability to manually trigger animations. I've highlighted four best practices for making animations useful (and safe) for all users, especially those who may have a condition that limits their ability to use or perceive visual information.

1. Deactivate Animations by User Preference

For readers with some cognitive disabilities, animation can distract or even be dangerous to them, such as by triggering seizures.

On the internet, you can detect and honor the reader's animation preferences. Programming languages for the web like CSS and JavaScript have the concept of a "media query," which allows a website to detect features of the user's browser, such as whether the user is using a touchscreen rather than a mouse, the written language the user has configured, and settings related to accessibility. One of these settings is the "prefers-reduced-motion" media query. If the user has indicated they prefer reduced motion, you can eliminate the animation or significantly reduce the speed or range of motion. For users who have not set their browser's animation preferences, you can also provide a website-specific option for users to turn off the animation.

2. Give Control of Animation and Processing Speed

Readers with memory or attention disabilities may not process information at the same pace as your animation.^[16] The same goes for people who are distracted. Consider giving the reader control over the playback speed or the ability to restart the animation. Allowing readers to interact directly with the animated element can also be effective.

3. Reduce the Number of Animated Elements

Having more than a couple of animated elements can overload working memory, leaving the reader no capacity to process and remember the changes before the animation ends.^[17] As a result, you should either ensure the animating objects move in a similar pattern to one another or reduce the number of animated objects.

4. Announce or Describe Animations for Blind Readers

If you do use animations in a meaningful way, be sure to provide a description or a live sequential text update for blind readers, such as with the “aria-live” attribute (see Chapter 5 on ARIA labels). This addition is especially important for users receiving feedback after interacting with a visualization.

Conclusion

As data professionals, we often rely on quantitative measures of chart effectiveness, through A-B testing, task assessment, or another means of using data; we do this because we want to solve problems with certainty. But the world has never been a hospitable place for certainty. Embracing the ambiguity of how people approach our data visualizations will often lead us to create more expansive, accessible products.

A guiding rule is to be mindful of the different audiences, expectations, tasks, and provenance for your chart, which can affect how you approach accessibility.^[18] Consider if your chart is for explanation or exploration; if it’s static, dynamic, or interactive; and if there are alternative ways to offer equivalent representations.

Applying some of these guidelines will help many people access and process data visualizations more effectively, although it will help some people more than others. As a data visualization practitioner, you can’t reach perfect accessibility, but you can and should experiment with making your visualizations as useful as possible for as many people as possible. You’ll be doing good, and doing good is often good enough.

Chapter Three Notes

- ^[1] W3C Cognitive Accessibility Task Force, World Wide Web Consortium, draft from September 21, 2021, <https://w3c.github.io/coga/user-research/>.
- ^[2] Ashley M. St. John, Melissa Kibbe, and Amanda R. Tarullo, "A Systematic Assessment of Socioeconomic Status and Executive Functioning in Early Childhood," *Journal of Experimental Child Psychology* 178 (2019): 352–368. <https://pubmed.ncbi.nlm.nih.gov/30292568/>.
- ^[3] For readability, I'll use the term "chart" generically to refer to data visualizations, and I'll explicitly call out advice diagrams, maps, infographics, and so on.
- ^[4] Richard C. Atkinson and Richard M. Shiffrin, "Human Memory: A Proposed System and Its Control Processes," in *Psychology of Learning and Motivation* volume 2, ed. Kenneth Spence and Janet Taylor Spence (New York: Academic Press, 1968), 89–195.
- ^[5] John Sweller, "Cognitive Load during Problem Solving: Effects on Learning," *Cognitive Science* 12, no. 2 (1988): 257–285; and John Sweller, Jeroen J. G. Van Merriënboer, and Fred G. W. C. Paas, "Cognitive Architecture and Instructional Design," *Educational Psychology Review* 10, no. 3 (1998): 251–96.
- ^[6] John Sweller, "Cognitive Load Theory, Learning Difficulty, and Instructional Design," *Learning and Instruction* 4, no. 4 (1994): 295–312.
- ^[7] Richard E Mayer, "Cognitive Theory and the Design of Multimedia Instruction: An Example of the Two-Way Street between Cognition and Instruction," *New Directions for Teaching and Learning* 2002, no. 89 (2002): 55–71.
- ^[8] See also Joseph R. Jenkins, Daniel C. Neale, and Stanley L. Reno, "Differential Memory for Picture and Word Stimuli," *Journal of Educational Psychology* 58, no. 5 (1967): 303–07.
- ^[9] Vahan Yoghoudjian, Yalong Yang, Tim Dwyer, Lee Lawrence, Michael Wybrow, and Kim Marriott. "Scalability of Network Visualisation from a Cognitive Load Perspective," *IEEE Transactions on Visualization and Computer Graphics* 27, no. 2 (2020): 1677–87.
- ^[10] Jordan C. Koone, Chad M. Dashnaw, Emily A. Alonzo, Miguel A. Iglesias, Kelly-Shaye Patero, Juan J. Lopez, Ao Yun Zhang, et al., "Data for all: Tactile Graphics That Light Up with Picture-Perfect Resolution," *Science Advances* 8, no. 33 (2022).
- ^[11] see Fizz Studio's open-source SparkBraille as an example: <https://fizzstudio.github.io/sparkbraille/>.
- ^[12] Elizabeth L. Glisky, "Changes in Cognitive Function in Human Aging," in *Brain Aging: Models, Methods, and Mechanisms*, ed. David R. Riddle (Boca Raton, FL: CRC Press/Taylor & Francis, 2007): 3–20.
- ^[13] K.N. Purnell, R.T. Sollman, and John Sweller, "The Effects of Technical Illustrations on Cognitive Load," *Instructional Science* 20, no. 5–6 (1991): 443–62.
- ^[14] Brian J. Scholl and Patrice D. Tremoulet, "Perceptual Causality and Animacy," *Trends in Cognitive Sciences* 4, no. 8 (2000): 299–309.
- ^[15] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. "Effectiveness of Animation in Trend Visualization," *IEEE Transactions on Visualization and Computer Graphics* 14, no. 6 (2008): 1325–32.
- ^[16] Kevin S McGrew, "CHC Theory and the Human Cognitive Abilities Project: Standing on the Shoulders of the Giants of Psychometric Intelligence Research," *Intelligence* 37, no. 1 (2009): 1–10.
- ^[17] Steven L. Franconeri, Lace M. Padilla, Priti Shah, Jeffrey M. Zacks, and Jessica Hullman, "The Science of Visual Data Communication: What Works." *Psychological Science in the Public Interest* 22, no. 3 (2021): 110–61.
- ^[18] See, for example, Jonathan Schwabish, "Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks" (New York: Columbia University Press, 2021).



PART THREE

Alternative (alt) Text
and Screen Readers

Writing Alt Text to Communicate the Meaning in Data Visualizations



ELIZABETH HARE

Innovative data visualization methods are increasingly found in many kinds of media, including news websites, government information portals, financial statements, and scientific communication, frequently with no description for users of screen reading technology.^[1]

And even when alternative texts (alt text)—text descriptions that convey the content and meaning to blind and low-vision readers—are available, they often fail to convey the main takeaways of the data.^[2]

For data visualizations that are produced as graphic files (e.g., JPGs, PDFs, and PNGs), alt text needs to be added. These files cannot be navigated with screen reading software, and although some methods of creating data visualizations can produce readable elements for screen readers (for example, Visa Chart Components, SAS Graphics Accelerator, and others reviewed by Elavsky and colleagues),^[3] most statistical software does not produce accessible charts and graphs. That responsibility falls to the author. Elavsky and colleagues summarize the situation well: “Accessibility is still an afterthought in data visualization.”^[4]

Graphs and charts follow conventions to orient the reader to the displayed data, including coordinate axes, axis labels, number labels, legends, and titles. These introduce the variables and their characteristics, allowing the reader to comprehend the data points, trends, and differences within the graph.

Several models have been proposed for thinking about getting the information in graphs and charts into alt text. This article explains two of those models: Canelón and Hare’s “four-ingredients model” for writing complete alt texts and the MIT Visualization Group’s “four-level model” for classifying types of information from graphs.^[5] I describe both models in the following section and demonstrate how they are related to one another across a pair of examples—a simple line chart and a less familiar tree map.

The Four-Ingredients Model

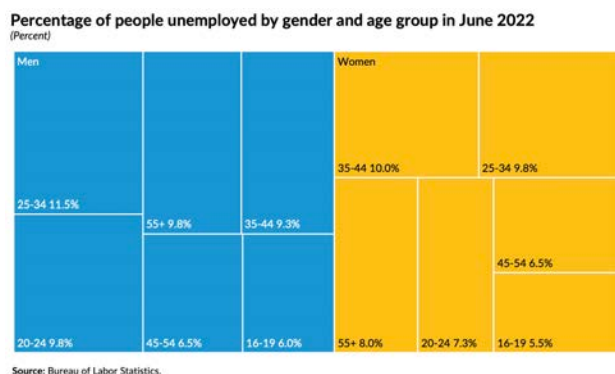
In 2021, Silvia Canelón and I coauthored a study in which we examined the frequency of alt text inclusion and the completeness of alt texts for data visualizations in an online learning community.^[6]

We identified four ingredient questions necessary for comprehension of a graph. We based this “ingredients model”

on the verbal descriptions of scientific graphs from sighted colleagues and our personal experience reading alt texts online. As a result of this work, we developed four questions for any data practitioner to ask when thinking about writing alt text.

1. What Kind of Graph or Chart Is It?

Beginning alt text with a description of the type of chart or plot helps the reader anticipate the presented information. Authors use common types of data visualizations such as bar charts, scatterplots, line plots, and pie charts to show trends like the shape of data, changes over time, a statistical distribution, or a comparison of two or more groups. Recent innovations in data visualization have led to more types of charts, and these deviations from more traditional types need to be described in detail if they contribute to the meaning of the graph.^[7] A tree map, for example, is a square or rectangular graph divided into groups to illustrate hierarchical part-to-whole relationships.^[8] The tree map below shows the share of people who are unemployed, divided by two gender and six age groups. Men are shown in the blue rectangles and women in the yellow. New and specialized charts and graphs like tree maps should be described more extensively than simpler ones, and guidelines for the description of each type should be developed.



In the overall description, alt text should also mention whether the figure is just one plot or a set of multiple graphs (also known as trellis plots, faceted plots, or small multiples). Well-organized alt text for a graph with small multiples would describe the common aspects of all the graphs first, then describe each

facet or subplot (with its variable name). When small multiples are arranged in a grid, start at the upper left and work across rows as you would when reading regular text.

2. What Variables Are on the Axes?

When describing the variables in alt text, data analysts need to keep a couple things in mind. It's always best practice to use the actual variable names that are present in the chart. Relying on descriptions of colors or textures for identification can lead to confusion or distract from the meaning. Research shows some people report that keeping track of colors from one part of alt text to another leads to a difficult cognitive load (see Chapter 3 in this volume).^[9] Others, however, would like to form a mental image of the graph as it appears and appreciate knowing the colors.^[10] No matter the preferences of the user, including the actual variable name and units will lead to better, more coherent alt text.

3. What Are the Ranges of the Variables?

As with any chart, the variables present exist over a limited range. Describing these ranges will provide an overview of the measurements displayed and context for understanding the relationships between variables. Including the ranges in the alt text also gives an indication of the order of magnitude of the observations (e.g., whether they are numbers or percentages).

4. What Does the Appearance Tell You About the Relationships Between the Variables?

To answer ingredient four, data analysts usually need to answer two important questions: "Why am I including this visualization in the media I am creating?"^[11] and "What are these data saying?"

These answers can lead to a description of what the graph shows about the data, which will vary by chart type. In a line graph, for example, the description would include whether the line is straight, curved, or jagged; where maximum and minimum values occur with respect to the x-axis; and the slope. To describe a scatterplot, we would indicate whether the distribution of points seems even over the whole plot, if there are

clusters, or if the density of points differs in specific areas of the plot. Plots of statistical distributions can often be described with reference to the normal curve, though that will depend on the anticipated knowledge of the target reader. For graphs that represent distributions, we would include the relative size of the tails, how steep the sides of the curve are, and the symmetry of the distribution.

In current practice, many data visualizations on the internet, in social media, or in the scientific literature do not contain all four ingredients and do not adequately describe what the graph is showing. We found that 3 percent of practice data visualizations contained alt text and, of these, only 34 percent included the answer to this fourth ingredient.

Four-Levels Model

The MIT Visualization Group, a team of researchers working to better understand how “computation can help amplify our cognition and creativity,” recently analyzed feedback on alt text from blind users. Based on reader responses, they categorized meaningful alt text into four levels. These levels are useful for considering what to include and what to leave out and offer guidance on alt text automation and ethics.^[12]

Level 1

The first and lowest level contains sentences that describe components “foundational to visualization construction—comprising the elemental properties of the ‘language’ of graphics.” These components include the title, legend, axis names and scales, and chart type (e.g., scatterplot, line graph, bar chart). At this lowest level, there is no “synthesizing or interpretation.”^[13]

The first three items in the ingredient model described above are all parts of Level 1.

Level 2

The second level includes statistical facts that are available from the dataset (e.g., mean, median, range, maximum, minimum) as well as quantitative relations (greater than, less than, or equal), and individual data points. Some of this information is available in the graph, but the rest is obtained from the underlying

dataset.^[14] The information at Level 2 that is available in the graph is a subset of the fourth ingredient from the ingredient model.

Level 3

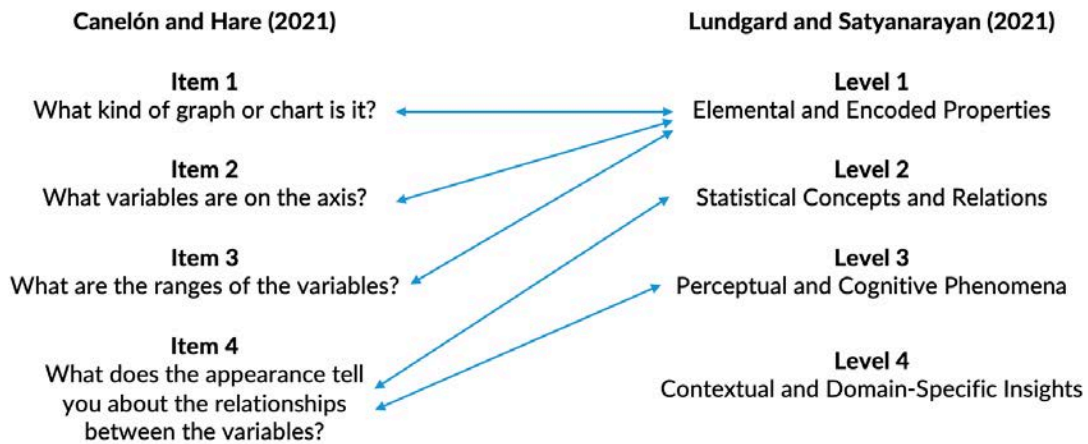
The third level contains “perceptual and cognitive phenomena.” These components of graphs are identified by inspecting the data and identifying the trends, outliers, or patterns. This level of information relies on the cognition of the reader and may be more complex than information at Levels 1 and 2. These elements may not be readily observed by reading raw data points, summary tables, statistics, or other nonvisual data summaries. The essential meaning of the data visualization is found at this cognitive and perceptual level of description. Level 3, like Level 2, is part of the fourth ingredient of the ingredient model.

Level 4

The fourth and highest level contains information (or opinions) based on contextual or domain-specific knowledge. The examples given rely on knowledge from outside the graph and make assumptions about causation.^[15] Level 4 content is not considered part of the ingredient model, which prioritizes objective reporting of the chart over subjective interpretations of application. This subjective information from outside the graph should not be included in alt text. Blind readers want the information and trends but not extra opinion or interpretation.^[16] For example, if you think you know the reason for the change in slope of a graph, don’t include it in the alt text. It should only be included if it is a printed annotation on the graph.

Comparing the Two Models

The two models I’ve described do not perfectly align, but both offer useful ways to think about writing alt text and graph descriptions. On the next page, I’ve mapped the four items (ingredients) from the Canelón and Hare model onto the Lundgard and Satyanarayan (levels) model. The first three ingredients from Canelón and Hare map to the first level in the Lundgard and Satyanarayan model, while the fourth ingredient maps to the second and third levels.

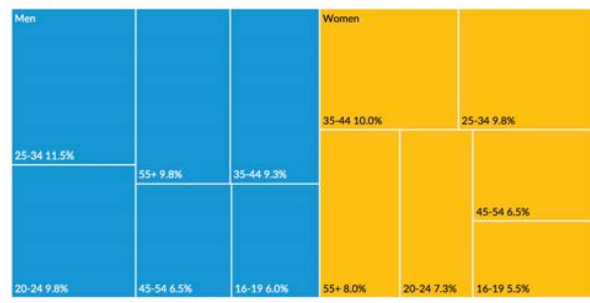


To show how the two models can differ, let's dive into two examples. First, we have a fairly basic line chart of the monthly unemployment rate in the US from January 1950 to June 2022. Most readers should recognize a line chart, so we can use less alt text to describe it. The table on the next page shows how different versions of the alt text correlate to the various pieces of the two models (in this report, the graph itself uses all four cells as the alt text).



The second example also uses unemployment rate data but uses the less familiar tree map format. Here, we focus on the share of people who were unemployed in June 2022 grouped into two gender and six age groups. In this case, the first stage of each model includes a more detailed explanation of the tree map and how it is organized.

Percentage of people unemployed by gender and age group in June 2022 (Percent)



Source: Bureau of Labor Statistics.

Length and Completeness of Alt Texts

Ideally, alt text should be just long enough to describe what the graph is illustrating visually. Although many guidelines prioritize brevity,^[17] most alt texts are incomplete and leave users with questions.

These brief alt texts provide inadequate insight into what the data are communicating. Many guidelines suggest using one or two sentences, but doing this can lead to alt texts for data visualization containing only one item from Level 1 (e.g., a title or caption) with no Level 2 or Level 3 information. Alt texts should be long enough to include essential Level 2 and Level 3 information.

To keep alt text succinct and comprehensible, avoid including decorations or “chartjunk”^[18] such as unnecessary gridlines, repetitive display of data, visual effects like shadows on bar graphs, or shimmering elements. These details do not add to the meaning of the data being displayed.

Four Ingredients model	Four Levels model	Example alt text
Line Chart		
1	1	This graph contains a blue line showing monthly unemployment rates in the US and gray vertical bars showing recessions. The source of the data is the Bureau of Labor Statistics and the National Bureau of Economic Research.
2 and 3	1	The x-axis shows the month and year from January 1950 to June 2022. The y-axis shows the unemployment rate with a range of 0 to 16 percent. The bars extend along the entire vertical area and correspond to the 11 recessions since 1950. The unemployment rose from 4.4 percent in March 2020 to 14.7 percent in April 2020, a 10.3 percentage point change. The second-highest change was a 1.3 percentage-point change between September and October 1949.
4	2 and 3	The unemployment rate line typically slopes upward during recessions, though not always, and it starts to move upward before recessions begin. The longest gap between recessions was from July 2009 to January 2020, a total of 120 months.
	4	The largest month-over-month increase in the unemployment rate during this period occurred during the most recent recession in spring 2020, resulting from the COVID-19 pandemic.
Tree Map		
1	1	A tree map, which is a series of squares and rectangles arranged inside a larger rectangle, describing the share of people unemployed by gender and age group in June 2022. Men are shown in a set of six blue boxes and women in six yellow boxes.
2 and 3	2	There are 12 boxes for each combination of gender and age, where gender is male or female and age is 16 to 19, 20 to 24, 25 to 34, 35 to 44, 45 to 54, and 55 and older. The percentages range from 6 percent for 16- to 19-year-old men to 11.5 percent for 25- to 34-year-old men.
4	3	Men account for a larger share of unemployed people than women, but the difference is barely perceptible. In June 2022, men accounted for 53 percent of all unemployed workers, though women account for a slightly larger share of the entire population.
	4	Although men between the ages of 25 and 34 have the highest unemployment rate at 11.5 percent, women ages 35 to 44 have the second-highest unemployment rate at 10 percent, possibly reflecting differential changes in the labor market as it recovers from the COVID-19 pandemic.

The context of the graphic will also influence the length of the alt text. A busy graph tracking a lot of variables will need more alt text than a simple line graph of two variables, and a graph in a scientific journal may need a more detailed description than one meant for a general audience. Authors should be aware of “the curse of knowledge,” which is the tendency for experts to prioritize elements of graphs they find important compared with nonexperts’ descriptions.^[19] In many contexts, such as education and science publications, a reader needs to have complete information in order to independently prioritize the displayed information.

Automatic Alt Text

With the rise of machine learning and artificial intelligence methods, automatic tools to produce alt text have also proliferated, but these methods always produce inadequate alt text. Whether these alt texts are generated by mining statistical code, by scanning a graphics file, or through artificial intelligence methods, information at Level 3 and the fourth ingredient will not be included. Without this information, the reader has to invest time building a mental framework for understanding the graph, possibly with elements from a title or variable names, but the alt text doesn’t provide the graph’s takeaway message. When the reader starts to hear the title and variable names, they don’t realize the automatic alt text won’t tell them the point of the graphic. If automatically generated alt text is used, it must be edited to include the Level 3 information about what the graph shows, otherwise it will be ineffective and frustrating to the reader.

Lundgard and Satyanarayan examined whether machine learning or artificial intelligence methods can produce the four levels of graphic information.^[20] Although the Level 1 elements such as chart type, title, and axis labels are often readily extracted, only the visible portion of Level 2 information emerges from a machine learning approach. Information about Level 3, which provides the “overall gist” of complex trends and patterns,” and Level 4—which requires human perception, cognition,

and interpretation—cannot be provided automatically. Because current automatic methods cannot produce alt text describing Level 3 information, they cannot produce alt text that provides meaningful access to blind readers.

For the foreseeable future, automatic alt texts will be incomplete. They can be used as a starting point if they provide useful Level 1 and 2 information, but these partial alt texts should be avoided altogether or edited to add Level 2 and 3 information. Alt texts that provide only the scaffolding without information about the building are frustrating and unhelpful. These partial alt text snippets are not “better than nothing” because the reader has invested time to build a mental model to understand the graph but never receives the critical information.

Recommendations

In addition to writing clear, informative alt text, authors can improve accessibility in many ways, including providing supplementary materials, such as the data behind a visualization, separate from the actual report or website. A new package for the R programming language,^[21] `ggdatasaver`, facilitates saving the dataset associated with data visualizations created with the `ggplot` package.^[22] Providing the raw data and tabulated summaries allows screen reader users to explore the data in a more accessible (though possibly less useful) format (see Chapter 5). When data are presented as maps, content producers should consider including a table as well (one useful example would be data on COVID-19 incidence by zip code). Including the table will provide access to the variables shown on the map but will not provide information on the spatial relationships between the geographical units.

In other contexts, supplementary materials are critical as the digital infrastructure to provide alt text is still missing. One study found that in an analysis of 300 open-science journals, there was nothing about alt text in their guidelines for authors.^[23] Most academic publishing systems have no infrastructure to include alt text along with images. One workaround option is

to provide full information on the displayed variables, trends, and patterns in the results and discussion sections of papers; another is to provide alt text as supplemental information that can be downloaded from the journal website.

Many social media platforms have infrastructure for alt text. On some platforms, alt text has a low character limit. In these situations, consider placing alt text in comments or replies and including a statement like “image description in comments” in the main body of posts.

Recognizing that all observers have some inevitable bias, alt text descriptions of data visualizations should present data as objectively as possible. Jung and colleagues report that blind users prefer that alt texts not use judgements of how large differences are.^[24] Readers do not want “subjective interpretations” or contextual information, and they do not want the author’s opinion.^[25] If a graph shows uncertainty, describe it. If numbers need to be estimated, state that. Do not conflate correlation of two variables with causation. General considerations around alt text, including punctuation and mathematical symbols, are explored in the box on the next page (and see Chapter 8 of this volume).

In their work on artificial intelligence for object recognition for blind people, Bennett and Keyes point out that these descriptions are likely to replicate the biases of human observers and programmers.^[26] But information from computers is often put on a pedestal and assumed to be correct. Whether the describer is human or artificial, a power differential between the describer and reader exists, which authors should consider.

Ultimately, writing effective, complete alt text is an important skill for authors to master in order to communicate the meaning of their data. With current limitations inherent with screen readers, machine learning, and artificial intelligence, this skill will remain necessary for the near future. Good alt texts should create a mental framework for the variables and quantities shown, include information about the

patterns found in the data, and tell a user why you chose to include the data visualization. Although many guidelines emphasize brevity, completeness is critical for equal access to information. If media infrastructure does not support alt text, data practitioners must find a way to provide it.

Strategies for Writing Good Alt Text

Alt text should provide meaningful information to the user. Of course, what is meaningful to one user may not be meaningful to another user, so there is not a single right answer to writing good alt text. You may even notice that the alt text in this report varies from chapter to chapter and author to author.

Generally speaking, alt text should be written with the same punctuation and capitalization as regular text, but again, that also will depend on the user and how the user has set up their screen reader.

Punctuation marks like periods, commas, and dashes inform the way the screen reader uses intonation and pauses to simulate natural speech, which can improve comprehension but can be problematic in certain cases. Unlike with regular text, it’s not possible to navigate by word or by character to clarify what’s being said, so it’s important to make sure the alt text can be read clearly.

There are some key aspects of alt text to consider, especially when it applies to data visualization:

- Describe the meaningful aspects of the graph, chart, or diagram. Tell the user the most important feature of the image and what they should take away from it. Provide enough context about the type of graph, variables, and quantities for the reader to understand the takeaways.

- Contain some measure of quantitative data if the figure does as well. If there is a specific data point or two you want to highlight, include it in the alt text.
- Remain brief and to the point. If you need to describe every data point, then provide an alternative format, such as a table or long description, in the appendix or elsewhere.
- Limit the amount of symbols, such as vertical bars (or “pipes”), slashes, colons, and semicolons, as well as em dashes and en dashes. These symbols are voiced by screen readers and add to the audio noise during the voicing of the alt text. Commas are usually voiced as short pauses and periods as long pauses, and many screen readers will vocalize dashes, colons, and semicolons. Writing “x divided by y” will be clearer than writing “x/y” because the slash also has a nonmathematical meaning.
- Limit the use of URLs because they are not activatable in alt text. If you must use them, spelling out “dot com” will help denote the URL.
- Capitalizing acronyms will usually force text to be read letter by letter (unless they are commonly known ones that are pronounced as words), but writing them in lowercase will usually lead to them being pronounced as words. If the alt text includes “ui” instead of “UI,” most screen readers will try to pronounce the two vowels as a word, which will certainly sound weird.
- Avoid repeating chart titles. If the title is available somewhere else in the text that a screen reader will read aloud, it shouldn’t be repeated in the alt text. (This advice differs for an image on social media, where the title is part of the image.) More generally, you can just tell the reader what the chart title is. The alt text for a line chart showing the unemployment rate might say, “A bar chart that shows the unemployment rate over time with the title, Unemployment Rate in the United States from 1950 to 2020.” In this case, the phrase “with the title” and the existing punctuation makes the title clear. If you wanted to omit the words “with the title” to save characters, quotation marks around the actual chart title might be useful. In this way, the reader could check to see if the phrase was the actual title or a description of the chart.

Ultimately, write alt text like you would write normal text. Screen readers are optimized for existing text patterns and expect alt text to be written in those forms. Users can control how much punctuation marks are spoken, but the specifics of that functionality will vary by tool and platform. Other guidelines for writing data visualization alt-texts are the DIAGRAM Center Image Description Guidelines and WGBH’s Guidelines for Describing STEM Images.^[27]

Chapter Four Notes

- [1] Crescentia Jung, Shubham Mehta, Atharva Kulkarni, Yuhang Zhao, and Yea-Seul Kim, “Communicating Visualizations without Visuals: Investigation of Visualization Alternative Text for People with Visual Impairments,” *IEEE Transactions on Visualization and Computer Graphics* 28, no. 1 (2021): 1095–105; and Jay Wickard, “Is Accessibility Part of ‘Open’ Science?,” *Practical Data Management for Bug Counters* (WordPress blog), July 13, 2021, <https://practicaldatamanagement.wordpress.com/2021/07/13/is-accessibility-part-of-open-science/>
- [2] Jung et al., “Communicating Visualizations without Visuals”; Silvia Canelón and Liz Hare, “Revealing Room for Improvement in Accessibility within a Social Media Data Visualization Learning Community,” presentation given at csv.conf.v6 (virtual conference), May 4, 2021 <https://github.com/spcanelon/csvConf2021>.
- [3] Frank Elavsky, Cynthia Bennett, and Dominik Moritz, “How Accessible Is My Visualization? Evaluating Visualization Accessibility with Chartability,” *EuroGraphics Conference on Visualization (EuroVis)* 41, no. 3 (2022), <https://www.frank.computer/chartability/#ref-USImproving>.
- [4] Elavsky, Bennett, and Moritz, “How Accessible Is My Visualization?”
- [5] Canelón and Hare, “Revealing Room for Improvement in Accessibility within a Social Media Data Visualization Learning Community”; and Alan Lundgard and Arvind Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content,” *IEEE Transactions on Visualization and Computer Graphics* 28, no. 1 (2021): 1073–83.
- [6] Canelón and Hare, “Revealing Room for Improvement in Accessibility within a Social Media Data Visualization Learning Community”
- [7] Jonathan Schwabish, *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks* (New York: Columbia University Press, 2021).
- [8] Ben Shneiderman, “Tree Visualization with Tree-Maps: 2-d Space-Filling Approach,” *ACM Transactions on Graphics (TOG)* 11, no. 1 (1992): 92–99.
- [9] See also Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [10] Jung et al., “Communicating Visualizations without Visuals.”
- [11] Amy Cesal, “Writing Alt-Text for Data Visualization,” *Nightingale Journal of the Data Visualization Society*, July 23, 2020, <https://medium.com/nightingale/writing-alt-text-for-data-visualization-2a218ef43f81>.
- [12] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [13] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [14] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [15] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [16] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content”; and Jung et al., “Communicating Visualizations without Visuals.”
- [17] See “General Guidelines,” *Diagram Center*, accessed October 21, 2022; “[Guidelines for Describing STEM Images](#),” *WGBH.org (Boston radio)*, accessed October 21, 2022, <https://www.wgbh.org/foundation/ncam/guidelines/guidelines-for-describing-stem-images>; and Cesal, “Writing Alt-Text for Data Visualization,”
- [18] Edward R. Tufte, *The Visual Display of Quantitative Information* (Cheshire, CT: Graphics Press, 2001).
- [19] Cindy Xiong, Lisanne Van Veelden, and Steven Franconeri, “The Curse of Knowledge in Visual Data Communication,” *IEEE Transactions on Visualizations and Computer Graphics* 26, no. 10 (2019): 2051–3062.
- [20] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [21] See <https://www.R-project.org/>.
- [22] Elio Campitelli (eliocamp), “ggdatasaver: Automatically Save Data Associated with a ‘ggplot2’ Plot,” *Github repository*, last updated October 6, 2022, <https://github.com/eliocamp/ggdatasaver>; see also Hadley Wickham, “*ggplot2: Elegant Graphics for Data Analysis*, Second Edition” (Berlin: Springer Nature, 2016).
- [22] Wickard, “Is Accessibility Part of ‘Open’ Science?”
- [23] Jung et al., “Communicating Visualizations without Visuals.”
- [24] Lundgard and Satyanarayan, “Accessible Visualization via Natural Language Descriptions: A Four-Level Model of Semantic Content.”
- [25] Cynthia L. Bennett, and Os Keyes, “What Is the Point of Fairness? Disability, AI, and the Complexity of Justice,” *SIGAccess newsletter* 125, October 2019, <http://www.sigaccess.org/newsletter/2019-10/bennet.html>
- [26] See <http://diagramcenter.org/> and “[Guidelines for Describing STEM Images](#),” *WGBH.org (Boston radio)*, accessed October 21, 2022, <https://www.wgbh.org/foundation/ncam/guidelines/guidelines-for-describing-stem-images>.

Coding Accessible Data Visualisations



LÉONIE WATSON

We visualize data because it is often easier to understand in the form of charts and graphs, but what happens when someone who cannot see encounters a data visualization? The secret to making data visualizations accessible for blind people and people who use screen readers is creating an alternative representation of the data.

We can do so by making a few additions to the code used to create web pages, content, and web applications. This supplementary code is known as Accessible Rich Internet Applications, or ARIA—a set of roles and attributes that define ways to make the code more accessible to people who use screen readers.

Unlike most chapters in this *Do No Harm Guide*, this chapter will consist of several code snippets that demonstrate how ARIA code can be added to existing web page code to make it more accessible. My goal is to demonstrate that like any coding language or library, using ARIA requires understanding syntax and implementation and how to modify existing code as well as writing new code. Creating a more inclusive and accessible web should not be a tremendous burden, especially for readers already familiar with writing code.

Why Code Is Important

Imagine a door with a sign that says “kitchen.” The door is closed, and it has a metal plate running down one side. You may not realize it, but you already know a lot of information about this door. You recognize it’s a door because you’ve seen one before, you know its purpose is to let you move from one room into another, and you recognize that despite the door being closed, you can open it by pushing against it and end up in the kitchen, not some other room. In other words, the visual characteristics of the door tell you what it is, what it’s for, if it’s the door you want, and how to use it.

Let’s transfer this process to the web. This time, imagine a button with the word “Search” as its label. You recognize it is a button because you’ve seen buttons before, you know it will execute a search when you activate it, and you know you can click on it with your mouse, tap it with a finger, or use the Space or Enter keys to make it work.

But what happens if you can't see the button? In that case, you need something else to give you the information you need. This is where the code behind a data visualization is important for data accessibility. For the button, you'll need the browser and your screen reader (see also Chapters 4 and 6) to interact and present all that information in the code you use to build a web page and its content.

HyperText Markup Language, or HTML, is the programming language web developers use to structure web pages and their content, and that language has a lot of accessibility information built in. We use the words "role," "name," and "state" to describe these built-in features and to assign them to elements on the web. Most HTML elements have "roles" that describe their purpose: an `` element—which is used to embed an image on an HTML page—has the role of "graphic." Many HTML elements can be given a "name" (sometimes called an "accessible name"). A button on a webpage, for example, can be defined by text inside the `<button>` element. Some HTML elements also have "attributes" that describe their state, like the `checked` attribute that can be used to indicate when a radio button has been selected. If the `checked` attribute is present, the radio button will be selected, and if it is not present, the radio button remains unselected.

When a document created with HTML is loaded in the browser, the browser puts all the accessibility information into a structure known as an "accessibility tree." When someone using a screen reader goes to the HTML document in their browser, their screen reader asks the browser for the available accessibility information about what's on screen and uses that information to let the user know what they're dealing with.

Unfortunately, this does not work so well with what are called scalable vector graphics, or SVGs. When it comes to the web, SVG images are useful because the images will maintain their resolution no matter the size of the screen, from a phone to a tablet to a huge monitor.

Understanding SVG Images

Images on the computer can come in various file formats, like PNG and JPEG. Those two types are examples of raster files, which are based on pixels or small squares. When increasing the size of a picture saved as a JPEG file, for example, the squares get bigger and bigger. At some point, the image may become grainy and blurry because there aren't enough squares in the picture to give our eyes the illusion of smoothness. On the other hand, SVG images are vector files, which store images based on mathematical formulas and use points and lines on a grid. As you make an SVG image larger, the formula updates and the clarity of the image is preserved.

Although SVG elements have built-in accessibility information that is recognized by browsers, screen readers generally do not utilize it. But data visualizations commonly use SVGs, so we have to take matters into our own hands if we want our data to be accessible to people who cannot see them.

To make SVG images accessible, we use ARIA to add accessibility information ourselves. ARIA, which is a web standard created by the World Wide Web Consortium, can define roles and other accessibility properties to HTML or SVG elements on a web page.

Let's see how ARIA works for SVG images. We start by writing some SVG code to draw a button. This code creates a rectangle that will have the word "Play" as the title. This rectangle will be 75 pixels wide and 50 pixels tall, and it will have a purple fill and a green border the width of one pixel.

```
<svg version="1.1">
  <rect width="75" height="50" rx="20"
    ry="20" fill="#AC4FC6" stroke="#228b22"
    stroke-fill="1">
  <title>Play</title>
</rect>
</svg>
```

We can use the ARIA role attribute to explicitly give this rectangle the role of a button. The only change in this code snippet from the original is the addition of `<role="button">`.

```
<svg version="1.1">
  <rect role="button" width="75"
    height="50" rx="20" ry="20"
    fill="#AC4FC6" stroke="#228b22"
    stroke-fill="1">
  <title>Play</title>
</rect>
</svg>
```

Since a button is supposed to be interactive—clicked with a mouse or tapped with a finger—we need to make sure that keyboard users (including screen reader users) who may not be able to use a mouse can focus on it with the Tab key. To do this, we use the `<tabindex>` attribute (`<tabindex="0">`), which allows users to select the button by pressing the Tab key.

```
<svg version="1.1">
  <rect role="button" tabindex="0"
    width="75" height="50" rx="20" ry="20"
    fill="#AC4FC6" stroke="#228b22" stroke-
    fill="1">
  <title>Play</title>
</rect>
</svg>
```

If the purpose of the button is to turn something on or off or to start or stop something, we could also use the `<aria-pressed>` attribute to show its state. To show the user that the “Play” button has been pressed, we can add the `<aria-pressed="true">` code to the existing snippet. At first the attribute would have a value of “false” to indicate the button is not pressed, then it would update to “true” to indicate that the button has been pressed:

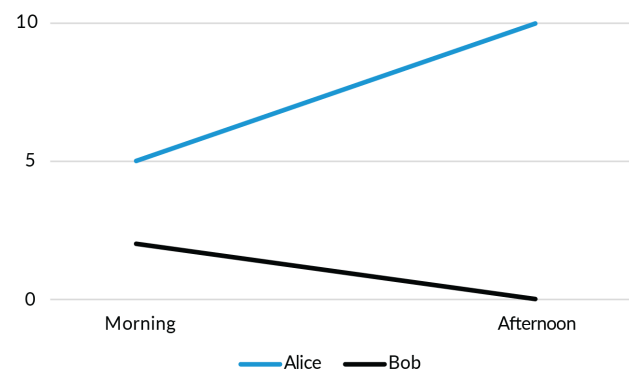
```
<svg version="1.1">
  <rect role="button" tabindex="0" aria-
    pressed="true" width="75" height="50"
    rx="20" ry="20" fill="#AC4FC6"
    stroke="#228b22" stroke-fill="1">
  <title>Play</title>
</rect>
</svg>
```

The changes we’ve made so far only apply to making the visual aspect of the button accessible to screen readers. To make this SVG button fully interactive, we have to use JavaScript to provide support for mouse and keyboard interaction so the button can function as expected.

Defining the Data Structure

Now let’s discuss how ARIA can be applied to data visualizations. When it comes to data structures, ARIA is not equipped for anything more sophisticated than bar charts, line graphs, or flowcharts. In other words, data that can be represented as simple tables or list structures.

Consider this simple line graph showing the average number of cups of tea had by people at different times of the day. The x-axis shows the different times of day, the y-axis shows a range of cups from 0 to 10 in increments of 5, and there are two lines that show how much tea Alice and Bob drank.



Visually, you might scan along the x-axis to find a time of day, then look up to find out how many cups of tea Alice drank. To put it another way, you might look along a row of data (i.e., the time of day), then up a column (i.e., the person) to get the data you want. Yes, you guessed it, a line graph can be represented as a table.

For the data in the line graph to make sense as a table—and readable for the screen reader as a result—we have to rearrange it so the rows represent Alice and Bob, the columns represent the different times of day, and the table cells contain the number of cups of tea that each person drank. Visually, the table will look like this:

	Morning	Afternoon
Alice	5	10
Bob	2	0

First, we use a `<g>` element in the HTML code to represent the entire table.

```
<g></g>
```

Then we need to nest three more `<g>` elements inside it to represent the rows, one for the column headers, one for Alice, and one for Bob:

```
<g>  
  <g></g>  
  <g></g>  
  <g></g>  
</g>
```

We need to explicitly define the cells inside the table, which we can do by adding more `<g>` elements. In this case, we will end up with three rows and three columns—thus, three cells in each row.

```
<g>  
  <g>  
    <g></g>  
    <g></g>  
    <g></g>  
  </g>  
  <g>  
    <g></g>  
    <g></g>  
    <g></g>  
  </g>  
</g>
```

We can then use the ARIA role attribute with different values to let browsers and screen readers know the purpose of the different `<g>` elements. Notice here that the ARIA role attributes define the overall table structure.

```
<g role="table">  
  <g role="row">  
    <g></g>  
    <g role="columnheader"></g>  
    <g role="columnheader"></g>  
  </g>  
  <g role="row">  
    <g role="rowheader"></g>  
    <g role="cell"></g>  
    <g role="cell"></g>  
  </g>  
  <g role="row">  
    <g role="rowheader"></g>  
    <g role="cell"></g>  
    <g role="cell"></g>  
  </g>  
</g>
```

Identifying Parts of the Visualisation

With the basic data structure in place with the appropriate roles, we can turn our attention to the next piece of accessibility information—an element’s name or accessible name. Here, we begin adding the data to the structure.

We’ll start with the information on the x-axis, the different times of day. We want this information to appear visually on the graph and to be accessible to screen reader users, so the `<text>` element is a good choice. We’ll add this to the first row of the table.

```
<g role="row">
  <g></g>
  <g role="columnheader">
    <text>Morning</text>
  </g>
  <g role="columnheader">
    <text>Afternoon</text>
  </g>
</g>
```

Think back to the door you imagined at the start of this chapter. Perhaps you pictured it as a wooden door with green paint and a sign in a bold font, or maybe you saw a neutral-colored door with a discrete sign. Whatever colour or sign type, it was still a door and that information let you make reasonable deductions about what it was for and how to use it. It’s the same with our line graph. To a blind person, it doesn’t matter how the data look; it only matters that they can identify what the data are and what the data’s purpose is so they can make some reasonable deductions about what to do with the data.

In other words, we’re putting in place an invisible structure that will be meaningful to someone who cannot see the line graph, then layering the visual representation of the data over the top.

The most likely SVG element for adding the lines to the graph is the `<path>` element. For the purposes of making our line graph accessible to screen reader users, the `<path>` elements are going to do double duty. They’re going to represent the visible lines on the graph that everyone will see, and they’re going to

be the content of the left-most cell in each row of our table structure for screen reader users.

Let’s start by adding a `<path>` element to the first cell in the next row of our table data structure:

```
<g role="row">
  <g role="rowheader"><path></path></g>
  <g role="cell"></g>
  <g role="cell"></g>
</g>
```

Next, we need to give the `<path>` element an accessible name so screen reader users know what the line represents. We can do so with the `<title>` element. (The highlighted text shows the new code we’ve added to the initial code block.)

```
<g role="row">
  <g role="rowheader">
    <path>
      <title>Alice</title>
    </path>
  </g>
  <g role="cell"></g>
  <g role="cell"></g>
</g>
```

Now, we need to make the screen reader think that the `<path>` element is actually an image, so we add `<path role="img">` to this bit of code.

```
<g role="row">
  <g role="rowheader">
    <path role="img">
      <title>Alice</title>
    </path>
  </g>
  <g role="cell"></g>
  <g role="cell"></g>
</g>
```

Within our code, we want to plot some points along each line so that screen reader users can know the specific data points. By doing so, we create the content for the cells in our table structure. Given that in our chart, the data is just plotted as a line, and as

such, each point has the same visual appearance, we need a common pattern to define the plot marker only once. Then, we can duplicate it using the `<use>` element. We use the same techniques as before to give each `<use>` element the “img” role and an accessible name using the `<title>` element, so the plotted points are accessible to screen reader users. Again, we are just adding a few short lines of code here, highlighted in yellow.

```
<g role="row">
  <g role="rowheader">
    <path role="img">
      <title>Alice</title>
    </path>
  </g>
  <g role="cell">
    <use role="img">
      <title>5 cups</title>
    </use>
  </g>
  <g role="cell">
    <use role="img">
      <title>10 cups</title>
    </use>
  </g>
</g>
```

These code examples are intentionally simplified for readability, but the `<path>` and `<use>` elements can be made to give the lines on the graph the visual representation you want. By adding the role and name information, we’ve created a table structure that a screen reader user can navigate by moving up and down through columns or left and right through rows, allowing them to obtain the same data as someone who is able to see the lines on the graph.

Hiding Unwanted Content

The invisible table structure we’ve put in place has all the information a screen reader user needs to make sense of the data, but there are other elements we need for the line graph to make sense visually. Although the y-axis isn’t needed in the table representation of the data, we do need it visually. To avoid screen reader users hearing the same

information twice, we can use a little bit more ARIA.

Suppose the y-axis will be represented using another `<g>` element. To hide it and all its content from screen reader users, we can use the `<aria-hidden>` attribute, like this:

```
<g aria-hidden="true"></g>
```

This same technique can be used to hide any other visual aspects of the line graph that would otherwise hinder screen reader users. Be careful with this technique, though. When you use `<aria-hidden>`, everything inside that element will be hidden from screen readers, so make sure you apply it thoughtfully to your code. You should never use `<aria-hidden>` to hide an element that is intended to be interactive.

If we put all the additions to our SVG code together, the basic code for our ARIA enhanced line graph looks like this:

```
<g role="table">
  <g role="row">
    <g role="columnheader">
      <text>Morning</text>
    </g>
    <g role="columnheader">
      <text>Afternoon</text>
    </g>
  </g>
  <g role="row">
    <g role="rowheader">
      <path role="img">
        <title>Alice</title>
      </path>
    </g>
    <g role="cell">
      <use role="img">
        <title>5 cups</title>
      </use>
    </g>
    <g role="cell">
      <use role="img">
        <title>10 cups</title>
      </use>
    </g>
  </g>
</g>
```

```

<g role="row">
  <g role="rowheader">
    <path role="img">
      <title>Bob</title>
    </path>
  </g>
  <g role="cell">
    <use role="img">
      <title>2 cups</title>
    </use>
  </g>
  <g role="cell">
    <use role="img">
      <title>0 cups</title>
    </use>
  </g>
</g>

```

Animating Information

You may want to make your line graph more interesting by adding a bit of animation. The graph could initially appear to be empty, then at the press of a button, animated lines could plot themselves along the x- and y-axes.

Visually this would most likely be done using Cascading Style Sheets (CSS), which are used to describe a webpage’s visual characteristics, such as colors, layouts, and fonts. To add animation through CSS and to make it accessible to screen reader users, we just need a little more ARIA. But first, we need to make sure the trigger for the animation is accessible to everyone. For that, we can reuse the SVG button from earlier.

```

<rect role="button" tabindex="0" aria
pressed="false" width="75" height="50" rx="20"
ry="20" fill="#AC4FC6" stroke="#228b22"
stroke-fill="1">
  <title>Animate graph</title>
</rect>

```

When the button has not yet been pressed, the `<aria-pressed>` attribute should have its value set to “false.”

For the animation itself, we need to create something known as a “live region”:

```

<g aria-live="assertive"></g>

```

A live region is a bit like a push notification for screen readers. As the content inside the live region is updated, screen readers will automatically announce it to the user. In this case, we want the announcement to happen right away, so we set the `aria-live` attribute to “assertive.”

When the button is activated and the animation starts, the `<aria-pressed>` attribute is set to “true” to show that the button has been pressed and a short description of the animation is read into the live region. The description could be something like the following:

```

<g aria-live="assertive">
Alice drinks five cups of tea in the morning,
and 10 cups in the afternoon. Bob drinks...
</g>

```

As the animation plays out on the screen, the screen reader automatically explains what the animation is doing. Notice that the `<aria-pressed>` attribute is set to “false” in the code above. Once the animation has stopped, the button shows that the animation can be triggered again by pressing the button.

We could also use data sonification, or representing the data as sound, instead of a live region. That entails creating an audio representation that matches the visual representation of the lines on the graph and triggering it to play at the same time as the visual animation. For simple patterns where the values plotted on a graph rise and fall, sonification can be a good technique. For example, the line on a graph may rise from a low value to a high value, which can be represented by a sound that increases in pitch or volume from low to high. For anything much more complicated though, sonification can become difficult to interpret without learning the patterns first.

Wrapping Up

We've begun to explore the importance of role, name, and state—collectively known as semantics—for making data visualizations accessible to people who cannot see them. By creating an underlying structure that makes the data meaningful to screen reader users and by using SVG images (possibly with a little CSS animation) to layer the visualization on top of it, we created a simple line graph that is accessible to screen reader users.

You can apply these same techniques to make other data visualizations accessible, but sadly there is a caveat: the technologies we have at our disposal, namely SVGs and ARIA, and the technologies used by consumers, namely browsers and screen readers, are limited in their capabilities. As mentioned, ARIA does not have the roles to support data structures more complex than tables or nested lists, and browser and screen reader support for SVGs (even accessibility-

enhanced SVGs) is remarkably inconsistent. For these reasons, you should remember two important things: First, accept the creative challenge and experiment with different ways to make your data visualizations accessible to screen reader users. And second, test what you produce with as many different browser and screen reader combinations as you can.

CHAPTER SIX

Creating Better Screen Reader Experiences



SARAH FOSSHEIM

Data practitioners have many tools to make data accessible, and data visualizations are one particularly powerful tool. Visualizations can summarize data into understandable points, highlight outliers, or shed light on trends. Interactive visualizations enable people to explore data differently than with a table and can be a great way of giving people access to both the high-level trends and the nuanced details.

As their name indicates, data visualizations typically use a lot of visual elements and properties to convey information. The color, opacity, width, height, coordinates, and shape of an element can all express meaning. And annotations, labels, and other chart elements can highlight certain properties of the data.

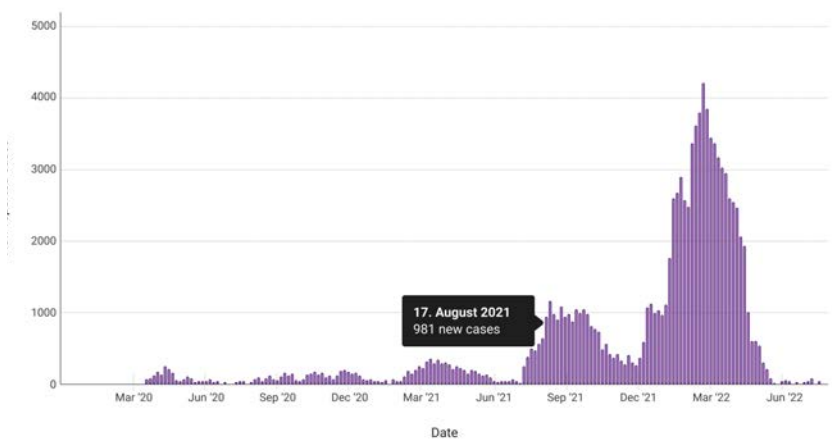
Take this bar chart for example. It shows two years of daily new COVID-19 cases: each bar represents a date, sorted chronologically; the bar height represents how many new cases were registered that day; the horizontal axis label tells us the date; and the vertical axis labels tell us the number of daily cases.

Number of new COVID-19 cases reported

Yesterday, **32 new cases** were reported. Last week's average was **47 new cases** a day. The highest amount of new cases in a day was 4167 on 03. March 2022.

View as

Bar chart Data table



Visually, we can spot the following relatively quickly:

- When we experienced spikes or outbreaks and approximately how long they lasted, which is shown by viewing where consecutive bars suddenly gain height.
- How the outbreaks throughout the pandemic compared with each other, which is shown by comparing the approximate average height of each of the spikes.
- How COVID-19 spread is progressing currently, which is shown by comparing the height of the bars on the right edge of the graph (the latest few days) with the bars before.
- The exact number of cases for any given date, which is shown by reading a tooltip that appears when hovering over any given bar.

With so much information primarily communicated through visual elements, how can we make this chart accessible to blind people? Many blind people use screen readers, a type of assistive technology that reads the page back to them (we'll discuss them more in the next section; see also Chapters 4 and 5), so we have to take a few extra steps when creating data visualizations to make sure the same information that's displayed visually can be read by screen readers.

In this essay, I provide guidance on what goes into creating data visualizations that are accessible to screen readers. I will do so by starting with an overview of what screen readers are and how they work. Afterward, I will demonstrate three different ways data visualizations can be made accessible for people using screen readers. Next, I argue why it is important to keep the end user in mind during the creation process. I then reflect upon how data structures influence which solutions we can use to make our charts accessible and how the chart structure plays into that.

How Screen Readers Work

First, let's take a closer look at screen readers and how they work. Screen readers are most commonly used by blind people and people with low vision, but

consuming text in audio format can also benefit people with cognitive impairments, neurological conditions, or learning disabilities.

On the web, screen readers use the content in a web page's framework (known as the Document Object Model) or page structure to read the elements on the screen back to the user. Practically, only elements containing text or specific properties are read to the user.

Users can move the screen reader's focus around the page through keyboard shortcuts on a desktop computer or touch gestures on mobile devices or computers with touchscreens. When the screen reader focuses on an element, its content gets announced. It is possible to jump between all elements on the page in the order they appear or to navigate between headings, sections, and interactive elements. The majority of screen reader users navigate web pages by headings first.^[1]

For this section I will focus on VoiceOver, Mac's built-in screen reader, as an example (I provide a list of others at the end of this chapter). Once VoiceOver is open, users can use the Ctrl + Option + U key combination to open the rotor menu (shown on the next page),^[2] where we can find a list with all extracted heading elements. Other screen readers have similar menus, which can be accessed through their own shortcut.


This rotor menu lets users browse the contents of the screen in a different way: it lists all the headings, landmarks, links and form elements on the page. For example, the following image shows VoiceOver's Headings menu. The number in front of the heading titles is the heading level and is used to give further context on the page structure.

By reading this list, people using screen readers can get a better understanding of the structure on the page, similar to how a table of contents in a book or paper might summarize a structure. It's possible to navigate this list using the arrow keys and select an item to navigate to. All of this information is based on the elements and properties the developers used when building the web page.

Headings

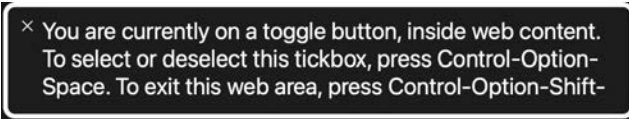
- 1: WCAG 2 Overview
- 2: Summary
- 2: Introduction
- 2: WCAG 2.0, 2.1, 2.2
- 2: Who WCAG is for
- 2: What is in WCAG 2
- 3: Supporting material and supplemental guidance
- 2: Translations
- 2: WCAG 2.0 is ISO/IEC 40500
- 2: Who develops WCAG
- 2: WCAG 3 and More Information
- 2: Help improve this page

Web pages can be built with both semantic and nonsemantic elements. Nonsemantic elements are those that don't tell the user anything about the content on the web page—these are the code pieces that define a section of the page or define groups of items on the page. Semantic elements, by contrast, define the content on the page—they are code pieces that define elements like tables, checkboxes, or submit buttons. When using semantic elements or equivalent aria roles,^[3] the screen reader can communicate extra information to the users. In the following example, the screen reader tells the user they are focused on a button, that the button is not pressed, what will happen if they press it, and which shortcut can be used to do so.



× Pause Music, selected, toggle button

When the user navigates to the pause button on a music player, VoiceOver will say, "Pause Music, selected, toggle button."



× You are currently on a toggle button, inside web content. To select or deselect this tick box, press Control-Option-Space. To exit this web area, press Control-Option-Shift.

When the user navigates to the toggle button on a web page, VoiceOver will say "You are currently on a toggle button, inside web content. To select or deselect this tick box, press Control-Option-Space. To exit this web area, press Control-Option-Shift."

The screen reader has access to all of the information that the developer includes in their code, turning those snippets into audible descriptions. To ensure webpages

are accessible for people who use screen readers, developers can include more specific information in their code.

List of Screen Readers

We used the VoiceOver screen reader tool as an example in this chapter. On computers running the Windows operating system, the default screen reader is called Narrator. It's also possible to use third-party software, such as NVDA or JAWS. According to WebAim's 2021 Screen Reader Survey,^[4] the most commonly used screen readers are JAWS (70 percent of respondents commonly use it, 53.7 percent use it as their primary screen reader); NVDA (58.8 percent of respondents commonly use it, 30.7 percent as their primary screen reader); VoiceOver (41.3 percent of respondents commonly use it, 6.5 percent as their primary screen reader); and Narrator (36.8 percent of respondents commonly use it, 0.5 percent as their primary screen reader).

How Data Can Become Accessible to Screen Readers

Data visualizations contain a lot of graphical information that is more difficult than simple text for a screen reader to assess. Sometimes even the visual properties of text elements (e.g., their positioning) in data visualizations carry meaning, further complicating matters. We can use a few common techniques to communicate the insights visualized in charts to screen readers. Each of those are navigated and read differently, so they have different implications for the user experience.

Text Alternatives

When images or icons are used in applications or on the web, developers commonly add something called alternative text to them by using the "alt" attribute (or as visible text next to the image, if adding an "alt" attribute isn't possible).^[5] If a graph isn't posted as a static image (e.g., a JPG or PNG file), it can be turned into one programmatically by using the "img" role.^[6]

The screen reader will read out this alternative text when focused on the image.

When writing alternative text, experts recommend you include in the description the chart type,

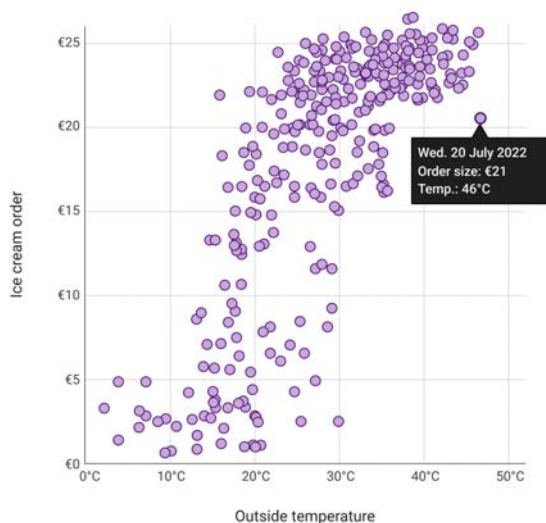
- the type of data visualized,
- the main conclusion or reason for showing the graph, and
- a link to the data source.^[7]

Chapter 4 goes more in depth on how to craft alternative text. In some situations, we can hand-craft the alternative text, write it once, and publish it with the image. This approach does not work as well with dynamic or frequently updated data because we would have to automate the construction of the alternative text as well.

Further, not every graph can be fully summarized or replaced with a few sentences of text. Let's take this scatterplot visualizing an ice cream shop's average order price (in Euros, visualized on the vertical axis) compared with the outside temperature (in degrees Celsius, on the horizontal axis).

How the outside temperature influences sales

More orders are placed when the temperature is higher, and the most expensive orders get placed when the temperature climbs over 25°C.



Though we may be able to write or generate a high-level summary of this graph's insights, such alt text still may not be enough to provide the same insights to

both people who are using screen readers and sighted people, who can see the individual points on the chart and interact with them to reveal more data. (Notice the alt text I've provided below the image describes the basic plot, the ranges on both axes, and the overarching point. Notice also that I have not assigned alt text to the actual image in this document because the alt text is included in the image caption. If both existed, a person reading this report with a screen reader would hear the description twice.)

As a result, sighted users can use all the information available to draw their own conclusions about what the data mean. If we want to give people using screen readers the opportunity to dive into the data in a similar way, we need to give them a way of navigating the dataset beyond images and text alternatives.

Table Alternatives

Although text descriptions are useful for summarizing data, table alternatives shine when it comes to giving access to detailed information. Tables can have infinite rows and columns, which can be structured through headings, subheadings, and pagination. Additionally, we can add sorting, filtering, or search functionality to let users explore the data in their own way.

When tables are built using the appropriate elements^[8] (e.g., using HTML elements such as `<table>` on the web), screen reader users get access to more context around the table and special keyboard commands to navigate it. Tables can be navigated horizontally and vertically, and the connected column and row headings are read out to the user.^[9] Because native tables come with those established interaction patterns, users don't have to relearn how to navigate tables across different products.

The familiarity of tables, along with their ability to contain a lot of detailed information in a structured way, offers a reliable and safe way of ensuring the data are always available. With data visualizations, I advise always having a table alternative, even when using other strategies such as alternative text. Where

possible, table alternatives should be displayed alongside or close to the graph. In this next example, the data visualization has a table and chart view available, which the user can toggle between using tab navigation.

How the outside temperature influences sales

More orders are placed when the temperature is higher, and the most expensive orders get placed when the temperature climbs over 25°C.

View as

Scatter plot Data table

All orders, sorted by temperature.

Order nr.	Date	Temperature	Price
#72543	20. July 2022	46°C	€21
#72544	20. July 2022	45°C	€26
#72614	21. July 2022	45°C	€23
#72183	19. July 2022	45°C	€24
#72526	20. July 2022	44°C	€25
#72627	21. July 2022	44°C	€20
#72173	19. July 2022	44°C	€26
#72274	19. July 2022	44°C	€24
#72602	20. July 2022	44°C	€23

Jump to page 1 2 ... 6

Despite those strengths, tables are not a perfect solution. They are not great for summarizing data or for communicating trends and patterns. Imagine more than two years of daily COVID-19 case numbers in tabular format. All the information about the peaks and duration of the outbreaks exists, just like in the bar chart at the start of this chapter, but the user has to interact much more to discover them.

Interactive Scalable Vector Graphics

As demonstrated in Chapter 5, when developing graphs using Scalable Vector Graphics, we can add more information for screen readers through ARIA labels.

Adding screen reader interaction to SVGs leaves us with a lot of control and choice over what information we expose and how we expose it. In Chapter 5, Léonie Watson shows us how a line chart can be turned into a table structure. Similar methods can be used to make screen readers announce elements such as lists,

buttons, images, or other native components. We can also group elements and custom navigation to provide multiple levels of detail.

Recent research from Jonathan Zong and coauthors at the MIT Visualization Group demonstrates how a user can explore a scatterplot across three different design dimensions.^[10]

- **Structural navigation**, where the user navigates based on the accessibility structure and jumps between nested elements.
- **Spatial navigation**, where the user navigates based on the chart's visual coordinate system, using the down arrow key to jump to the element that is visually below their current location, for example.
- **Targeted navigation**, where the user navigates based on landmarks and regions. The user could open the rotor menu, which provides the full structure of the page, and jump directly to the relevant section.

Understanding User Needs

Now that we have a better understanding of how screen readers work and what common accessibility strategies are, we should take a step back before we go further and reflect on why we want to create data visualizations.

A fitness tracking app, for example, may use a bar chart to show how many steps someone took today compared with this month's average. Users might like this chart because they can see if they're on track to meet their fitness goals. By comparison, a bar chart in a scientific report that shows daily new cases of COVID-19 over the span of the pandemic may require more functionality so users can look up values or trends for specific dates and periods.

In the first graph, the main point—how many steps the user has taken today and whether they have taken more or less than their average—can be summarized in one sentence. The second graph, on the other hand, provides a lot of data to an audience that wants to explore the details. For this second graph, giving

people who use a screen reader just a one-sentence summary would provide a less robust view of the data and an insufficient experience compared with the experience given to sighted users.

Who Are We Building For?

If we want to create data visualizations with a good user experience, we need to understand who our users are and include them in our processes. In other words, we need to understand who we are building our visualizations for rather than creating the graph and hoping it will match our users' needs and expectations.

We should ask ourselves the following:

- Who are our users?
- What are their accessibility needs?
- What are their technology and data literacy skills?
- Under which circumstances are they visiting the graph?
- How do they expect to interact with the graph?
- What information are they after?
- What actions will they need to take based on the data?

To find some of this information, we can conduct user interviews or user testing. Other tools, such as accessibility personas,^[11] can help us get a better overview of different user needs.

Accessible Data Structures

All graphs, regardless of how they work or who their users are, have one thing in common: they visualize data. Because screen readers don't have access to the visual properties of a chart, it is useful to reflect on what type of data we're working with before building the structure. Being strategic with these considerations will improve the final product and make your workflow better and more efficient.

Is the Dataset Static or Dynamic?

Whether a visualization is static or dynamic—that is, whether the data will update within the visualization—and how frequently it updates, can limit or change our solutions. If the visualization is static or only updates occasionally and with manual intervention, we have more opportunity for hand-crafted captions or custom interactions. But the less control we have over the data, the more generic our solution may have to be or the more edge cases we will have to consider.

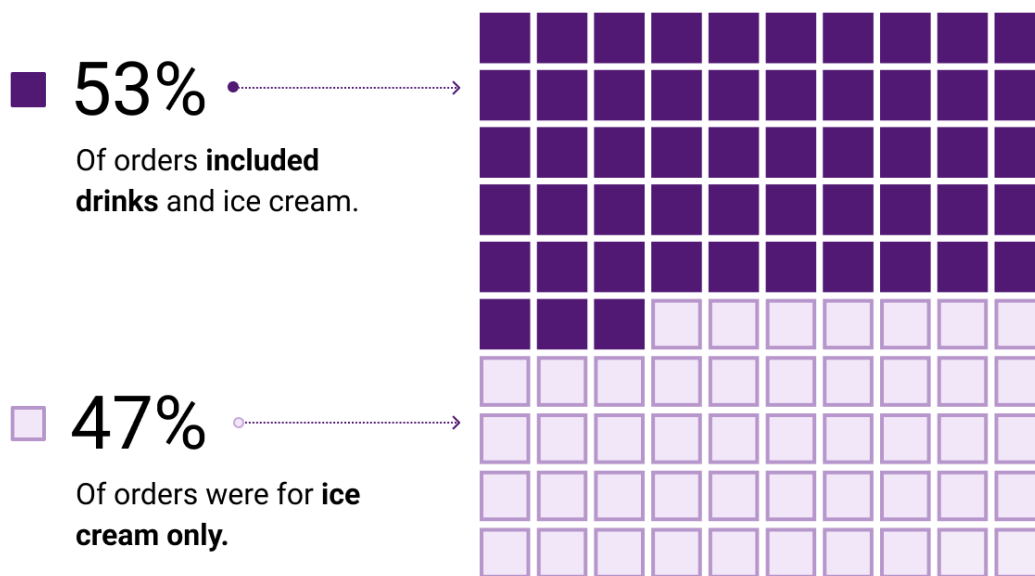
If the chart updates “live,” meaning without user interaction, screen reader users will need to be notified. On the web, we can [add the `aria-live` attribute on the region that will be updated to prompt the screen reader to announce the updates, either by interrupting the user \(for time-sensitive actions\) or by waiting until they are idle \(in most cases\).](#)^[12]

How Large Is the Dataset?

A bar chart showing the number of visitors a website had each day over the past week only has seven data points, whereas a bar chart showing the daily visitors over the entire lifespan of the website may contain thousands. When a chart has only a handful of data points, it is easier to remember specific data or spot trends. But the larger the dataset, the harder interpreting the data becomes. When exploring a table or interactive chart with many data points, the user will need to tab through a lot of information, which means we may need to provide them with additional labels or controls (e.g., pagination or sort and filter functionality).

What Type of Data Are Visualized?

The type of data we are visualizing often influences how we display it. Let's take the following chart visualizing the share of ice cream orders that also included a drink. For accessibility, it makes more sense to focus on the underlying data, the percentages, rather than the visuals of the chart. We don't really want screen reader users to have to tab through the values of a hundred cells. In this case, the author chose to add the percentages in text format next to the graph.



For the example visualizing COVID-19 cases over time from earlier in the chapter, the data were shown in a bar chart. Each bar represented a date, and the bars were sorted chronologically, from the oldest date on the left to the newest on the right. We read from left to right visually, so the screen reader should have access to the data in a similar way to give screen reader users a good experience.

There are, of course, an unlimited number of graph types, and graphs do not necessarily map to data types one-to-one. In other words, one type of graph may be used to show several kinds of data.^[13]

Are Any Trends or Patterns Identifiable?

Graphs can be powerful at highlighting trends, patterns, and outliers. Think of the graph with the daily new COVID-19 cases we explored earlier. It didn't just give us access to all the numbers like a table would, it also showed us how many large spikes we experienced and when there were prolonged surges.

A user should be able to draw the same conclusions when browsing the chart with a screen reader. We can solve this problem in different ways, including the following:

- Describing the general trends
- Grouping data points and offering custom navigation

- Creating a different graph focusing only on the trends
- Adding sorting and filtering functionality to table alternatives
- Mapping different values to different sounds (a process called sonification)

The best strategy will depend on the goals for the visualization, the user's needs, and the existing data properties.

Accessible Chart Structures

Besides data points, data visualizations contain a lot of perceivable information, including headings, legends, axis labels, and annotations. We should thoughtfully design these other supporting chart elements to provide contextual information for screen readers as well.

Headings and Captions

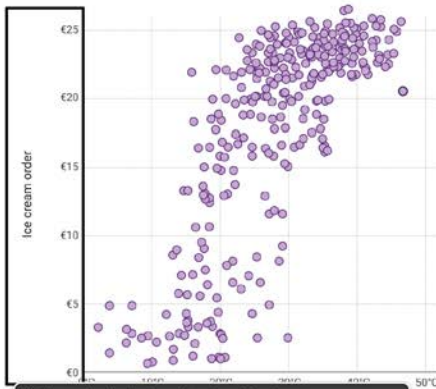
Proper use of heading elements can make content easier to find, as demonstrated in Chapter 7. This organization can also help users understand the page structure, which provides a way of grouping the data.

Titles should set a clear expectation about the purpose of the graph. Additional descriptions can explain how to read the chart, interact with it, or interpret the data.

How the outside temperature influences sales

More orders are placed when the temperature is higher, and the most expensive orders get placed when the temperature climbs over 25°

View as

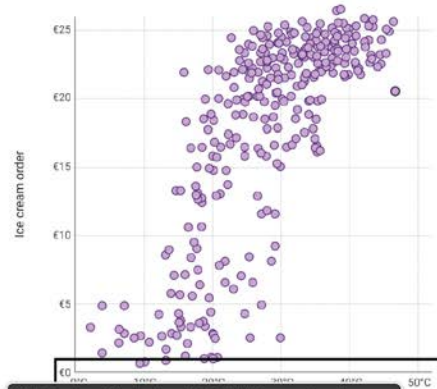
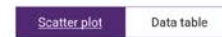


× The vertical axis shows the price of the ice cream orders. The orders range between €1 and €25.

How the outside temperature influences sales

More orders are placed when the temperature is higher, and the most expensive orders get placed when the temperature climbs over 25°

View as



× The horizontal axis shows the Outside temperature at the time of the order. The recorded temperatures range between 0°C and 50°C.

We can also summarize the visualization in text as part of the heading or description of a graph, which everyone can read and understand.

Explanatory Chart Elements

Additional explanatory chart elements can be instrumental for understanding the data. The ticks on the vertical axis of a line chart, for example, give us an idea of the minimum and maximum values in the dataset. The legend of a graph tells us which categories of data are represented.

Without extra context, those elements might not make much sense. We can choose to hide elements that only serve a visual purpose, as demonstrated in Chapter 5, or we can extend their labeling and functionality to provide more context to screen reader users. The label of either axis can be extended to include information about the range of numbers visualized on it, which we can see demonstrated in the mockup below. The scatterplot comparing the outside temperature to ice cream sales has a summarizing label added to each axis but visually hidden; this gets read instead of the ticks.

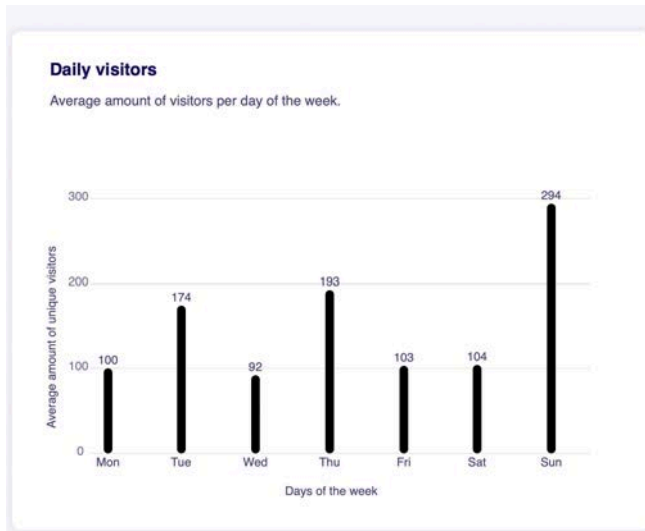
Order of Elements

The coordinates and styling determine how data points are sorted visually, while screen readers base themselves on the order of the elements in the accessibility tree.^[14]

For this chart, the screen reader experience differs greatly from a visual experience depending on the underlying organization.

Following this structure, a user can tab through all of the elements in an order that matches how we expect to browse the chart visually:

1. Title
2. Description
3. All the bars
 - Sorted by date
 - First the date is announced, then the value



Simplified code:

```
<h3>Daily visitors</h3>
<p>Average amount of visitors per day of the week
</p>
<g>
  <text x="0">Monday: 100</text>
  <text x="20">Tuesday: 174</text>
  <text x="40">Wednesday: 92</text>
  <text x="60">Thursday: 193</text>
  <text x="80">Friday: 103</text>
  <text x="100">Saturday: 104</text>
  <text x="120">Sunday: 294</text>
</g>
```

A screen reader would read this chart like this:

Transcript of graph 1: "List, seven items. Monday: 100 visitors, 1 of 7. Tuesday: 174 visitors, 2 of 7. Wednesday: 92 visitors, 3 of 7. Thursday: 193 visitors, 4 of 7. Friday: 103 visitors, 5 of 7. Saturday: 104 visitors, 6 of 7. Sunday: 294 visitors, 7 of 7. End of list." [15]

Now let's take a different chart that visually looks the same but has the following structure (next page):

Transcript of graph 2: "List, 7 items. Friday: 103 visitors, 1 of 7. Monday: 100 visitors, 2 of 7. Wednesday: 92 visitors, 3 of 7. Tuesday: 174 visitors, 4 of 7. Sunday: 294 visitors, 5 of 7. Thursday: 193 visitors, 6 of 7. Saturday: 104 visitors, 7 of 7. End of list." [16]

In this second chart, the dates are not read in chronological order. Here, the screen reader announces Friday first, then Monday, then Wednesday, and so on. The reason this happens is because while the styling determines the order elements are placed visually, it's the Scalable Vector Graphics structure that generally determines the order for screen readers.

Therefore, you must be mindful of sorting your datasets correctly before rendering them. Data should be sorted in a way that makes sense and follows the visual representation. Similar to how the order of the data points matters, think of how the other chart

elements fit into this structure. For example, it might make more sense to make the screen reader announce the title and range of each axis as a summary before going through the actual data.

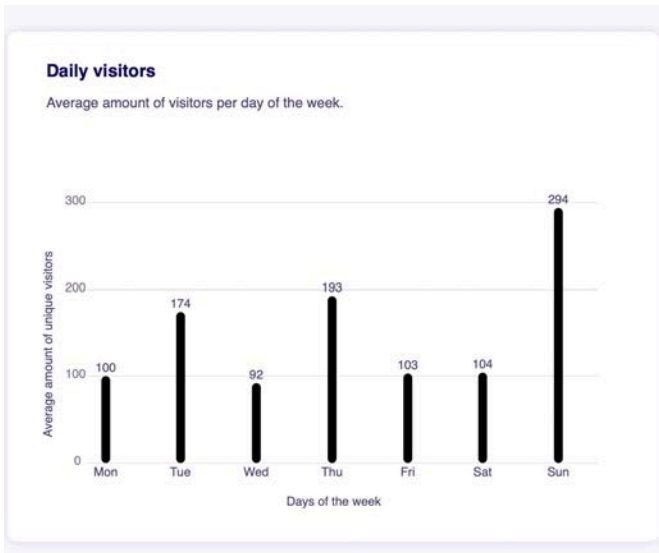
Interaction

If a data visualization has pointer action available, the same actions should be accessible for users who only use a keyboard. A tooltip that exposes more information when hovered over should expose that same information when it is focused on through keyboard interaction.

Screen reader users should also clearly be shown which elements are interactive and what state they are in. As demonstrated in Chapter 5, we can use the button role, combined with the "aria-expanded" or "aria-pressed" element, to do this. [17]

Accessibility-Driven Visualizations

How we design the screen reader experience depends on what data are visualized, who the visualizations are for, which context they exist in, and under which circumstances they are used. Designing this experience fits right into other usability or information architecture processes we may already be following, because assistance technology bases itself on the structure of the visualization and the elements it contains.



Simplified code:

```

<h3>Daily visitors</h3>
<p>Average amount of visitors per day of the week
</p>
<g>
  <text x="80">Friday: 103</text>
  <text x="0">Monday: 100</text>
  <text x="40">Wednesday: 92</text>
  <text x="20">Tuesday: 174</text>
  <text x="120">Sunday: 294</text>
  <text x="60">Thursday: 193</text>
  <text x="100">Saturday: 104</text>
</g>

```

Although a common workflow may be to design a visualization first and later think of how or if that visualization can be made accessible to screen readers, we should instead start thinking about accessibility as early as possible so we can make the right decisions from the start. Integrating accessibility into existing processes or creating new ones that account for accessibility can help us prioritize accessibility in data visualization. This includes the testing and quality control of our charts. User tests with blind people are important to ensure we're providing a good and meaningful experience, and audits or internal reviews can help us discover and mitigate issues before they reach the user.

Although making data visualizations accessible to screen readers may seem like extra work, data practitioners can actually create a good and meaningful experience for all audiences if they have the foresight to consider accessibility from the beginning.

Chapter Six Notes

- [1] "Screen Reader User Survey #9 Results," WebAIM, last updated June 30, 2021, <https://webaim.org/projects/screenreadersurvey9/>.
- [2] "ARIA - Accessibility," MDN Web Docs, last modified October 2, 2022, <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA>.
- [3] "Semantics," MDN Web Docs, last modified September 20, 2022, <https://developer.mozilla.org/en-US/docs/Glossary/Semantics>; "WAI-ARIA Roles," MDN Web Docs, last modified September 16, 2022, <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles>.
- [4] See the Screen Readers Commonly Used section of "Screen Reader User Survey #9 Results," WebAIM, last updated June 30, 2021, <https://webaim.org/projects/screenreadersurvey9/#used>.
- [5] "Alternative Text," WebAIM, last updated October 19, 2021, <https://webaim.org/techniques/alttext/>; <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img#attr-alt>
- [6] ", the Image Embed Element," MDN Web Docs, last modified October 10, 2022, https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles/img_role.
- [7] Amy Cesal, "Writing Alt Text for Data Visualization," Nightingale, the Journal of the Data Visualization Society, July 23, 2020, <https://medium.com/nightingale/writing-alt-text-for-data-visualization-2a218ef43f81>.
- [8] "Creating Accessible Tables," WebAIM, last updated September 18, 2017, <https://webaim.org/techniques/tables/data>.
- [9] Léonie Watson "How Screen Readers Navigate Data Tables," Tink.uk, September 28, 2020, <https://tink.uk/how-screen-readers-navigate-data-tables/>.
- [10] Jonathan Zong, Crystal Lee, Alan Lundgard, JiWoong Jang, Daniel Hajas, and Arvind Satyanarayan, "Rich Screen Reader Experiences for Accessible Data Visualization," Computer Graphics Forum (Proc. EuroVis) 41 (2022): <http://vis.csail.mit.edu/pubs/rich-screen-reader-vis-experiences/>.
- [11] "Accessibility Personas," AlphaGov, accessed November 28, 2022, <https://alphagov.github.io/accessibility-personas/>.
- [12] "ARIA Live Regions," Mozilla Developer Network, last modified October 26, 2022, https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/ARIA_Live_Regions.
- [13] See, for example, Jonathan Schwabish, "Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks" (New York: Columbia University Press, 2021).
- [14] "Accessibility Tree," MDN Web Docs, last modified September 20, 2022, https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree.
- [15] Sarah Fossheim, "Daily Visitors, Example #1," CodePen, accessed October 19, 2022, <https://codepen.io/fossheim/pen/LYQKYxZ>
- [16] "Sarah Fossheim, "Daily Visitors, Example #2," CodePen, accessed October 19, 2022, <https://codepen.io/fossheim/full/dydByJv>
- [17] "Aria-expanded," MDN Web Docs, last modified September 13, 2022, <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes/aria-expanded>; "Aria-pressed," MDN Web Docs, last modified September 25, 2022, <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes/aria-pressed>.



PART FOUR

Accessibility Testing
and Remediation

Practical Accessibility Testing for Data Visualizations



LARENE LE GASSICK

My father has [retinitis pigmentosa](#), a genetic disorder that gradually cost him his sight as I was growing up. As a result, I was introduced to software screen magnifiers, high-contrast themes, and screen readers at an early age. He listened to audiobooks with a [DAISY player](#)—a special hardware audio player invented for people with low vision or blindness—and used a white cane to navigate in public.

If my father didn't have his disabilities, I would have not known very much about assistive technologies. Many people who do not work in spaces directly related to disability and accessibility have never heard of a screen reader, a braille display, or a switch control, all of which are known as “assistive devices” or “assistive technologies.”

Unfortunately, most of the web is built with sighted or mouse-first users in mind. But assistive technologies can help people navigate their computer and smart device.

A screen reader uses a synthetic voice to describe what is happening on the screen. A refreshing braille display uses a live-updating braille tactile display instead of a synthetic voice. Screen-magnifying software enlarges the screen from 150 percent to over 500 percent of the original display size to aid those with low vision. Text readers read aloud content while visually highlighting each word for those with cognitive disabilities such as dyslexia, and alternative input devices, such as voice control software, head pointers, eye trackers, and switch controls allow people with some physical disabilities to navigate a website using only their keyboard. Most screen reader users are also keyboard-only users. Operating systems also come with personalization options, which allow a user to adjust text size, reduce animation, and turn on a high-contrast theme.

In this essay, I describe an easy-to-follow testing process so people who have never tested their data visualizations for accessibility can ensure their products work just as well for people who use any of the above assistive technologies. Many existing articles about accessibility testing simply provide a checklist of tests and a list of tools, but they do not go into details about the process. I have also created a list of additional tools you can use in your own work, available later in this volume.

Digital Accessibility Standards

Creators of digital apps, content, and services should be aware of how people with disabilities use these technologies in their daily life. How would you build a wheelchair ramp if you didn't know how a wheelchair worked or that wheelchairs even existed? More importantly, what if you didn't know who uses a wheelchair and why?^[1]

With so many different types of assistive technology, web browsers, and device operating systems, how can we support all users? Luckily, there is some standardization.

Just like the physical construction industry has building standards and guidelines for accessibility, such as the angle and width for a wheelchair ramp,^[2] the web also has standards and guidelines for digital accessibility. The [Web Content Accessibility Guidelines \(WCAG\) working group](#),^[3] which is part of the Web Accessibility Initiative, an international initiative to develop standards to create a more accessible internet, publishes guidelines and “success criteria” that can be applied to a range of websites and applications and serve as a foundation for many countries' compliance and antidiscrimination laws. Unfortunately, the technical nature of the guidelines has made them quite inaccessible to someone new to web accessibility. A deep understanding of WCAG is not needed to create accessible content or conduct accessibility testing and the WCAG 2.1 At a Glance web page succinctly summarizes the most important aspects and requirements to creating more accessible online content.^[4]

In 2021, Frank Elavsky created a set of guidelines called the Chartability project to allow anyone in the data field to conduct accessibility testing of data experiences in any context, not just the web.^[5] Elavsky worked closely with experts in the cross-over of accessibility, disability, and data visualization communities, and he has tested the usability of the Chartability workbook with many data visualization practitioners.^[6]

The Testing Process

We use accessibility testing to identify and remediate inaccessible web practices.^[7]

Two types of accessibility testing exist: personal testing, which all data practitioners should learn to ensure people with disabilities aren't excluded from their work, and professional “accessibility auditing,” which is conducted by someone trained to report a list of issues for the client to fix—a process known as “remediation.” In this essay, I will cover how data practitioners can begin to do personal accessibility testing, but companies should schedule regular accessibility audits for their publications, websites, or applications. Audits, at a bare minimum, test for WCAG compliance, and in the best cases, will check for usability issues and recommend solutions based on inclusive design principles, user research, and testing with assistive technology users and people with disabilities.

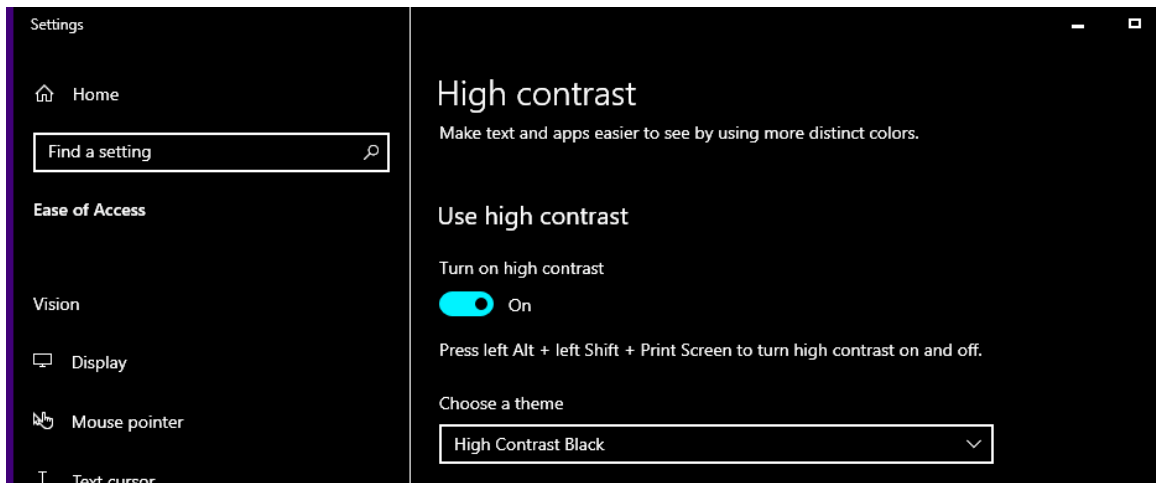
The testing process I discuss here is my personal process, but every accessibility tester's techniques and tools are tailored to their needs. Some methods are shared and new and improved tools are released regularly, but I have found these basic tasks and tests to be sufficient for general purposes.

I break up my testing procedure into the following four categories:

- Content
- Visual
- Nonvisual
- Interaction

For purposes of this essay, I assume that the publication, website, or application *surrounding* the data visualization—whether a chart, graph, diagram, or other piece of content—is accessible enough for the chart to be reached by the reader or user.

Before we begin testing, we need to have the right tools at our disposal. Here I list common accessibility issues and the tools or settings that I currently use to test charts and data visualizations, but a longer list



with additional resources and reference materials is available at the end of this volume. If you are new to accessibility testing, start with these tools, but don't be afraid to try others based on your personal preference and workflow. For testing with screen readers in particular, I recommend you watch the introductory videos linked in endnote 1.

Color Contrast

For low-vision users, a data visualization must have sufficient contrast between text and background. Contrast measures the difference in luminance between two colors. The contrast ratio between black (hex code #000000) and white (#FFFFFF) is 21:1. The WCAG 2.1 AA standard defines an acceptable contrast ratio as at least 3:1 for large or bold text and icons and 4.5:1 for general reading text.

Dozens of contrast checking tools are available. I use the TPGi Colour Contrast Analyser,^[6] which is a downloadable tool that enables you to test the contrast of your color palette against WCAG guidelines.

A user's operating system might also include settings, such as high-contrast mode, that can test the color contrast of a data visualization. In the Windows 10 operating system, you can activate high-contrast mode in Settings > High Contrast > Turn On High Contrast. The most commonly used high-contrast theme is white text on a black background with bright yellow accents.

In the supplemental list of accessibility tools at the end of this volume, I include a guide to changing color contrast in Windows 11.^[9]

Mac operating systems don't have a "high-contrast mode" like Windows, but the Dark Background and Light Text Firefox add-on for the Firefox browser offers a comparable simulator.^[10]

Color Vision Deficiency

The data visualization field often focuses on color vision deficiency (CVD), commonly known as "color blindness." But people with vision impairments can struggle with different kinds of color issues, not just distinguishing between, say, red and green (a condition known as deuteranopia).

Many tools can test for different kinds of CVD, but Color Oracle is my favorite.^[11] Color Oracle is a free simulator for Windows, Mac, and Linux that takes the guesswork out of designing for color blindness by showing you in real time what people with common CVDs see.

Reduced Motion

People with vestibular disorders or who are easily nauseated by animation and movement often turn on their system's "reduced motion" or "animation off" setting. Web developers can use "@media screen" and "(prefers-reduced-motion: reduce)" in their CSS animations to provide a nonanimated alternative for users who are affected by strong animation. More

generally, when creating animation or movement, it is good practice to turn off autoplay and give the user control to start or stop an animation.

For testing, you can adjust these settings on Windows and Mac operating systems:

- Windows: Settings > Ease of Access > Display > Simplify and Personalize Windows. Set Show animation in Windows to off.
- Macs: System Preferences > Accessibility > Display and check the box for Reduce Motion.

Screen Reader

Using screen readers to test your content will allow you to catch many critical accessibility issues in your data visualizations and your content more generally. The most common screen readers are VoiceOver for Mac and NVDA, JAWS, or Narrator for Windows. I use VoiceOver and NVDA because they are free and most familiar to me. I work on Windows and Mac computers, but you only need to test with one. I recommend watching the introductory videos linked in the list below and having on hand a quick reference guide of screen reader keyboard commands (I usually have mine printed out). I don't use a screen reader every day, so I often forget the keyboard commands between each testing session.

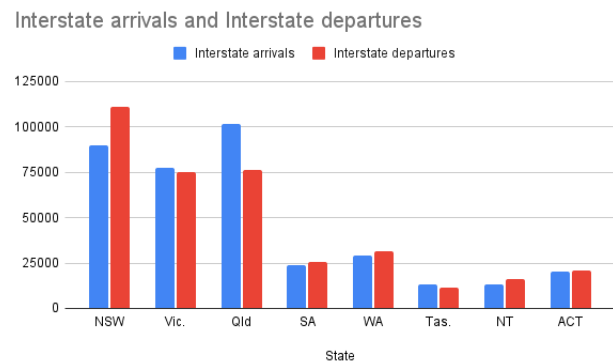
- Windows: [NVDA](#) is one of the most popular screen readers for Windows users. It works with all modern web browsers, email programs, music players, and Microsoft Office programs. NVDA does not offer any introductory tutorial, so I recommend you watch the following video:
 - [Screen Reader Basics: NVDA tutorial \(9 minutes\)](#)^[12]
 - [Reference for NVDA Keyboard Shortcuts on Windows](#)^[13]
- Mac: VoiceOver comes installed with MacOS. When you first open VoiceOver, you will be given the option to complete an interactive learning tutorial, which I recommend you take the time to complete; you should also watch the following introductory video:

- [Screen Reader Basics: VoiceOver tutorial \(12 minutes\)](#)^[14]
- [Reference for VoiceOver Keyboard Shortcuts on a Mac](#)^[15]

Part 1: Testing an Image of a Chart

A great starting point to learn accessibility testing is with a chart in an image format like JPG or PNG (rather than one that is natively interactive on a website). If a chart is an image, it is usually complementary to the surrounding text and has a single clear takeaway.

To demonstrate, let's begin with this default chart image exported from Google Sheets. It's a column chart titled "Interstate Arrivals and Interstate Departures," with two data series, interstate arrivals and interstate departures, denoted by blue and red columns, respectively. The x-axis is the categorical Australian State (for readers who doesn't know Australia's geography, the country has six states and two territories), and the y-axis is the number of people (not labeled).



Despite this chart being a default chart for one of the most popular charting tools in the world, it has many accessibility and usability issues.

As I mentioned, accessibility tests are split into four categories: content, visual, nonvisual, and interaction.^[16]

Content Testing

By making the content in the graph accessible, readers are better positioned to understand the graph and your argument. So the checklist below applies to all

good data visualizations, not just those for people with disabilities.

Content Testing Checklist

- Chart contains a descriptive title.
- Chart contains a suitable summary either in the chart or in the surrounding text.
- Chart uses plain language.
- Chart axes are labeled clearly.
- Chart series are labeled clearly.
- The numerical data is formatted clearly.
- If the chart can be used for analysis or to form one's own insights and is not adequately described by the chart summary, then a raw data table or a CSV download of the raw data is provided as an alternative.

Content Testing Process

We start with the content review process because it could necessitate quite drastic changes in the design of the visualization. Accessible content is essential for people with cognitive or intellectual disabilities, people who have low data literacy, or in this case, people who might not be familiar with Australian geography. In this example, the title was not descriptive, and the summary and y-axis label were missing. Also, the number formatting did not contain a comma, making it hard for people to read. A good rule of thumb for accessible text is to aim for a ninth-grade reading level; I recommend the Hemingway Editor for assessing the reading level of your writing.^[17]

Let's look at what works and what doesn't:

■ Passes

- **Chart uses plain language.** The chart doesn't use a lot of jargon or technical verbiage.

■ Failures

- **Chart contains a descriptive title.** The title does not mention important details such as the country and year.

- **Chart contains a suitable summary, in the chart, or in surrounding text.** No summary is present.
- **Chart axes are labeled clearly.** The y-axis is not labeled.
- **Chart series are labeled clearly.** The series labels contain too much text, and a descriptive title makes them redundant.

■ Extra consideration

- **If the chart can be used for analysis and is not adequately described by a chart summary, a raw data table or a CSV download of the raw data should be provided as an alternative.** This consideration would depend on the larger context in which the chart is used.

No real science exists on how to fix these issues, so it takes practice and testing with users. If you're not sure what to write, consider these leading questions:

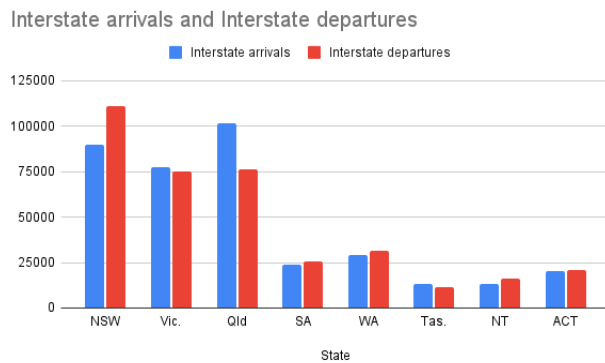
- What is the main takeaway for this chart? Describe what you want people to take away from this chart in a concise, active sentence.^[18] This description becomes the chart title.
- How would I describe this chart for a person who cannot see it? Think about what is interesting about this chart and what about this chart informs the context in which a reader will see it. That explanation becomes the chart summary.

For this chart, I'd also provide the raw data, either as an HTML table or as a downloadable CSV file (Note that CSV is the most accessible format for table data—not everyone has Microsoft Excel).

Content Testing Results

By working through this checklist, we have made several changes to the original chart. The chart title now reads "Interstate Migration in Australia, Year Ending June 2020" and we have a chart summary just below: "Queensland had the highest net gain from interstate migration of 25,300 people." We've also

Before content testing



added a y-axis label— “Persons (thousands)”—and simplified the legend to “Arrivals” and “Departures.” But we’ve also introduced a new accessibility issue: the chart summary uses the default text color of light gray, which will fail the contrast tests in the next step.

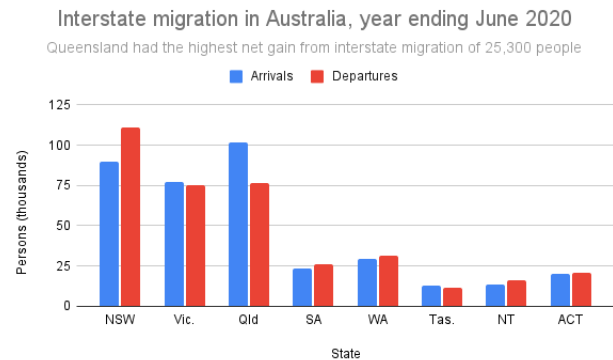
Visual Testing

Visual testing focuses on making the experience inclusive for users with low vision or CVD. These tests can also ensure that text is more readable for someone with dyslexia and that animations do not cause nausea or epileptic seizures.

Visual Testing Checklist

- All text is at least 12 pixels (or 9 point) in size, ideally 16 pixels (or 12 point).
- Small text and its background colors meets a minimum contrast ratio of 4.5:1. “Small text” is defined as text smaller than 24 pixels (or 18 point) or as bold text smaller than 18.5 pixels (or 14 point).
- The contrast ratio of shapes, icons, and large text relative to their background color meets a minimum ratio of 3:1.
- Color alone does not denote meaning—that is, contrast, markers, or patterns are used to differentiate data and are labeled appropriately (i.e., with a legend or nearby text).
- Text font is sans serif (that is, the letterforms in the font do not have decorative marks or

After content testing



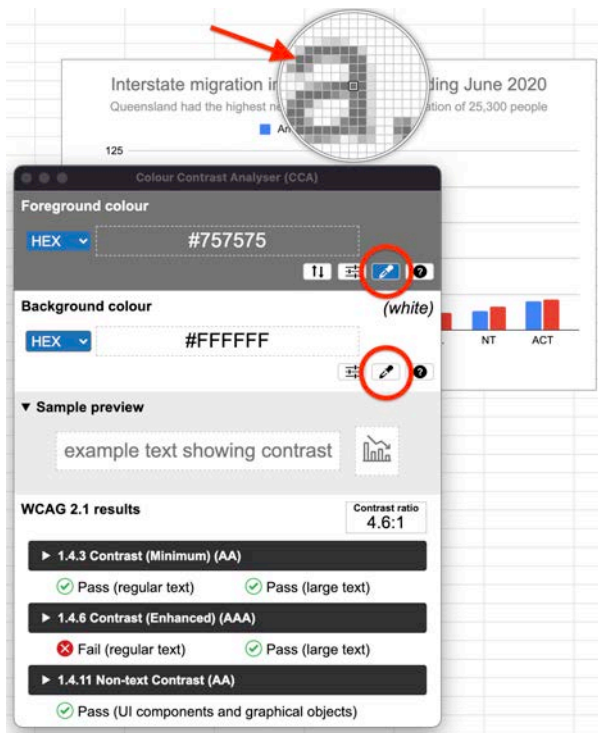
flourishes. Examples of sans-serif fonts include Arial, Courier, and Helvetica).

- The chart can be zoomed or magnified.
- If the chart has a transparent background, it meets the above contrast requirements in high-contrast (forced styles) mode.
- If the chart is animated, it must not contain flashing that could cause seizures.
- If the chart is animated, it must be able to be paused or stopped, and the animation should not automatically play.

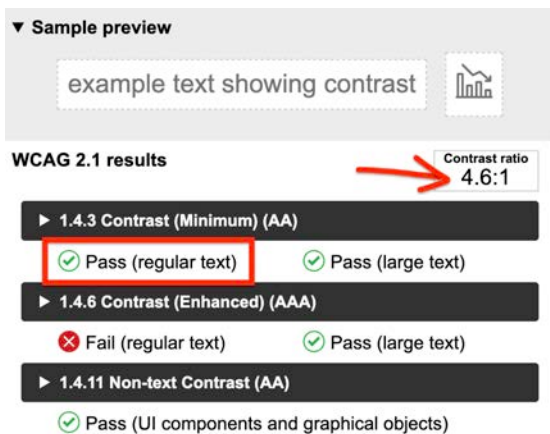
Visual Testing Process

The visual testing process requires using a color contrast checker and CVD simulator tools. I use the two tools I demonstrate here in my own workflow, but there are many other tools that you might find better fit your style, operating system, and workflow (see the list at the end of this volume for more tools).

Let’s begin with testing color contrast. First, open the TPGi Colour Contrast Analyser and select the eye-dropper tool in the “Foreground colour” section of the window, then sample the darkest color of the chart title text. If you know the hexadecimal color code or have a tool you can use to obtain it, you can also enter that code directly. Then, select the eye-dropper tool in the “Background colour” section of the window, and sample the background behind the title text.

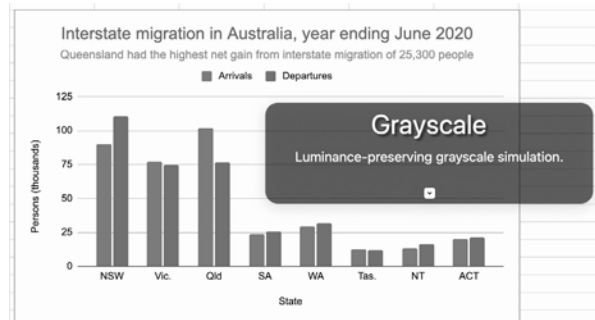
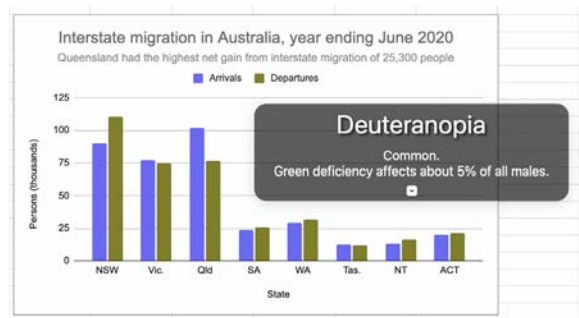


Once you have set the foreground and background colours, the “Sample preview” section will show you how those colours look together on text and on an icon. And the “WCAG 2.1 results” section displays the contrast ratio in a box in the top-right corner. For our example, the contrast ratio is 4.6:1, which means people with low vision can likely see this text color on a white background. For general accessibility testing, you can ignore any failures in the middle row (1.4.6 Contrast Enhanced AAA)—most audits only check for AA compliance. Repeat this process for all text in the chart.



Next, we’ll test for issues that may affect those with CVD.

Open the Color Oracle app (the icons will appear in your computer’s taskbar). Click on the Color Oracle icon (or right-click in Windows) and choose to simulate Deuteranopia (commonly known as “red-green colorblindness”) followed by Grayscale. For our example chart, there is very little difference in the shades of gray. This is a contrast issue: readers with CVD may not be able to tell the difference between different data series on the chart (including the legend).

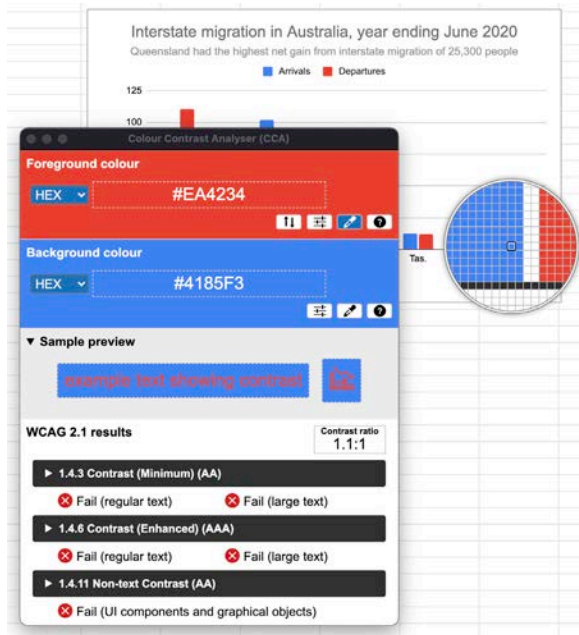


We can also cross-check this contrast issue using Colour Contrast Analyser. The results confirm a contrast ratio of 1.1:1, which explains the result from the grayscale simulation.

So where do we stand with our visual testing?

■ Passes

- **Font is sans-serif.**
- **All text is at least 12 pixels or 9 point in size, ideally 16 pixels or 12 points.**
- **The contrast ratio of shapes, icons, and large text relative to their background meets a minimum 3:1.**



■ Failures

- **Small text meets a minimum contrast ratio of 4.5:1.** “Small text” is text smaller than 24 pixels or 18 point, or bold text smaller than 18.5 pixels or 14 point. The chart summary does not meet minimum contrast.
- **Color alone is not used to denote meaning.** The chart uses color to differentiate between Arrivals and Departures.

■ Extra consideration

- **The chart is able to be zoomed or magnified.** Check that the parent webpage hasn’t disabled zoom.
- **If the chart has a transparent background, it meets the above contrast requirements in high contrast (forced styles) mode.** Not applicable.
- **If the chart is animated, it must not contain flashing that could cause seizures.** Not applicable.
- **If the chart is animated, it must be able to be paused or stopped and the animation should not auto-play.** Not applicable.

Visual Testing Fixes

To resolve these issues, we can darken and enlarge the title and summary text slightly, and we can choose new colors for the arrivals and departures data series that have at least 3:1 contrast between each other, as well as both having 3:1 contrast against the white background.

Using Color

It is very difficult to find a color set where all colors have a contrast ratio of 3:1, even for a chart with only two categories or series of data.

My recommendation is to choose a monochrome set with as much contrast between colours as possible. Perhaps you can find a color set that accounts for CVD and grayscale printing (you could also consider 3D printing of tactile charts). This contrast might be as low as 1.3:1. You might use spacing, position, labels, or interactive styles and tooltips as other ways to know what data belongs to which category.

Although patterns are widely recommended as a way to provide visual contrast, patterns should be used with care because they can actually make the chart less accessible, since patterns can be visually confusing for some readers. For more information, see “Patterns and Contrast,” HighCharts, accessed October 24, 2022, <https://www.highcharts.com/docs/accessibility/patterns-and-contrast>.

Nonvisual Testing

For a static image chart, a text description must be provided (also known as alternative text or alt text). For web developers, your `` tag must include the short description on an “alt” attribute. If the short description does not adequately explain the chart, a longer text description should be present in the article content, ideally directly before or after the chart. You can learn more about alt text and strategies in Chapters 4, 5, and 6 of this volume.

Nonvisual Testing Checklist

- There is an appropriate text description of the chart.

Nonvisual Testing Process

Usually, we can check a nonvisual description of an image by testing it with a screen reader. If you are a screen reader user, you already know how to test for alt text. Again, I recommend that everyone learn how to use a screen reader to help test your own work for accessibility.

But there is another way to test online images for alt text without using a screen reader, using the built-in debugging tools in the browser. Simply right-click on the graph and select the Inspect option at the bottom of the menu (I’m using Chrome here; your options may vary slightly). If you are reading the document version of this article, you can try this yourself by visiting this chapter’s [companion website](#). This companion website provides the text of this chapter in HTML and includes accessible, interactive charts.

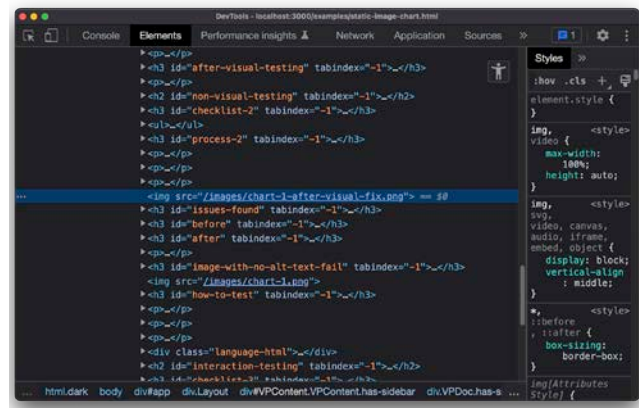
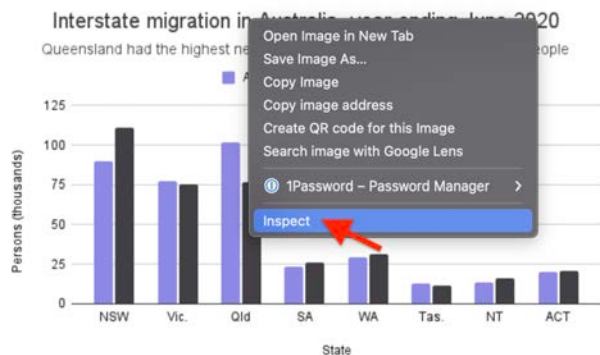
For our chart, you will find the following line of HTML in the Elements tab of the Developer Tools panel:

```

```

This line of code conveys to a screen reader user that an image is on the page, but it does not explain what the image is, what it is showing, or what it represents. Without the presence of an alt attribute, most screen

readers will read out the filename of the image (in this case, “chart-1-after-visual-fix.png”), which is a frustrating experience.



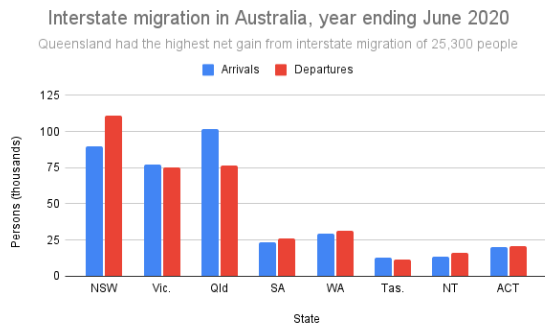
The correct HTML would include an alt attribute in the `` element, like so:

```

```

The screen reader will read this bit of text to the user so they know what the graph is and what it represents. The hardest part of this process is learning how to write a good image description for a graph—Chapters 4 and 5 in this volume go into more detail on how to write better alt text.

Before visual testing



Nonvisual Testing Results

Where do we stand on our nonvisual testing results?

- **Passes**
 - **There is an appropriate text description of the chart.** Because this essay was built with accessibility in mind, all chart images have been given text descriptions, except for the test image above.

Interaction Testing

Although we've been using a static example chart so far, what if we made it interactive by adding tooltips to each bar? To test the interactive version, we would need to do some additional tests for the three categories we've already covered. This takes us into the second part of our accessibility testing: What to do if the chart is interactive.

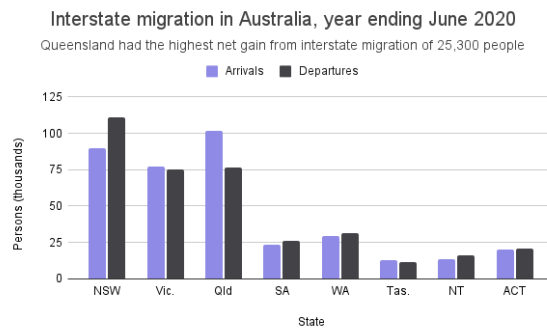
Part 2: Testing an Interactive Chart

If you are reading the document version of this article, please visit this chapter's [companion website](#) to conduct the tests described in this section.

Interactive charts are generally used for larger sets of data and allow the user to explore and form insights about the data. The charts also necessitate some added complexity when creating equitable experiences between sighted users and blind users or between mouse users and keyboard-only users.

Based on the image-only chart from part one, I've created an interactive chart using the popular chart and data visualization framework Highcharts.^[19] Thanks to the fantastic work of Ted Gies, Øystein

After visual testing



Moseng, and their team, Highcharts prioritizes accessibility in their charts.

Let's continue our accessibility testing. First, we'll test accessibility access with a keyboard, then conduct additional tests for content, visual, and nonvisual elements.

Interaction Testing (Interactive Charts)

If you are a sighted mouse user, you will be able to hover over the columns in the example chart and an informational tooltip will appear. Most interactive charts on the web will have this feature, though they tend to only show tooltips when a user uses a mouse to hover over the bar. The needs of keyboard-only and touch users are generally neglected.

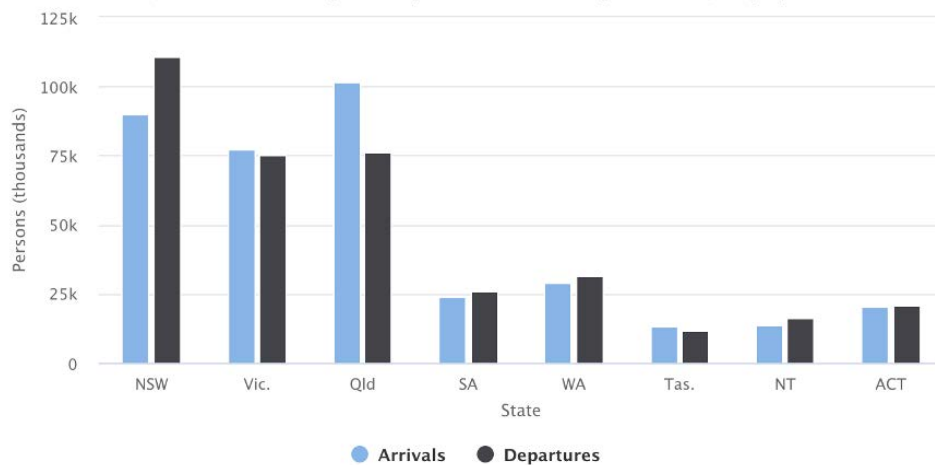
Interaction testing ensures that anything that can be accessed by mouse hover is also available to those using a keyboard or touch device.

Interaction Testing (Interactive Charts)

- Keyboard tab to the chart focuses on the first interactive element.
- The Tab key should navigate to different sections of the chart—such as the data series, legend, and menu—rather than between every single interactive element.
 - Arrow keys should navigate between data points or toggle between data series, legend items, and menu options.
 - Space/Enter should activate buttons or toggles.

Interstate migration in Australia, year ending June 2020

Queensland had the highest net gain from interstate migration of 25,300 people



- Mouse hover, pressing Space or Enter, or touch on a mobile device shows or hides annotations such as tooltips.
- The chart does not have “keyboard traps,” something that occurs when a user can get into, but not out of, a component or element of a web page when using a keyboard.
- The chart does not rely on custom keyboard shortcuts.

This checklist does not cover accessibility testing of dropdown menus, select boxes, or other components that the chart may contain, but they absolutely need to also be tested with keyboards and screen readers.

Interaction Testing Process

Using only the keyboard, use Tab key and arrow keys to check whether you can reach all parts of the chart. Make sure Tab isn't used to navigate between individual data points, because that creates a tedious experience for keyboard-only users, particularly when the dataset is large.

Ensure that buttons or toggles can be activated by pressing Space or Enter. All tooltips that appear on mouse hover should also appear on keyboard focus. Check chart interactivity on a touchscreen device to ensure that using tap input also toggles tooltips correctly.

If you get stuck in any part of the chart and cannot use the Tab, Shift-Tab, or Esc keys to exit, you're in a keyboard trap. Such traps lead to frustrating experiences for keyboard-only users, because the users will not be able to reach the interactive items on the rest of the page.

Lastly, if there are any custom keyboard shortcuts that are essential to the operation of the chart, screen readers probably cannot use them because the screen reader takes precedence when it comes to keyboard shortcuts (of which there are many), meaning many keystrokes will not make it to the browser.^[20]

Content Testing (Interactive Charts)

In addition to the content tests from Content Testing for image charts, interactive charts must also ensure the following:

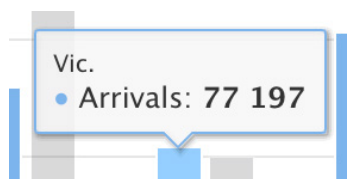
- Tooltips contain clear information

This process is similar to testing a static image chart; fix the content first, because returning to the content later could lead to drastic changes in the way the chart is presented.

Our interactive chart only adds tooltips. In the tooltip, we should include the data series label, the x-value, and the y-value, because people who use magnification may not be able to see the tooltip and

the chart axes at the same time. Having all of the information can also benefit screen reader users as they explore the chart.

Highcharts' default tooltip includes the x-value "Vic." (for Victoria), the data series label "Arrivals," and the y-value "77,197."

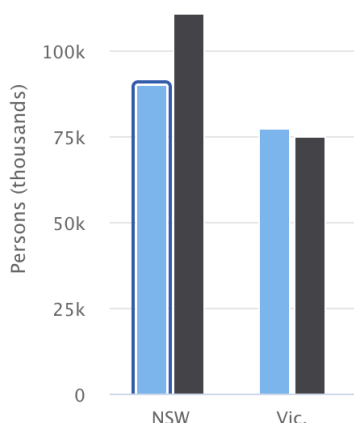


Visual Testing (Interactive Charts)

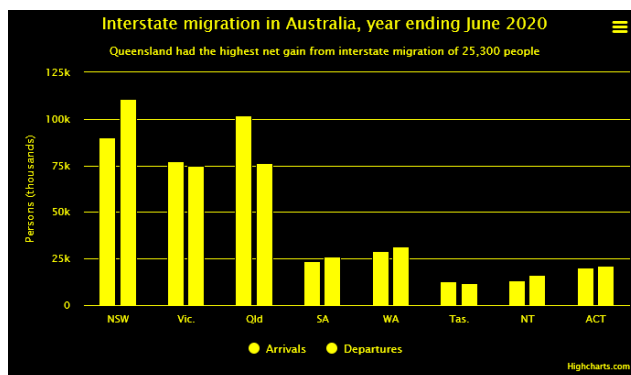
In addition to the visual tests for image charts, we must also consider the following:

- Active elements have clear focus styles that meet a contrast minimum of 3:1.
- The chart can be understood in high-contrast (forced styles) mode.
- If the chart uses animation, the chart respects the user's system animation setting. For longer animations, the graph must contain a stop/start button and must not automatically play.

While testing interaction with the keyboard, you can check that focus outlines (or other suitable focus styles) appear on the interactive element. In the example below, the dark blue outline around the first light blue bar indicates the current location of keyboard focus. The user should also be able to use the Tab and arrow keys to navigate the chart.



To test the chart with forced styles, either turn on High-Contrast Mode in Windows, or open Firefox (on either Windows or Mac computers) and use the Dark Background and Light Text add-on.

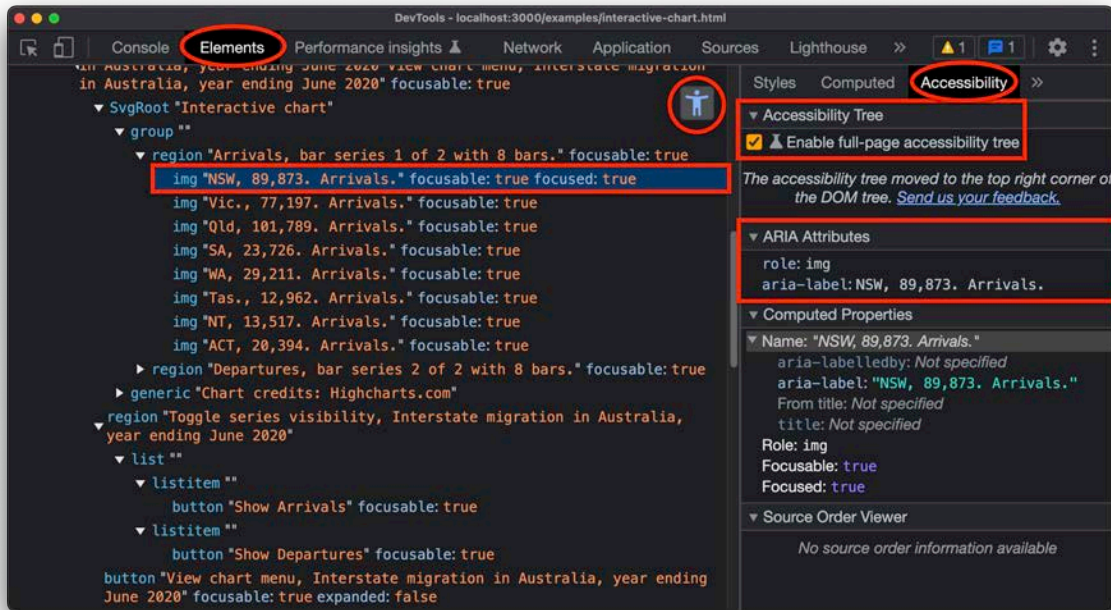


With High-Contrast Mode enabled, your system overrides application and website colors with a set of chosen "high-contrast" colors, which helps people with low vision who cannot easily differentiate between colors. In our example, all chart colors have been overridden with a bright yellow on a black background, which can cause problems because we can no longer differentiate between the arrivals and departure columns (except to hope that the legend lists them in the corresponding order). Luckily, the reader can still use tooltips or toggle on/off of each data series in order to know which data points are arrivals and which are departures.

To test animation, we need to activate the Reduced Motion setting and return to the chart to visually check whether any animation occurs. On page refresh, the columns grow vertically from the x-axis line in our example chart, and toggling the data on and off prompts some animation. Unfortunately, this interactive chart fails to respect a user's reduced motion preferences.

Nonvisual Testing (Interactive Charts)

Screen reader interaction with an interactive chart is a complex test because the way a screen reader user will navigate the chart depends on how it has been coded. As mentioned in Chapters 5 and 6, most complex or dynamic charts are coded in an SVG format, but that is not a universal rule.^[21]



If built with screen reader users in mind, the individual elements of the chart will be accessible by a screen reader through appropriate ARIA roles and labels.

In this specific example, each column in the interactive chart (we're using the [HighCharts](#) JavaScript library) has `role="img"`, and an ARIA label with the datapoint information. Screen readers will list each column as a labelled image, allowing users to inspect each one.

HighCharts is one of the few chart libraries that have this level of screen reader support—most of the popular chart tools do not even include keyboard navigation. Apart from a lack of awareness of accessibility in the software industry, very few resources exist on recommended techniques for screen reader experience of charts (except for articles like those found in Chapters 4, 5, and 6).

Nonvisual Testing Checklist

- A chart heading is provided so screen reader users can quickly skip to the chart.
- The anatomy of the chart is provided, including how many data points it contains.
- If necessary, interaction instructions are provided (for example, toggling data series on or off).

- Interactive elements on the chart have the appropriate ARIA roles and labels.
- Visual changes to the chart are announced by an `<aria-live>` element or `<role="status">`, for example whether a data series is toggled on or off.

Nonvisual Testing Process

Inspecting the Document Object Model

The Document Object Model defines the logical structure of a document—for example, a web page—and enables programmers to add, modify, or delete elements and contents from the page. We can explore the Document Object Model directly on a web page by right-clicking and selecting the Inspect option, which will enable us to see whether the chart element is using `<svg>` or `<canvas>`.

If you are familiar with the accessibility tree—a list of the accessibility properties on a web page—you can also see the semantic structure of the chart by inspecting the accessibility properties of each element in the chart or by enabling the full-page accessibility tree in the Google Chrome browser.^[22] I've provided an example above.

Testing with a screen reader

Next, open up your screen reader and navigate around.^[23]

For VoiceOver: Use Control + Option + Right Arrow and Control + Option + Left Arrow to navigate the chart by. You can jump to the chart heading first, by pressing Control + Option + Command + H until you reach it, before using Control + Option + Arrow Keys to navigate between items.

For NVDA: Press Tab to navigate to the next interactive element. Press H to jump to headings on the page, and use the Down Arrow to navigate to the next item. Press G (for “graphic”) to navigate to the next image.

Our example chart is quite screen reader accessible and passes all of our nonvisual testing criteria.

Congratulations!

If you’ve made it this far, thank you, and well done! Especially if you are new to accessibility and assistive technologies. My goal here is to make it easy for data practitioners to test their charts and visualizations. That being said, especially for someone

new to accessibility, conducting these tests could take anywhere from a few minutes to several hours, depending on the visualization. In the end, this process will save you time and help you and your team design visualizations that are accessible for a broader audience.

I cannot stress enough the importance of usability testing for accessibility, which should include a diverse range of participants. Even the most experienced auditors cannot predict the way assistive technology users and users with various disabilities will interact with a visualization. Two people with similar disabilities and assistive devices can have different ways of using their technology. You cannot make assumptions when designing data visualizations. Working with people who have different kinds of disabilities will enable you to determine how they are using your website, what needs they have, and how you can ultimately create a better experience for everyone.

Happy testing!

Appendix

The source dataset used for interstate arrivals and departures comes from “Migration, Australia, 2019–20 Financial Year” Australian Bureau of Statistics, accessed October 24, 2022, <https://www.abs.gov.au/statistics/people/population/migration-australia/2019-20>.

State	Interstate arrivals	Interstate departures
NSW	89,873	110,760
Vic.	77,197	74,954
Qld	101,789	76,441
SA	23,726	25,886
WA	29,211	31,621
Tas.	12,962	11,749
NT	13,517	16,214
ACT	20,394	21,044

Checklists: Practical Accessibility Testing for Data Visualizations

Content Testing Checklist

- Chart contains a descriptive title.
- Chart contains a suitable summary in the chart or in the surrounding text.
- Chart uses plain language.
- Chart axes are labeled clearly.
- Chart series are labeled clearly.
- The numerical data is formatted clearly.
- If the chart can be used for analysis or forming one’s own insights and is not adequately described by the chart summary, then a raw data table or a CSV download of the raw data is provided as an alternative.

If chart is interactive,

- If present, tooltips contain clear information.

Visual Testing Checklist

- All text is at least 12px (or 9pt) in size, ideally 16px (or 12pt).
 - Small text and its background colors meets a minimum contrast ratio of 4.5:1 (see explanation of contrast ratios in Color Contrast Checkers section, above). “Small text” is defined as text smaller than 24px (or 18pt) or as bold text smaller than 18.5px (or 14pt).
 - The contrast ratio of shapes, icons, and large text relative to their background color meets a minimum of 3:1.
 - Color alone is not used to denote meaning – that is, contrast, markers, or patterns are also used to differentiate data and are labeled appropriately (i.e., with a legend, or nearby text).
 - Text font is sans-serif (that is, the letterforms in the font do not have decorative marks or flourishes. Examples of sans-serif fonts include Arial, Courier, and Helvetica).
 - The chart is able to be zoomed or magnified.
 - If the chart has a transparent background, it meets the above contrast requirements in high-contrast (forced styles) mode.
 - If the chart is animated, it must not contain flashing that could cause seizures.
 - If the chart is animated, it must be able to be paused/stopped, the animation should not automatically play.
- If chart is interactive,
- Active elements have clear focus styles that meet a contrast minimum of 3:1.
 - The chart is able to be understood in high-contrast (forced styles) mode.
 - If there is animation, the chart respects the user’s system animation setting, and for longer animations, it must contain a stop/start button and not automatically play.

Nonvisual Testing Checklist

- There is an appropriate text description of the chart.

If the chart is interactive,

- A chart heading is provided so that screen reader users can quickly skip to the chart using the heading shortcut.
- The anatomy of the chart is provided, including how many data points it contains.
- If necessary, interaction instructions are provided (for example, toggling data series on or off).
- Interactive elements on the chart should have the appropriate ARIA roles and labels.
- Visual changes to the chart are announced through an `<aria-live>` element or `<role="status">`, for example whether a data series is toggled on or off.

Interaction Testing Checklist

If the chart is interactive,

- Keyboard tab to the chart focuses on the first interactive element.
- Tab key should navigate to different sections of the chart, such as the data series, legend, and menu, rather than between every single interactive element.
 - Arrow keys should navigate between data points or toggle between data series, legend items, or menu options.
 - Space/Enter should activate buttons or toggles.
- Mouse hover, pressing Space or Enter, or touch on a mobile device shows or hides annotations such as tooltips, if present.
- The chart does not have keyboard traps.
- The chart does not rely on custom keyboard shortcuts.

Chapter Seven Notes

- [1] If you're new to assistive technologies, please take a few minutes to watch these videos on browsing with assistive technology videos for an introduction to how they are used to navigate the web. See Henny Swan, "Browsing with Assistive Technology Videos," TetraLogical blog, December 24, 2021, <https://tetralogical.com/blog/2021/12/24/browsing-with-assistive-technology-videos/>.
- [2] See Chapter 4 (section 405 in particular) of "2010 ADA Standards for Accessible Design," ADA.gov, accessed October 21, 2022, <https://www.ada.gov/regs2010/2010ADASTandards/2010ADASTandards.htm#c4>.
- [3] See the Web Content Accessibility Guidelines section at "W3C Accessibility Standards Overview," W3.org, last updated June 29, 2022, <https://www.w3.org/WAI/standards-guidelines/#wcag2>.
- [4] "WCAG 2.1 at a Glance," W3.org, last updated June 5, 2018, <https://www.w3.org/WAI/standards-guidelines/wcag/glance/>.
- [5] See Chapter 2 of this volume as well as Frank Elavsky, Cynthia Bennett, and Dominik Moritz, "How Accessible Is My Visualization? Evaluating Visualization Accessibility with Chartability," *Computer Graphics Forum* 41, no. 3 (2022): 57–70.
- [6] See "The Chartability Workbook," Github.io, accessed October 21, 2022, <https://chartability.github.io/POUR-CAF/>.
- [7] As a disclaimer, I am not a professional accessibility auditor. I've learned accessibility testing as a software engineer and designer while collaborating and learning with the accessibility community. The software I've built using this testing process has produced relatively few issues in professional accessibility audits. Using this testing process will not guarantee WCAG compliance, but it will produce more inclusive data experiences.
- [8] See "Colour Contrast Analyser (CCA)," TPGi, accessed October 21, 2022, <https://www.tpgi.com/color-contrast-checker/>.
- [9] See "Change Color Contrast in Windows," Microsoft Support, accessed October 21, 2022, https://support.microsoft.com/en-us/windows/change-color-contrast-in-windows-fedc744c-90ac-69df-aed5-c8a90125e696#WindowsVersion=Windows_11
- [10] Mikhail Khvoinitsky, "Dark Background and Light Text," Firefox Browser Add-Ons, last updated February 7, 2021, <https://addons.mozilla.org/en-US/firefox/addon/dark-background-light-text/>.
- [11] See <https://colororacle.org/>.
- [12] Google Chrome Developers, "Screen Reader Basics: NVDA – A11ycasts #09," Youtube, December 2, 2016, https://www.youtube.com/watch?v=Jao3s_CwdRU.
- [13] "NVDA Keyboard Shortcuts," Deque University, accessed October 24, 2021, <https://dequeuniversity.com/screenreaders/nvda-keyboard-shortcuts>.
- [14] Google Chrome Developers, "Screen Reader Basics: Voiceover – A11ycasts #07," Youtube, October 14, 2016, <https://www.youtube.com/watch?v=5R-6WvAihms>.
- [15] "VoiceOver Keyboard Shortcuts on a Mac," Deque University, accessed October 24, 2021, <https://dequeuniversity.com/screenreaders/voiceover-keyboard-shortcuts>.
- [16] Choosing the correct chart format is important for helping your reader understand your message. A column chart is probably not the best choice for the data we're using, but it is an easy format to test. Good accessibility begins with design, and I highly recommend reading the following two resources: Cole Nussbaumer Knaflic, *Storytelling with Data* (Hoboken, NJ: John Wiley & Sons, 2015), <https://www.storytellingwithdata.com/books>, and Jonathan Schwabish, *Better Data Visualizations* (New York: Columbia University Press, 2021), <https://policyviz.com/books/>.
- [17] See <https://hemingwayapp.com/>
- [18] See also Schwabish, *Better Data Visualizations*.
- [19] See <https://www.highcharts.com/>
- [20] See Léonie Watson, "Time to Revisit Accesskey?" Tink.uk, April 29, 2015, <https://tink.uk/time-to-revisit-accesskey/>.
- [21] There is also a popular graphic display element in HTML called `<canvas>`. The HTML canvas allows a programmer to build visuals using geometric elements such as circles and rectangles along with x and y coordinates. It allows developers to build charts using programmable graphic elements and interactions (`<canvas>` is often used for web 3D graphics or web games). But the elements within a `<canvas>` are not reachable by the screen reader, and it takes quite a lot of JavaScript knowledge to make it an equivalent experience. When I see a chart built in `<canvas>`, I assume it is not accessible to screen reader users or keyboard-only users.
- [22] "Accessibility Tree," Mozilla Developer Network, last modified September 20, 2022, https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree.
- [23] Navigating the chart by screen reader is not the same as navigating the chart using the keyboard. You will not see the focus outline nor the tooltips when navigating between data points. Screen readers use the "virtual cursor" to navigate the page. When screen reader shortcuts are pressed, like the Down Arrow in NVDA, keyboard events are not sent to the browser (i.e., the browser doesn't know you've pressed the Down Arrow). This is a common misunderstanding for people who are new to using screen readers; see this comment by Oysteinmoseng on "A11y: Highlight Problem When Using NVDA," Highcharts Github repository, issue #15303, March 11, 2021, <https://github.com/highcharts/highcharts/issues/15303#issuecomment-796670885>.

Infographic Equity in PDF Documents: Designing with Accessibility in Mind



DAX CASTRO

Current data visualization trends rely heavily on infographics to convey meaning, but this method frequently creates unintended accessibility barriers. In my work as a PDF remediator (meaning I make PDF documents accessible), I have seen countless infographics that are unapproachable or that contain meaningless descriptions for people who rely on assistive technology or keyboard-only interaction.

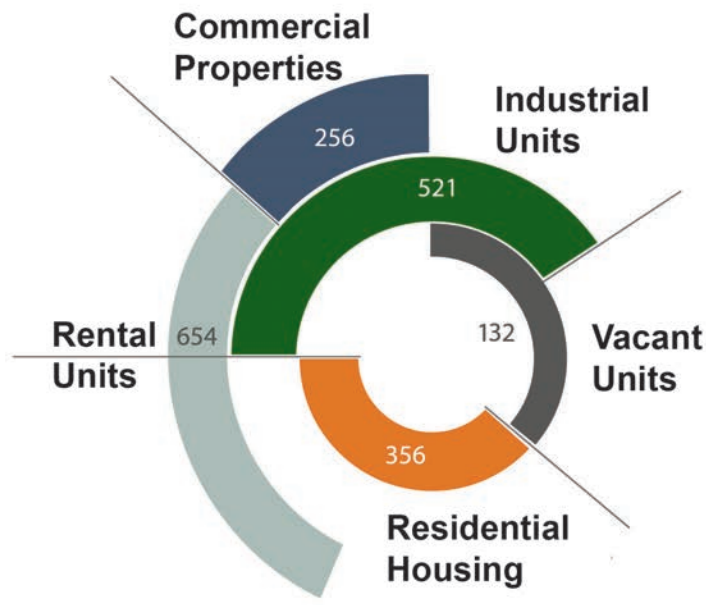
In my experience, content creators usually take an “accessibility at the end” approach to design. They design first and then try to figure out how to make that design accessible. This essay explores different ideas for creating richer experiences for people who use assistive technology such as screen readers, text-to-speech software, or text-to-braille devices to interpret data visualizations.

Not everyone perceives all inputs equally. We must take additional steps to increase what I call infographic equity. Simply put, infographic equity is the attempt to provide an equal experience for all readers regardless of the methods they use to read or interact with a visualization. Let’s start with a simple example. Consider the alternative (alt) text in the sunburst diagram on the following page.

We have an image of a complex sunburst diagram showing five different property types. Each one has a numeric value attached to it. Technically, alt text version one (“Investment Property Sunburst Diagram”) would pass any accessibility checker, but it contains almost no meaningful data. Version two has a single value and explains the main takeaway the author intended by including the chart. Again, this is technically compliant, but is it an equitable experience given the other four missing data values?

The third version of alt text contains the full meaning of the chart with all values and associations, including the author’s intent. But what should you do when you have a bar chart with 50 data points or a graphic with a complex story? There are several ways we can create more accessible experiences.

This essay focuses on four main methods for presenting richer, more meaningful visual content in PDF documents. In particular, I focus on infographics, a broad term that describes how we present data in a visual way to make complex information easier to



Alt-text Version 1:

Investment Property Sunburst Diagram

Alt-text Version 2:

Sunburst diagram showing Investment Property by type. Rental units make up the majority of the investment with 654.

Alt-text Version 3:

Sunburst diagram showing Investment Property by type. Commercial at 256. Industrial with 521. 132 Vacant units and 356 Residential housing units. Rental units make up the majority of the investments with 654 units.

understand. Unlike singular data visualizations, such as standalone charts and graphs, infographics incorporate text and images together to convey a complex idea in a creative, easy-to-understand way.^[1] Infographics may take the form of printed, standalone, leave-behind documents; downloadable PDFs; or large image files. Some infographics tell stories, others present data in a series of steps, and others simply present data for users to explore and understand. Before we can learn how to create more meaningful experiences, we must understand that the point at which we consider accessibility during the development phase is critical.

When You Start Matters Most

It is important to incorporate accessibility considerations at the earliest stage possible. The stage at which we consider accessibility directly impacts how long it will take to make the product accessible, what approach we use to do so, and ultimately how usable and informative the content is. If we start late, our goal of a single, equitable, accessible experience for everyone often fails because “there’s not enough time” or “this has already been approved and we can’t change it.”

When teams start designing with accessibility in mind from the beginning, there are more options to present a single, meaningful user experience. Depending on what stage you begin incorporating accessibility, you can increase or limit your ability to employ accessible design by affecting your timeline or your attachment to current designs.

Starting at the Concept Stage

In addition to obvious considerations like choosing an accessible color palette (see Chapter 9), designing an interactive experience for accessibility starts with the most important question of all: “What do I want the user experience to be?” The answer to this question affects every aspect of your design and implementation. Without answering this question, most developers find themselves at the end of the project with several accessibility barriers they had not considered. And listening to an image description is an interactive experience that should be considered. Mapping out the user experience will help identify how the information is presented and possible options for the user to interact with it while absorbing meaningful content.

Other considerations should include the following:

1. Am I using color alone as a way to understand content or interactions?
2. Am I providing keyboard-based navigation that offers important information and context?
3. Does my presentation method meet minimum contrast requirements?
4. Am I providing text alternatives for image-based data visualizations?

Considering accessibility in the concept stage allows the most freedom for change and has the least impact on schedule and development.

Starting at the Draft Stage

If you are starting to consider accessibility once a draft has already been developed, you still have time to make more accessible design and implementation choices. Color palettes can be reviewed. Text alternatives and presentation methods can be evaluated with assistive technology for barriers and meaningful information. But any changes may be harder to implement at this stage. Those who approve the content may have already formed attachments to the draft colors and methods. The best way to shift sentiment at this stage is to provide solutions when encountering barriers. In my career, I have experienced overwhelmingly negative pushback when my evaluations only surfaced accessibility failures without also providing solutions.

Starting at the Final Stage

Even if you are starting at the end, there is still hope for creating an accessible product. With many visualizations, you can slightly darken colors to meet contrast thresholds, adjust alternate descriptions to be more meaningful, and modify tags or tag-only content to present a better user experience for those using assistive technology. But taking these steps is not always easy. Starting accessibility work after the project is fully developed is like baking an apple spice cake for someone who doesn't like apples and asking, "can't you just take them out?"

Now that we've established that the earlier we consider accessibility in our designs, the better they'll be, let's walk through some approaches, both basic and complex, for creating accessible visualizations and PDFs.

The Basics

Several essays in this volume have discussed strategies and tools to write sufficient alt text for graphs and images. The Web Accessibility Initiative—an international public-interest nonprofit organization working to develop standards for the web—has established a basic approach for providing text that describes nontext content such as images, charts, and graphs.^[2] Although the Web Accessibility Initiative technique suggests supplying short, meaningful alternative descriptions, that approach can leave a substantial information gap between what a visual reader can perceive and what is provided to people who rely on alternative descriptions or formats. Even though the Web Accessibility Initiative does not suggest a minimum or maximum length for alt text, the most common approach is two to three meaningful sentences about the key features or important data. But how can we present a richer user experience? How can we relay more than overviews and summaries while providing an equitable user experience?

Again we must ask ourselves the driving question mentioned earlier: What is the expected user experience? This question drives every decision we make. Do we want to enable users to explore every data point in a chart or graph? Do we want to explain relationships or tell stories?

The environment of the presentation also matters. In an HTML web page, we can use a variety of tools to enable readers to interact with the visualization. Image maps, JavaScript code, ARIA-described tags (see Chapter 5), and long-description ("longdesc") tags can all be placed along an image on a webpage. In a PDF document, we do not have access to the same HTML tags, so we need to take different approaches.

Accessibility Tools & Terms

Some of the benefits of presenting complex nontext content in a PDF document include the ability to use layers, content key metadata, alt text, actual text, and figure tags. Another unique benefit is the ability to maintain visual presentation while providing an alternative experience through assistive technology. Testing your approach with a screen reader or other assistive technology is key when applying any of these methods to verify the technique provides a good user experience (see Chapters 6 and 7).

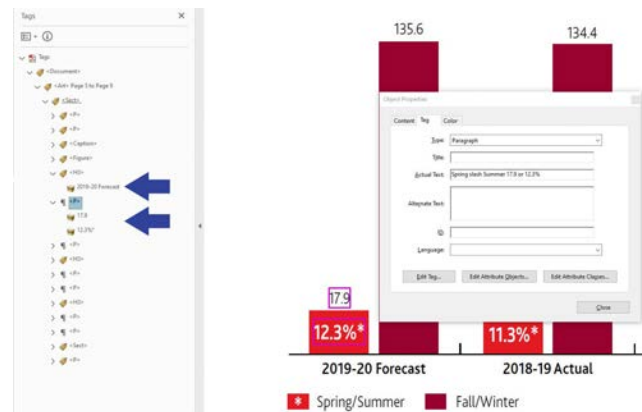
- **Figure tag:** A structural marker given to a nontext object that identifies it and holds additional programmatic information about location, content type, and other properties.
- **Content key:** A property of the figure tag that holds descriptive text in accordance with PDF/UA (Portable Document Format/Universal Accessibility) requirements. However, this value is not currently being voiced by JAWS (Job Access with Speech) or NVDA (NonVisual Desktop Access), two of the major screen reader tools currently available.
- **Alt text:** Alternative description that describes the meaningful elements of a photo or graphic. This text cannot be paused when being read. It should be two to three meaningful sentences and avoid excessive punctuation.
- **Actual text:** A one-to-one replacement for images of text (e.g., “Sale.” for a photo of a banner that says ‘SALE’).
- **Expansion text:** Description metadata for defining acronyms or abbreviations. (e.g., “IAAP: International Association of Accessibility Professionals”).
- **JAWS:** A commercial screen reader used mainly in professional environments.
- **NVDA:** A free screen reader used worldwide.

The Four Basic Approaches to Richer Infographics

In addition to alt text, the following four approaches push the envelope for richer infographic presentations. I won’t show all of the steps to add or edit these features, but these techniques can be a great starting point for you and your team to consider when building more accessible content into your PDFs.

1. Repurposing text labels for charts and graphs:

Accessible bar charts, pie charts, and line graphs should include clear labels and data points for each meaningful element in the series. If we keep these objects as selectable text, we can target these elements as headings inside the tags tree (indicated with blue arrows in the next image) and apply “actual text” to include the repeating axis information along with the actual data value as shown in the “actual text” field. This approach allows the user to step through each data point at their own pace. Giving the user control over how they review the data is far better than presenting them with a generalized alternative text description that cannot be paused for data-point review.



Alternatively, you can tag each bar as a figure and add alt text that includes the data point, axis labels, and any trends or meaningful correlations visually present. You can use the Content Panel in the PDF document to find each element and create a figure tag from each group. Any axis labels can be repurposed as headings. The main difference of this approach would be that a person using a screen reader would hear “graphic” for each

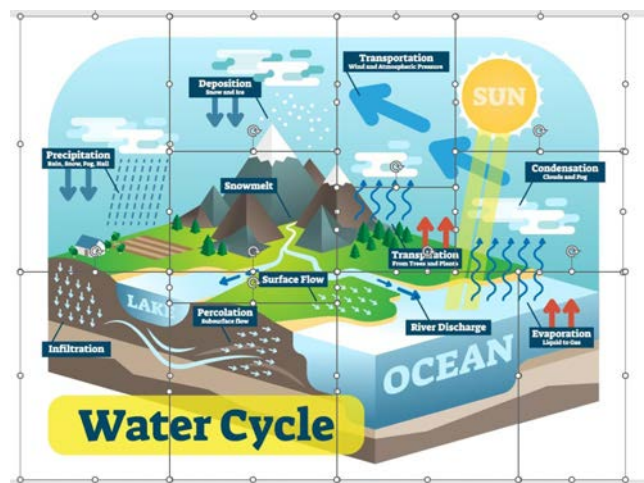
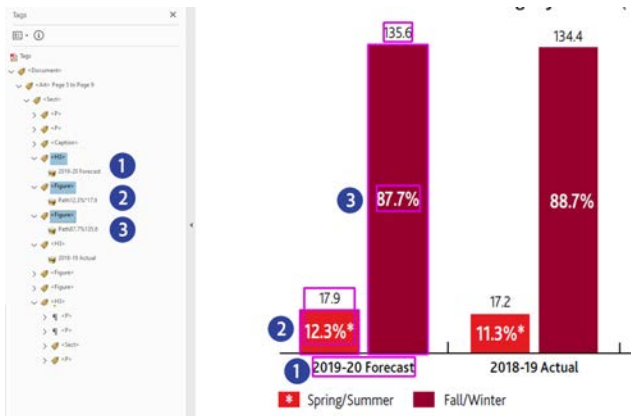


figure element before the alt text is read. Again, your approach is dictated by the question “What do I want the user experience to be?”

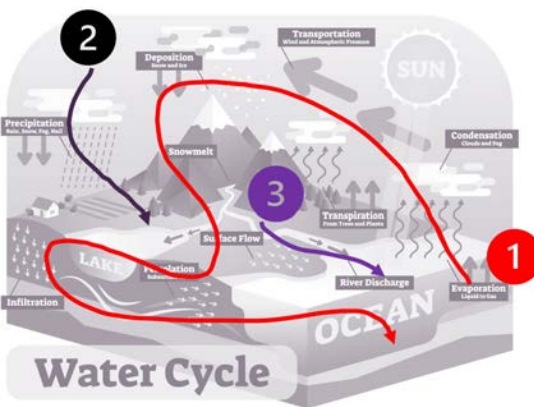
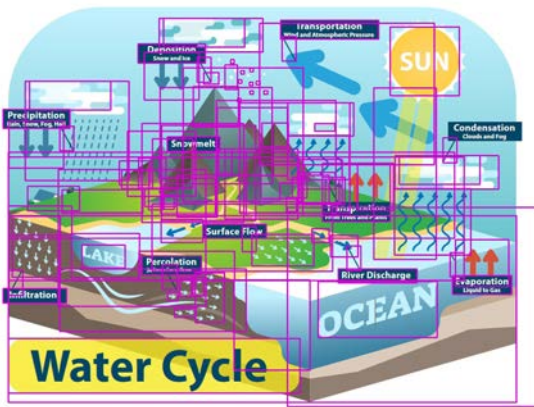
2. Breaking up the pieces: If you are a graphic designer, you should be familiar with the idea of slicing. This is an older method of presenting large graphics in web pages by cutting the image into several pieces that load more quickly on the site. In a PDF document, you can use alt text to employ a similar approach by presenting a series of images that tell a sequential story. The basic approach is to start the series by introducing the series of graphs to the reader. In the image below, the first piece of alt text might say, “The following nine graphics illustrate the lifecycle of water. First, water evaporates from the ocean heated by the sun’s rays.” The number of slices is dictated by the depiction of the graphic. Then we

apply appropriate alt text to each slice of the image to describe the process or data values as needed.

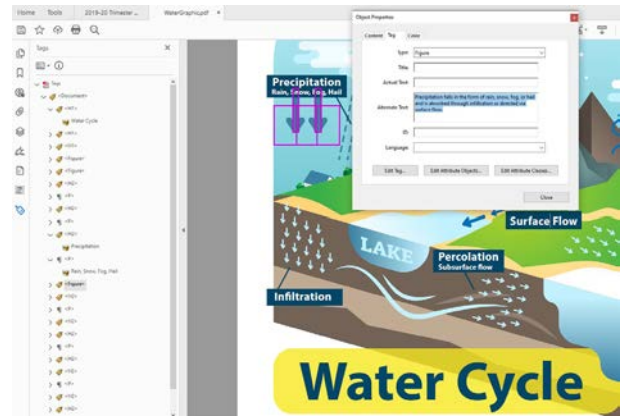
As the content creator, you will need to figure out how many separate thoughts are presented and ensure that you have sliced your images into enough pieces to tell the story. It is not critical that the visually represented slices are perfectly limited to the elements in the description. For screen reader users, the story being presented needs to represent the correct number of descriptions to tell the visual story. But there is a balance to consider for those using assistive technology who also have sight. You should slice your images in a way that considers the user might try to draw correlations between the slice and the associated text. Be as close to logical as possible given the complexity of your graphic.



3. Accessing Taggable or Live Content: This approach is similar to how we selected the bars in the first approach. If we keep our graphic vector based (e.g., in Adobe Illustrator or Encapsulated PostScript file formats) and taggable, we can tell stories through these elements using a combination of imagery, symbols, and text rather than using a single raster-based graphic (e.g., JPEG and PNG). Keeping elements separate and taggable allows us to create accessible experiences in the source document when using programs like Adobe InDesign or by tagging them individually in the PDF. How you tell the story is important so the user can step through the elements in a logical and meaningful way. This order will also determine the tag order.



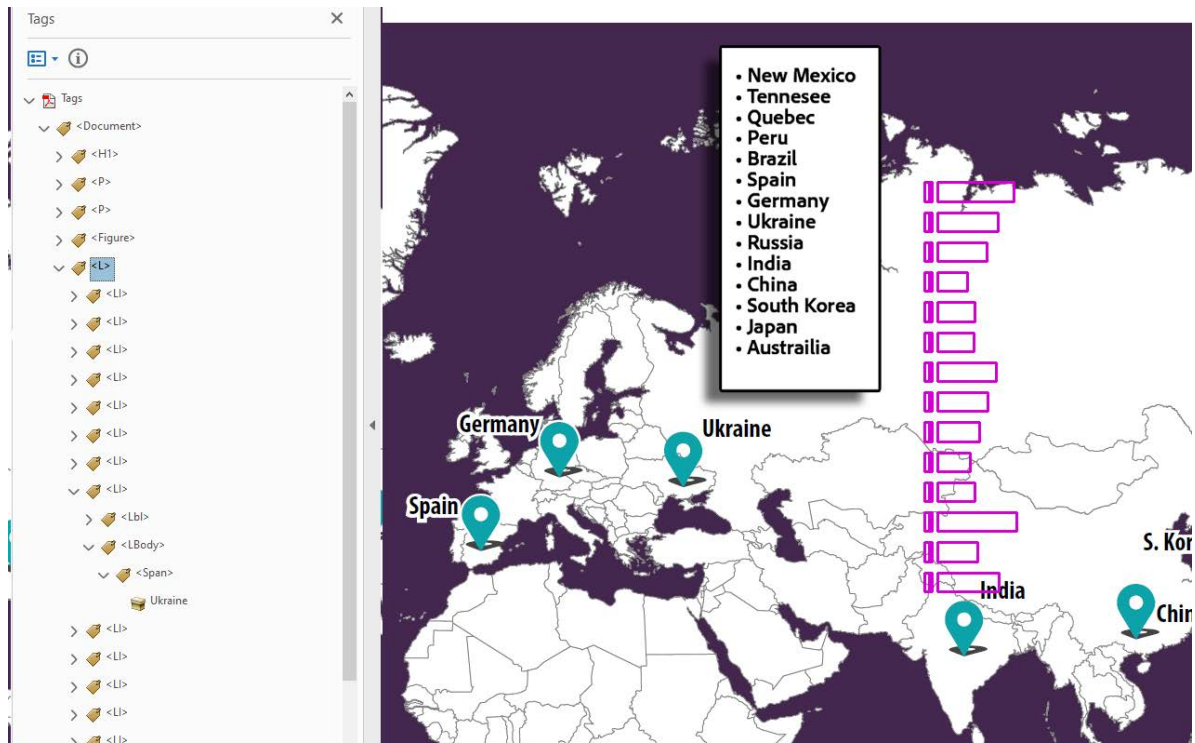
4. Using layers: One of the easiest ways to present a text-based alternative experience is by using Adobe Acrobat's ability to layer content. This feature is especially useful when we need to show things like organizational charts, process trees, or other text-heavy infographics.



The process of placing images on top of background text allows the visual reader to have the experience that they are used to and allows assistive technology users to access structured, meaningful, accessible descriptions. This layered approach is especially useful for organizational charts that can be presented with nested lists and headings to mimic the visual presentation. With flow charts, for example, the text equivalent can follow the same format.

If we evaluate the world map graphic on the following page, we can see there are pin points all over the map, making it essentially one big list. We could use the accessible tagging approach (approach number three) to tag the live content if the graphic were placed in an Adobe Illustrator file. But if the graphic was a PNG or JPG image file, then using layers might be our best approach. We place a text-based list under (behind) the map that is accessible to people using assistive technology. This method is most easily implemented when we have access to files where we can control the flow of text. It is also possible to add this list using Acrobat, but that process is much longer and requires individually creating a series of List Item () tags and placing the text in each one.

Again, designing with accessibility in mind saves time and presents a more informative user experience.



Conclusion

No matter which of the four solutions or combinations we use, the first question we must always ask of our graphic is “What is the expected user experience?” The answer will provide more clarity for the path that is most appropriate for your situation and your content. The development of accessible infographics hinges on the premise of designing with accessibility in mind. The accessibility skill level of the remediator must also be considered. Each approach has time and skill considerations that can affect your project. The sooner you consider accessibility solutions, the better prepared you will be to implement your approach and test the outcome against your desired experience.

“Accessibility at the end”—where accessibility features are tacked on at the end of the project rather than throughout the entire workflow—limits our creativity

and our ability to reach more readers and users. Meeting the basic guidelines of the Web Content Accessibility Guidelines creates compliance, but considering the user interaction pushes us closer to a more equitable user experience for all. Our expected outcome dictates our approach. Understanding what we expect the user experience to be molds every decision we make from that point forward. I am a firm believer that infographic equity can be achieved. Accessibility does not have to be ugly, and we can create a good user experience with a little planning. First, we just need the answer to the ultimate question, “What do I want the user experience to be?”

Chapter Eight Notes

^[1] Matthew Pritchard, "Data Visualization vs. Infographics," Infographics, Killer Visual Strategies News & Updates, Visual Communication (blog), December 1, 2016, <https://killervisualstrategies.com/blog/data-visualization-versus-infographics.html>.

^[2] "Tips and Tricks," World Wide Web Consortium Web Accessibility Initiative, last updated April 12, 2017, <https://www.w3.org/WAI/tutorials/images/tips>



PART FIVE

Accessibility in Teams
and Organizations

Building Accessibility Best Practices Into Your Organization's Data Visualization Style Guidelines



AMY CESAL

Data visualization style guides offer the potential to build accessibility best practices into organizational workflows and culture. Organizations, their designers, and their content teams often rely on style guides to maintain brand consistency in terms of color, font, feel, and visual style, but style guides are more than just guidelines. They can set organizational standards and function as tools for organizational process change.

Although design style guides have typically focused on the traditional aspects of a brand's visual style, I and others have argued that these guides should be extended to include rules for data visualization as well, with a particular focus on accessible visualizations.

Because style guides can establish standards for the content an organization produces, they provide a powerful opportunity to incorporate accessibility standards from the ground up. Including a topic within a style guide signals importance and organizational commitment. Because data visualization style guides detail standards and practices for every aspect of a visualization, such as colors, fonts, visual encodings, and branding, these standards can ensure production visuals meet the highest standards of accessibility.

In previous work,^[1] I defined data visualization style guides as “standards for formatting and designing representations of information, like charts, graphs, tables, and diagrams. They include what (e.g., types of charts) and why (e.g., reasons for using specific colors).” Although style guides are distinct from templates, it is often common (and helpful) for organizations to produce templates for the tools they use to accompany the style guide (such as Excel, R, D3.js, or Tableau). These templates can show how to create charts that meet the standards set out in the guide and make it easy for people to apply the standards.

Data visualization style guides should be designed to fit within an organization's larger design system. Style guides maintain uniformity across the different tools and software that produce charts. An organization's charts should be consistent across tools and look visually similar to the publications they are part of. Having a style guide with principles and components that work across multiple

tools—rather than just one template for one tool—helps achieve this consistency. Consistency builds trust and ensures minimum quality standards are met.

From an accessibility standpoint, creating a data visualization style guide allows an organization to codify best practices into their data visualization design and production workflow. Accessible design considerations should be built into every aspect of the style guide, including color selection, typography, line thickness, and the use of white space. It is also appropriate for style guides to have a section devoted to additional accessibility measures beyond just visual aspects. This section can educate team members about the importance of accessible design and the ubiquity of users who benefit from accessible design, and it can provide tools such as alternative (alt) text and Accessible Rich Internet Application (ARIA) labels to make designs work better with screen readers and other assistive devices.

Accessible Design Best Practices

Data visualization style guides commonly include sections devoted to different aesthetics, like color or typography. In each of these, accessibility should be considered and enumerated for team members as they build and style their graphs, charts, and diagrams.

Color

Often when we talk about accessibility and color, we talk about color blindness, particularly [deuteranopia](#), also known as red-green color blindness, a condition experienced by about 8 percent of men and about 5 percent of women of northern European ancestry.^[2] The focus on color blindness to the exclusion of other color accessibility considerations mirrors our collective tendency to pay the most attention to problems that affect (usually white) men. Although designing to accommodate color blind users is certainly important, we can do more.

About 6 percent of the US population has some sort of vision impairment. Thus, we should also consider the needs of users with low vision, and we can do so by ensuring there is enough contrast between

foreground and highlight colors and the chart background color. Colors within a palette should also be visually distinguishable from each other; this is a particular challenge when an organization has a large color palette.

The contrast ratio between foreground chart colors and the chart background should be measured according to the Graphical Objects and User Interface Components standard from the Web Content Accessibility Guidelines and can be tested using a tool such as WebAIM's Contrast Checker tool. This tool includes values for compliance based on guidelines defined by the Web Content Accessibility Guidelines, an international group responsible for setting guidelines for the web (see also the appendix to this volume for a list of similar tools). Color blindness checkers like the Coblis Color Blindness Simulator can be used to test whether colors within a palette are too similar for someone with color blindness. Larene Le Gassick discusses a variety of other tools in Chapter 7, and others are listed in the appendix at the end of this volume.

Some colors have strongly associated social meanings developed over repeated use. In US politics, blue is strongly associated with Democrats, and red with Republicans. Green means “go,” yellow means “caution,” and red means “stop” or “warning.” Style guide color selection should be conscious of, and work with rather than against, these associations. Using colors in a manner inconsistent with existing associations increases the cognitive load on the viewer.

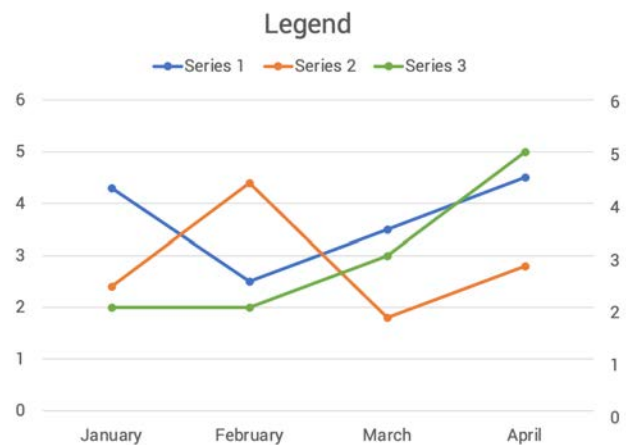
Practitioners should either not use color pairings or palettes that have particularly strong preexisting associations or should use these colors in alignment with users' expectations. At the same time, some existing color associations (e.g., blue for boys and pink for girls) may be problematic because they reinforce existing stereotypes, so practitioners should be cautious about their usage.^[6]

On the next page is a brand-consistent color palette I created for the Sunlight Foundation that includes consistent categorical colors for data we frequently visualized (e.g., spending and political party), which played into existing associations.

When choosing brand-consistent palettes for a style guide, designers attempt to optimize for brand recognition, visual distinctiveness, color vision deficiency friendliness, number of colors, contrast, and any potential implied or associated meanings of colors. But one set of colors may not be capable of balancing all those considerations and achieving all those aims. Instead, a set of palettes within a style guide should represent compromises across these considerations to meet the organization's needs and the needs of their users. Some palettes might be better than others for achieving certain aims, and organizations can have several approved options for designers to choose from depending on the application. A detailed guide to colors in data visualization style guides by Lisa Charlotte Muth goes into further examples from organizations.^[6]

For some users, color will always pose a problem even when we do our best to find accessible palettes. Given the constraints that color presents, we should consider how much we really want to ask colors to do in our charts. Often it can be helpful to “dual encode” our data—that is, combine color and another shape or texture to communicate the data. In a scatterplot, points can be colored by a categorical variable and have their shapes determined by that variable. Dual encoding this way always increases accessibility and makes a chart more readable to a wider audience. The image on the following page, again from the Sunlight Foundation, provides an example of dual encoding in a scatterplot.

In line charts, instead of dual encoding, we use direct labeling. Whenever possible, directly label a chart's lines, points of interest, or bars. This feature tends to be more accessible and requires less work from a reader than a chart legend, which requires the user to look back and forth between the legend and the chart area.



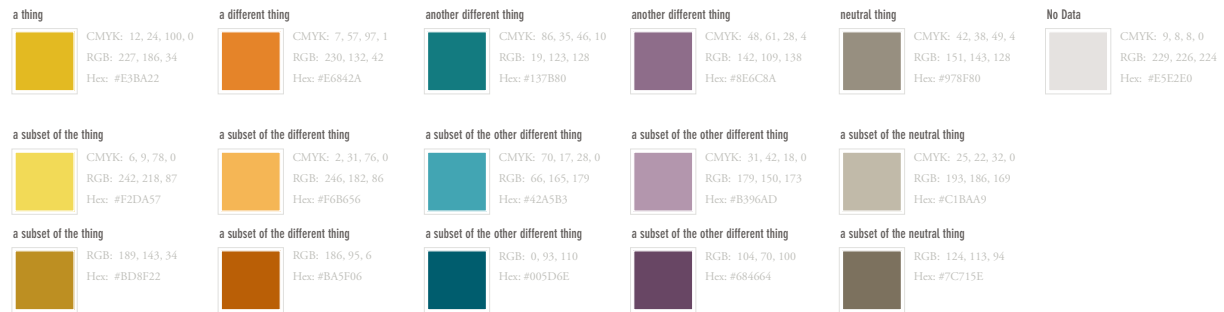
Similarly, a chart with multiple abutting colors, such as a stacked bar chart, can be difficult for a vision-impaired user to process (see top of page 90). Separating each of these color sections with a white stroke is an easy way to provide visual distinctiveness between sections, increasing contrast and accessibility.

Labeling and Typography

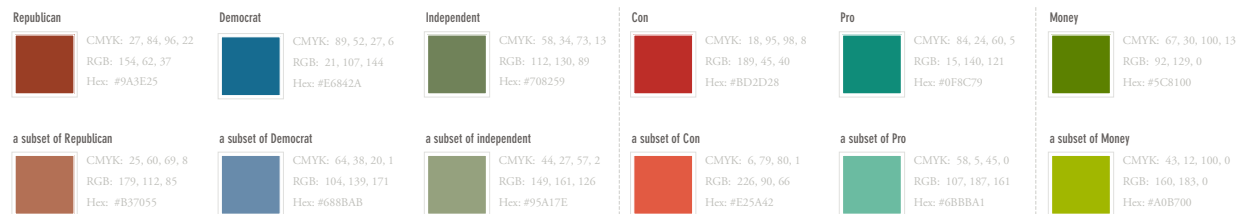
When writing the text for a chart, we need to consider what we say and how we say it. The style guide specifies how headings, subheadings, axis labels, and legends should look and be written. Using plain language goes a long way toward ensuring that the broadest possible audience can understand a chart—the Plain Writing Act of 2010 defines plain language as “writing that is clear, concise, well-organized, and follows other best practices appropriate to the subject or field and intended audience.”^[8]

Data Colors

Main Colors

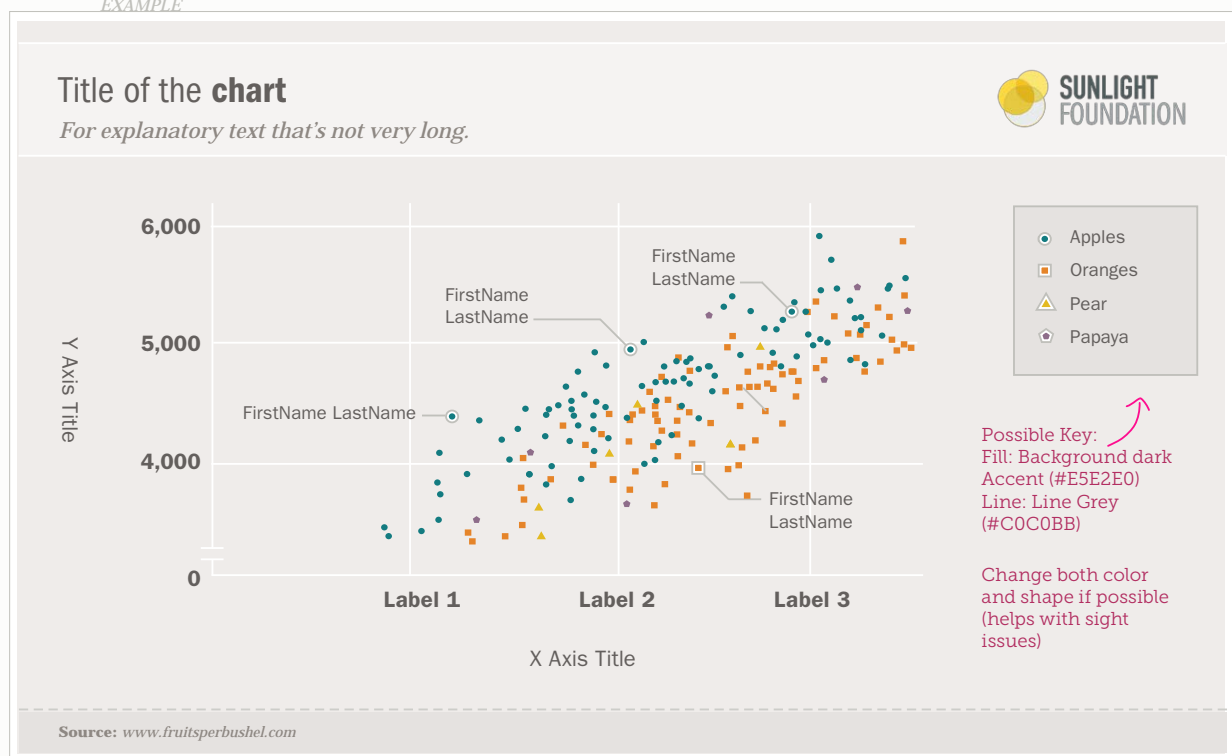


Specialty Colors

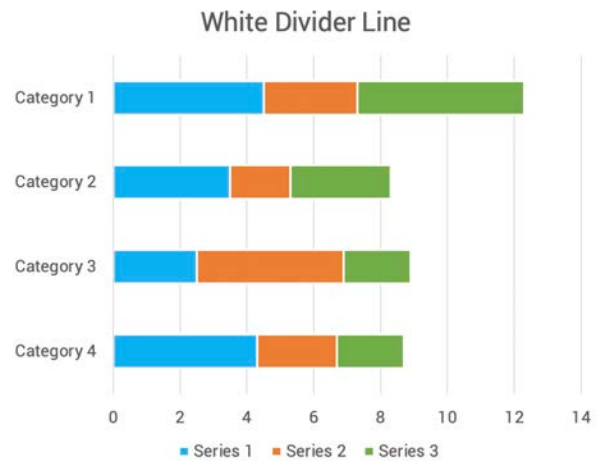
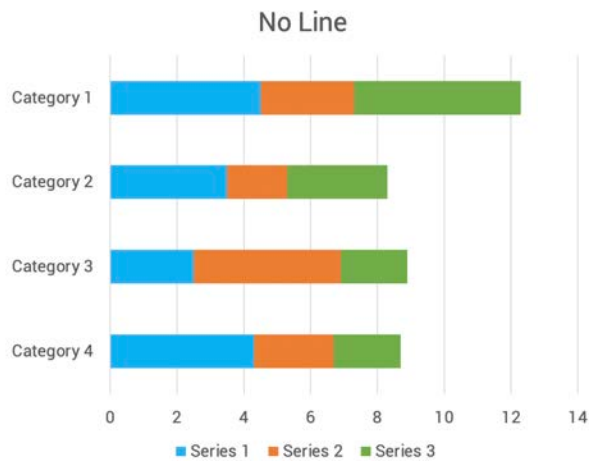


Source: Sunlight Foundation, "Data Visualization Style Guidelines," accessed October 13, 2022, <https://github.com/amycesal/dataviz-style-guide/blob/master/Sunlight-StyleGuide-DataViz.pdf>.

EXAMPLE



Source: Sunlight Foundation, "Data Visualization Style Guidelines," accessed October 13, 2022, <https://github.com/amycesal/dataviz-style-guide/blob/master/Sunlight-StyleGuide-DataViz.pdf>.



A data visualization style guide should provide guidance on how to use labels, titles, and legends effectively, and it should detail the most readable fonts, sizes, and spacings. For maximum accessibility, charts should include “takeaway titles”^[9] that succinctly communicate the main takeaway of the visualization. By telling viewers what you want them to get out of a chart, you prepare them to view the chart with that in mind, decreasing the cognitive load required to understand the data presented. Titles should be pitched to the level of the chart’s intended audience, communicating findings in terms familiar to them. Subtitles or footers can be used to provide more technical details if necessary for accuracy.

In terms of visual design, typography needs to be large enough and have high enough contrast relative to the background to serve low-vision populations. For maximum legibility, the number of different fonts used should be kept to a minimum—the chart should be the focus rather than the fonts. Across charts, be consistent with fonts as well as the sizing of titles, labels, and legends; this will maintain brand identity and ensure all charts produced by an organization are readable.

Historically, sans-serif fonts (meaning letterforms that do not have extending features at the ends) have been considered easier to read on computer screens because of low screen resolutions. Now that high-quality screens are more common, that thinking is less true. Current evidence on font readability is mixed

and suggests there is not a universally accessible font. Rather, font readability depends on the individual and the kind of disability they have. In this context, I suggest choosing relatively simple and common fonts and to use few font families in the same chart. Condensed fonts are often useful in data visualization applications where space is limited, but their compact nature can hinder vision-impaired users.

In general, the most important considerations for text on charts are maximally informative content and appropriate spacing, size, and contrast. Rather than leaving these selections to ad-hoc decisions made for each individual chart design process, data visualization style guides set up rules for functional and accessible chart text.

Layout and Style

A style guide should include layout instructions that specify the spacing between chart area, legends, titles, axis labels, footer, the organization’s logo, and any other visual features that go on a typical chart. Overcrowding can be a serious impediment to readability because charts that are cluttered are more difficult for users with visual or cognitive impairments. Unnecessary clutter increases cognitive load, and insufficient spacing between items may cause them to bleed together when viewed by those with vision impairments. By prespecifying the spacing of elements in a chart, every chart will be more consistent, readable, and accessible.

Accessibility Section

Although accessibility should be built into the design choices throughout the style guide, an accessibility-specific section should also be included. Here, you can provide specific instructions on how to write alt text for data visualizations and how to include alt text in whatever publication formats will be used for distribution (see Chapters 4 and 8 of this volume for strategies).

This section should also discuss the reasons that accessibility is an important consideration. When charts are made with accessibility in mind, they are better and useful for everyone, not only for people with disabilities.

Accessibility-Specific Additions

Thoughtful visual design with accessibility in mind is a good start, but it is not enough on its own. Additions need to be made to improve accessibility for screen readers. These additions also improve general computer readability (including search engine optimization, making your content more easily found through web searches).

Alt Text

Alt text provides a description of static-image charts in digital mediums and is added as a tag on the image. Organizations should define how to add this tag for their content and what should be included in it to help people understand the content of the chart.

[WebAIM](#), a group housed at Utah State University, explains that alt text allows the content and function of the image to be accessible to those with visual or certain cognitive disabilities. They also add that alt text provides a semantic meaning and description to images that can be read by search engines or be used to later determine the content of the image from the page context alone.^[10]

As I have written for the Nightingale blog,^[11]

- If an image contains meaningful information, adding alt text is better than not doing anything at all.

- If you can add HTML properties to the image, add a long description^[12] to more fully convey the image's meaning.
- Supplement your image with a link to the raw data so curious readers can access the data in their preferred program.
- Keep alt text short. Alt text is read linearly by screen readers, which means that people can't go back a word if they miss something. For search engine optimization purposes, Google cuts off its crawl after a certain number of characters.

Alt text for data visualization can follow the following format:

alt= "**Chart type** of **type of data** where **reason for including chart**"

Include a **link to data source** somewhere in the text

ARIA Labels

ARIA labels can be applied to interactive visualizations to help users understand the data elements being displayed.

WebAIM says, "[WAI-ARIA \(Accessible Rich Internet Applications or ARIA\)](#) is a W3C [World Wide Web Consortium] specification for enhancing accessibility in ways that plain HTML cannot. When used properly, ARIA can

- enhance accessibility of interactive controls, such as tree menus, sliders, pop-ups, etc.,
- define helpful landmarks for page structure,
- define dynamically-updated 'live regions,'
- improve keyboard accessibility and interactivity,
- and much more."^[13]

For more information on how to properly use ARIA labels and apply them to data visualization, see Chapter 5 of this volume.

Testing

So far, I have mentioned and linked to several useful testing tools, specifically for colorblindness and contrast testing. When designing a style guide, all

palettes and templates should be pretested using these tools so people don't need to redo that work every time they make a chart. But another type of testing, user testing, is the best step an organization can take to make more effective and accessible charts. The more inclusive that user testing is, the better: user testing with people with disabilities is a best practice for building accessible charts. However, ensure a basic level of accessibility is in place before engaging in user testing—don't present a product that is completely nonfunctional on screen readers to your screen-reader testers and expect them to provide meaningful feedback.

By getting the charts in front of the intended users before publication, you can see which aesthetics and mappings are confusing, which labels are ineffective, and which interactions are lost. More testing is better, but incorporating people with disabilities into the building and testing process is best. Watching a user with a screen reader try (and probably fail) to paginate across elements in your online, interactive chart will tell you a lot about what you need to be doing better. Working with designers who are colorblind, are vision impaired, or use assistive technology will help ensure that you build the most accessible products you can. Many organizations and consultancies, such as Fable or Knowability,^[14] are equipped to help with this work.

Unfortunately, not all projects have the scope and budget to hire experts or pay user testers. But you shouldn't give up on user testing. Doing some testing is better than doing no testing, and the former can be as easy as asking a coworker or friend to give your product a try. This process, having a handful of users test something for a limited amount of time, is sometimes called "guerilla testing."^[15]

Adoption

A style guide is not useful if no one at the organization uses it. People have to know about it and find benefit from using it. If the style guide is seen as a burden and its value isn't clear, it will be ignored and forgotten.

Involving people from several parts of the organization in the creation of a style guide can increase its adoption. If people are involved in building it, they will feel some ownership over it and champion it to their departments. You will also get a larger variety of perspectives and users, which will make the guide more robust and inclusive. I recommend beginning the data visualization style guide process with a series of informational interviews with chart creators and consumers throughout the organization to identify the organizational needs the style guide will serve and to engage people across the organization in crafting a product that makes their jobs easier.

If the style guide is followed by much of the organization, the charts and graphs created using it should be more accessible. Templates based on the best practices from the guide and available for several different applications the organization uses can help improve efficiency and adoption.

Training

Training can also increase style guide adoption, and it involves a two-pronged approach that reaches both new and existing employees. When new employees join the organization, they should be trained on using the style guide and templates. If they learn to use these resources when they start, they are much more likely to do so consistently. From the beginning, an organization should emphasize accessibility and why it is important.

However, onboarding is often overwhelming, and the company and style guide will evolve over time.^[16] Offering ongoing advanced or refresher training on best practices allows people to stay current on the style guide or catch information at times when they actually need it. Ongoing trainings also allow current employees to stay in the know and to ask questions about how to apply the guide to specific use cases. The guide needs to work for them and provide a conversation rather than a mandate.

Persuade and Adapt

Some people will proactively seek out ways to use a data visualization style guide. These people should be rewarded and highlighted as good examples.

It is also helpful to look at where charts are not working or why creators are not using the templates and style guide. Maybe they don't know about them or how to use them. Or maybe the guide doesn't work for them, which would suggest a need to tailor or adapt its recommendations to fit their needs. Such cases present an opportunity to expand the guide and define a new use case to increase adoption.

Culture of Data Viz and Accessibility

Creating a culture of talking about data visualization and accessibility (together and separately) will increase awareness and knowledge in the organization.

Creating conversations about current accessibility practices will increase awareness and bring it to mind as a relevant topic for content creators. Sharing good examples from inside and outside the organization can further these conversations.

Both the accessibility and data visualization communities are extremely active, with speakers on this topic at events and conferences. The Data Visualization Society has an annual conference, *Outlier*, which includes talks on accessibility. The society also has a calendar of events that includes discussions of accessibility. Deque, a software accessibility company, hosts an annual conference, *Axe-Con*, which includes talks about data visualization. Creating a culture where data visualization and accessibility are topics discussed promotes continuous learning and encourages people to keep up with current standards.

Style guides are a critical first step toward maintaining this culture of continuous learning. As data visualization professionals, it is imperative that our work be accessible for all people, and having consistent standards that include accessibility best practices can help us achieve those goals. Accessible charts tend to be better charts. They are easier to read for everyone—they have colors that stand out, communicate their meaning, and don't confuse readers. They are not too cluttered, the text is easily understood, and they don't place undue cognitive burden on the readers. They can be read in multiple ways, such as on a screen or with a screen reader.

When we design accessibly, we design for broad impact, so as many people as possible can engage with our work. An effective style guide helps us front-load the design work and lets us set standards for how we want to manage the trade-offs necessary to build the best, most accessible visualizations we can.

Chapter Nine Notes

- ^[1] Amy Cesal, “What Are Data Visualization Style Guidelines?” Nightingale Journal of the Data Visualization Society (Medium blog), July 10, 2019, <https://medium.com/nightingale/style-guidelines-92ebe166addc>.
- ^[2] Bang Wong, “Color Blindness,” Nature Methods 8, no. 6 (2011): 441. <https://www.nature.com/articles/nmeth.1618.pdf>.
- ^[3] “Web Content Accessibility Guidelines (WCAG) 2.1,” W3.org, accessed October 13, 2022, <https://www.w3.org/TR/WCAG21/>. See also “Understanding Success Criterion 1.4.11: Non-Text Contrast,” W3.org, accessed October 13, 2022, <https://www.w3.org/WAI/WCAG21/Understanding/non-text-contrast.html>.
- ^[4] “Contrast Checker,” WebAIM, accessed October 13, 2022, <https://webaim.org/resources/contrastchecker/>.
- ^[5] “Coblis —Color Blindness Simulator,” Colblindor, accessed October 13, 2022, <https://www.color-blindness.com/coblis-color-blindness-simulator/>.
- ^[6] Lisa Charlotte Muth, “An Alternative to Pink & Blue: Colors for Gender Data,” Datawrapper Blog, July 10, 2018, <https://blog.datawrapper.de/gendercolor/>.
- ^[7] Lisa Charlotte Muth, “A Detailed Guide to Colors in Data Vis Style Guides,” Datawrapper Blog, March 30, 2022, <https://blog.datawrapper.de/colors-for-data-vis-style-guides/>.
- ^[8] Plain Writing Act of 2010, Pub. L. 111 – 274, 124 Stat. 2861–63 (2010). <https://www.govinfo.gov/app/details/PLAW-111publ274/summary>.
- ^[9] Cole Nussbaumer Knaflic, “So What?” Storytelling with Data blog, March 23, 2017, <https://www.storytellingwithdata.com/blog/2017/3/22/so-what>.
- ^[10] “Alternative Text,” WebAIM, accessed October 13, 2022, <https://webaim.org/techniques/alttext/>.
- ^[11] Amy Cesal, “Writing Alt Text for Data Visualizations” Nightingale Journal of the Data Visualization Society (Medium blog), July 23, 2020, <https://medium.com/nightingale/writing-alt-text-for-data-visualization-2a218ef43f81>.
- ^[12] “Complex Images — Long Descriptions,” W3.org, accessed October 13, 2022, <https://www.w3.org/WAI/tutorials/images/complex/#long-descriptions>.
- ^[13] “Introduction to ARIA – Accessible Rich Internet Applications,” WebAIM, last updated June 30, 2020, <https://webaim.org/techniques/aria/>.
- ^[14] “Accessibility Testers,” Fable, accessed October 13, 2022, <https://makeitfable.com/testers/>; “Usability Testing with People with Disabilities,” Knowability, accessed October 13, 2022, <https://knowability.org/services/usability-testing>.
- ^[15] Guy Ligertwood, “Guerilla Testing: Hallway Usability Tests for UX,” Adobe XD Ideas, April 15, 2020, <https://xd.adobe.com/ideas/process/user-testing/hallway-usability-test-guerrilla-testing/>.
- ^[16] Ben Chartoff, “Urban Institute’s Data Visualization Style Guide: A Living Document,” Data@Urban (Urban Institute Medium blog), March 19, 2018, <https://urban-institute.medium.com/urban-institutes-data-visualization-style-guide-a-living-document-b90710702a9f>.

Nontechnical Barriers to Data Visualization Accessibility in Government



MELANIE MAZANEC

As a new software developer in city government, I was tasked with making the city’s public-facing dashboards accessible. Up to that point, all the technical documentation I had needed as an engineer had been easily discoverable in a web search. So I was flummoxed to find almost nothing about data visualization accessibility.

After weeks of reading blog posts, the “talk” sections of wikis, and comments on coding forums, I hesitantly implemented automated text labels and keyboard navigation patterns. I was never able to secure funding to conduct usability research, nor was I successful in any efforts to partner with local disability advocacy groups to formalize a government technology advisory council.

Although my work was experimental and incomplete, it helped me connect with expert practitioners from whom I have learned a great deal. Knowing how hard it had been for me to find resources, I proactively shared new accessible data visualization guidance that I learned about and volunteered implementation support to colleagues in government technology. I assumed that people drawn together by mission-driven work would be excited to make democracy more inclusive and that the only thing holding them back was a lack of information.

Unfortunately, I was wrong. Fellow tech workers consistently told me that accessibility was an edge case that only affected a few people, that it didn’t matter because dashboards weren’t public facing, that the client hadn’t asked us to work on accessible data visualization, or that we would address accessibility problems “after we are done building the product.” Even though I had the information readily available, that wasn’t enough to make people value, understand, or apply it.

In retrospect, I should have known better. Despite the many web accessibility resources and trainings available, most websites still have easily remedied problems.^[1] And although large employers like governments, banks, and universities have accessibility mandates, computer science programs and coding bootcamps consistently fail to teach accessibility. It’s no wonder that a developer fresh out of college rolled their eyes when I pointed out basic accessibility problems in a code review.

Making information available about data visualization accessibility is a crucial first step. Connecting a community of practitioners to build on one another's ideas is the next. To get government agencies to adopt new practices, we need to also recognize and address the nontechnical barriers that public servants and vendors face in implementation.

Why Work on Data Accessibility in Government?

Public infrastructure shapes our lives for better or for worse. The placement of an urban highway can cut off a neighborhood from economic opportunity. A road widened to facilitate the flow of traffic may preclude kids from walking safely to school. Conversely, a reliable bus system allows people to drive less, leading to less congestion and demand for parking. And a wide enough sidewalk with adequate room for both people and trees can alleviate urban heat while still making room for strollers and wheelchairs.

Public infrastructure is not an inevitable natural landscape, but the result of design choices made by people. Human beliefs and biases influence whose needs are prioritized and what kind of infrastructure receives funding or support. People in power design processes determining who can provide input or make decisions. Public meetings held during the day may exclude people with day jobs from showing support for city initiatives. Input processes that happen only in person could exclude people who are immunocompromised. And state laws that require cities notify homeowners, but not renters, of public meetings for proposed large-scale developments might amplify the voices of wealthier people.

Modern public infrastructure includes more than roads and buildings. Software has become an integral part of the built environment and makes our lifestyles possible. And just as urban designers shape wealth inequality, public safety, climate change, and mobility, software designers can create access or inequality in the digital world. Biases and power dynamics influence every aspect of technology, such as who may be encouraged to pursue an engineering career, which pitches receive funding, or which projects are prioritized within a company.

Design decisions that prevent people with disabilities from having equal access are ubiquitous in technology. The 2022 WebAIM Million report evaluated the top million websites using an automated test for compliance with the Web Content Accessibility Guidelines, or WCAG. Errors were detected on 96.8 percent of pages, with an average of 50.8 errors per page.^[2] And, as demonstrated in 2017 by the Gov.UK web team,^[3] automated tests can only flag the types of issues that are easiest for engineers to find and remedy, so there were likely many more errors present than those found. The prevalence of automatically detectable WCAG compliance errors indicates a much deeper problem with the usability of the web.

The visual representation of information in charts, graphs, and diagrams is one area where undetectable issues often arise: the use of red and green to distinguish categories in a stacked bar chart; a tooltip that is shown on mouse hover but is not operable by keyboard; design software that requires hardware keyboard commands to operate but is incompatible with on-screen keyboards; or maps that use images and textures to convey information not otherwise available to people who are visually impaired or blind. All of these issues can create inequitable experiences.

Although government is mandated to serve all people equitably, "the state is embedded in and enmeshed with civil society, not apart from or above it."^[4] The software industry's proliferation of visual tools with accessibility problems cannot be separated from its impact on government. As bureaucracy is digitized, data are generated by each process and transaction. Governments use tools built by private companies to collect, analyze, and visualize data. In turn, government data analysts help monitor service performance, inform policy changes, and share information with the public. Programmers and analysts generate reports and build dashboards for elected officials, agency leaders, city managers, and the general public. People whose access needs are not considered in data analysis and communication are excluded from these bureaucratic and democratic processes.

Public administration scholars coined the term “administrative discretion” to describe the way public workers unofficially shape policy through implementation decisions, which can even be weaponized to advance political agendas.^[5] Beyond elected officials and public workers, others who wield power in government include nonprofits that deliver public services or receive grant funding, and private companies contracted to work on behalf of government. It is necessary, then, to ensure these means of exercising political power beyond elected representation are accessible too. Work toward accessible data visualizations in government should not be limited to public-facing dashboards and reports; rather, it should include all tools used in governments and organizations that work with governments.

For government to represent all the people it serves, it must make the information and systems that facilitate participation in governance available to everyone. And for a government to generate inclusive public infrastructure, it must also use accessible digital infrastructure to support its operations and decisionmaking processes.

Implementation Barriers

Although official guidance is lacking, many informational resources created by data visualization accessibility practitioners have begun to form a body of knowledge that can support anyone looking for technical advice. However, building for accessibility necessitates resources in the form of not just information but also support from colleagues, public leaders, and private-sector partners.

Ambiguity and a Lack of Standards for Accessibility

The current version of WCAG covers some data visualization topics, including color, keyboard interactivity, animation, and text alternatives to images. But the application of WCAG guidance is ambiguous when it comes to even simple interactive

charts, let alone complex collaborative design interfaces. In cases where the visual is a static image secondary to the rest of the content, a text alternative might be enough to offer a comparable experience (see Chapters 4 and 8 in this volume for some guidance). However, the proliferation of interactive dashboards, maps, and other tools with visual interfaces make it both necessary and possible to find novel ways to build inclusively.

Thanks to a robust community of thoughtful practitioners, finding examples and advice is now easier than ever. Individuals and organizations are publishing code snippets, essays, and documentation to help others build accessible data visualization. The Data Visualization Society incorporated accessibility as a category on which a visualization can be judged in their design contests. But the lack of formal standards still holds back progress on data visualization accessibility in government.

First, the ambiguity of WCAG’s applicability to data visualization leaves room for interpretation in an environment where experimentation is frowned upon. At the federal level, where Section 508 (see the Section 508 box on the next page) incorporates WCAG by reference, should only keyboard-navigable dashboard tools be used, or are text alternatives acceptable? How should a topographical map, a Sankey diagram, or a scatterplot be described in text? In local government, where WCAG is not mandated but the less-specific Americans with Disabilities Act still applies, to what standards should vendors be held? In the context of government, where the quality of technology is measured more by its track record of stability than by its future potential, workers and vendors are likely to err on the conservative side and follow precedent. For data visualization, this reticence may mean providing a text alternative rather than trying to follow new, informal data visualization advice.

Section 508

Section 508 of the Rehabilitation Act of 1973 (later amended by additional laws) is one of several federal disability laws. Section 508 requires federal agencies to “develop, procure, maintain, and use information and communications technology (ICT) that is accessible to people with disabilities— regardless of whether or not they work for the federal government.”^[6] This requirement extends to websites, user guides for software and tools, online training, webinars and teleconferencing, PDF documents, and more.

The requirements of Section 508 have been updated several times to better align with modern technology and international standards. Today, Section 508 incorporates WCAG 2.0 by reference. Although these guidelines do not apply to private-sector businesses, the federal government must ensure that it procures only 508-compliant software from the private sector.

Second, it takes more resources to start from scratch than it does to start with standards. Even when starting from blog posts, research papers, and YouTube videos about a topic, an individual visualization creator must put in hours of effort just to learn enough to discern which advice to follow. In the absence of clear consensus, they may end up reinventing the wheel and creating their own accessibility patterns, even though established patterns have already begun to emerge across different visualization tools. Although following rules is never a replacement for usability research, building on the experience of others can allow developers to focus government’s exceedingly scarce resources where they can make a larger impact.

Lastly, standards allow governments to convey accessibility expectations to vendors. WCAG compliance is a convenient shorthand to include in agreements with software companies. It is easy to look for on a website, ask about by email, or write into a contract. In the absence of standards to reference,

a government might ask for accessibility in such a vague way that any agreement is unenforceable. Or requesting accessibility without referencing standards may require so much prior knowledge of technical implementation that doing so is not feasible for nontechnical government workers.

Referencing WCAG standards in Section 508 for the federal government did not magically change culture, funding, or entrenched legacy software overnight. The (gargantuan, unfinished) project of improving federal web accessibility has taken this mandate, as well as a corps of digital services professionals motivated by a shared dream of high-quality public services, and a generation of disability rights activists holding government accountable. Neither can we expect data visualization standards to accomplish any miracles. But working a clearer mandate into existing digital accessibility guidelines, especially for simpler cases, would give public employees a place to start.

Limited Resources Are Subject to Power Dynamics and Diffusion of Responsibility

Even if clear standards for data visualization existed and were mandated by law, enforcing them would still be a separate matter. Whose responsibility is it to ensure that government data visualizations are accessible, and what support might they have?

Analysts

An individual government worker who wants to make their data visualization work accessible might face limitations. Analysts may not have the training or funding to learn how to create accessible data visualizations. Most government software users, including creators of documents, presentations, PDFs, web content editors, and data visualizations, are not trained in accessibility.

Even those who are experts in data visualization accessibility may still be stymied by the limitations of the tools they are using. Most statistical software limits control over data presentation to minor choices like graph type, color, and axis labels (see, e.g., Chapter 5). If an analyst has the freedom to choose different

software, they may not have the time or capability to evaluate the comparative accessibility of different tools. And many analysts, especially in government, are restricted in their choice of tools to whatever their predecessors chose (or whatever the organization procured) long ago.

Analysts are also limited in their choice of tools by what is available from the private sector. Custom software is so time consuming and expensive that it is usually only justifiable for a highly specialized product that will serve many users, a product that will need to be changed frequently, or small projects. For most products that are commercially available, including software for word processing, email, spreadsheets, and statistics, a government or agency could never hire enough engineers to come close to Google or Microsoft's economy of scale. Instead, governments use the software products that technology companies choose to make. If private-sector companies do not make their products accessible, government workers who use these tools in turn are restricted in their accessibility efforts.

Engineers

Although most data visualizations in government are generated by statistical software procured from the private sector, some are custom made by front-end software engineers. Because engineers have a greater degree of control over the appearance and functionality of their visualizations, they can more easily implement accessibility recommendations. But having a greater degree of freedom in technical choices does not always mean having more control over one's work environment.

Engineers whose tasks are set by a project manager may have very little say in prioritization. An engineer with autonomy in choosing day-to-day tasks may still be stuck in a "work order" paradigm within a siloed IT department, where they receive full project plans and deadlines from another department with no opportunity to collaboratively plan for what kind of work a project requires. Finding support to learn about accessibility, write patterns, and conduct usability research can be hard if the people in power

have little understanding or patience for the behind-the-scenes work necessary to maintain high-quality technical products. These limitations exist in and outside of government.

Many software companies employ user researchers and designers to understand what end users need, communicate these needs to engineers, and conduct usability testing. Design and research are crucial for building software that is not just compliant with accessibility and business requirements but also usable. Unfortunately, design and research roles are still uncommon on government teams. Instead, many government systems are "designed" by an engineer who interprets business requirements from an agency director. The system may be evaluated by domain experts in government but never tested with the people who will have to use the system. An engineer in such an environment might not have the skills or permission to engage members of the public in usability research on their own. The result is systems that make sense only to the people who build them.

Managers

Individual public workers may not have the authority to prioritize data visualization accessibility or have access to the tools or resources needed to create more accessible work. In these cases, people who hold positions of power may clear the path.

Because most software is purchased and not built in-house, the single most effective way to work on data visualization in government is to procure accessible products. For off-the-shelf products that are not specific to government, accessibility may be a last priority, because private companies are not held to any specific standard for digital accessibility. For specialized needs where a company might only serve government clients (e.g., a maker of permitting software), governments can only choose from few alternatives. The same can be said of domains where a few large companies dominate, like cloud infrastructure or spreadsheets. Finding an accessible product can be challenging or impossible.

One might assume that custom software would be better because it is tailored to meet the client's needs. But building custom software comes with the huge risk of sinking large amounts of money, which cannot be recovered, into projects that are either never delivered or poorly delivered. And regardless of how many options exist or how bad a custom system is, the cost and time commitment required to switch software systems is prohibitively high. A government may be stuck with a newly procured product riddled with accessibility problems for years until they can afford to replace it. Vendors of off-the-shelf and custom software hold a lot of power because of these transaction costs associated with software procurement.

Vendors also hold power in relationships with governments because of information asymmetry. Many public managers do not have the knowledge or expertise to demand or enforce digital accessibility. The silos and red tape of bureaucracy may keep them from collaborating with engineers and analysts to develop specific, actionable requirements for contracts.

Governments can exercise the power they do have. A well-intentioned vendor may worry that data visualization accessibility or usability research for accessibility are beyond the scope of a contract, especially because data visualization accessibility requirements are not spelled out in WCAG, Section 508, or the Americans with Disabilities Act. To address some of these barriers, governments can specify data visualization accessibility requirements in requests for proposals and contracts in addition to WCAG compliance, allocate time and funding in custom software projects for accessibility usability research, and ask vendors about how they work on accessibility and whether internal tools and data visualization are included.

Bias about Who Works, Holds Power, and Makes Decisions

In spring 2022, I interviewed for a front-end development job with the cofounder of a

cybersecurity startup. He told me that they had many federal agencies as clients. Because I had previously worked on dashboards for a cybersecurity startup, I understood that most of the data visualization work would be internally facing, to be used primarily by IT workers in government and managers to monitor and respond to cybersecurity incidents. I was still excited: using accessible internal tools in government can make it easier to hire people with disabilities into positions of power and thereby build a representative bureaucracy.

When it came time for me to ask questions, I asked how they approach building accessible dashboards. He said, "We don't really worry about accessibility. Section 508 doesn't apply to cybersecurity." He seemed shocked when I told him I would not continue the interview process because I was disappointed that they knowingly exclude people with disabilities from federal employment. I was shocked that he was shocked. What made him assume that no person with a disability would ever hold a job in government and need to use his product?

This cofounder is not alone in his beliefs that accessibility is only relevant for public-facing projects and does not extend to internal tools. Most government technology companies do not evaluate internal tools for accessibility, even if they have a social impact mission and push for inclusivity in the products they build for others. These companies assume people with disabilities might be the recipients of services, but not colleagues or decisionmakers who will need to fully participate in design processes, incident response, or project planning.

These beliefs indicate unexamined ableist assumptions about who can or should hold power. When people are separated into those "doing" and those "receiving," existing disparities in representation in government create a feedback loop of exclusion. Workers in environments with accessibility barriers believe they have no colleagues with disabilities and choose internal tools with accessibility problems. Internal tools become crucial parts of jobs and public processes. People with disabilities are prevented from working

or participating in government and advocating for themselves from the inside.

Conclusion

Cultural and institutional factors can make it hard to implement accessible data visualization practices in government. But an individual can still strive, within the context of their role and skill set, to influence their organization for the better.

It is no accident that tech industry tools and ways of working are being imported into government. In the past decade, several organizations and fellowships (e.g., US Digital Service, US Digital Response, Office 18F in the General Services Administration, the Presidential Innovation Fellowship, Tech Congress, Code for America, and the Tech Talent Project) have sprung up to recruit tech workers from Silicon Valley to bring their experience to bear on our most challenging public problems. Amazon, Microsoft, Google, GitHub, and other tech companies have eagerly onboarded customers and launched initiatives to digitize and modernize public services. Universities are scrambling to rework their master of public administration programs to include more digital governance skills.

Many of these efforts have created more reliable digital services, cleaner interfaces, and smoother interactions with governments. But technological change, if not led by the public service values of equity, democracy, participation, representation, and transparency, can create a tangled knot of inaccessible infrastructure that cannot be unbuilt. Unchecked proliferation of inaccessible external and internal visual tools in government decisionmaking processes will perpetuate a cycle of exclusion.

Public leaders, workers, and communities can mitigate this potentially negative impact. For large procurements, explicitly naming accessibility as a requirement for not only public-facing projects but also internal tools can go a long way. Vendors building custom software should also be required to meet a standard of accessibility that goes beyond mere compliance, toward usability and inclusion. To handle smaller-value software acquisitions under the procurement threshold where an individual public servant might exercise discretion, governments should evaluate and recommend accessible products in advance for common needs.

But procuring accessible software will not be enough. Even tools built to be accessible can create documents, dashboards, and other work products with accessibility problems. Governments also need to create accessibility policies and provide internal, job-specific training. Accessibility is the responsibility of everyone in an organization who uses digital tools.

The disability rights movement has been clear from the start that people with disabilities should lead the way: “nothing about us without us” has been a common mantra since the 1990s. Building inclusive public institutions that serve everyone equitably requires that our technical tools support people with disabilities not just as end users, but as political leaders, decisionmakers, participants in public discourse, and public workers in a representative bureaucracy.

Chapter Nine Notes

[1] “The WebAIM Million Report,” WebAIM, last updated March 31, 2022, <https://webaim.org/projects/million/>.

[2] “The WebAIM Million Report,” WebAIM, last updated March 31, 2022, <https://webaim.org/projects/million/>.

[3] Mehmet Duran, “What We Found When We Tested Tools on the World’s Least-Accessible Webpage,” Accessibility in Government blog, Gov.UK, February 24, 2017, <https://accessibility.blog.gov.uk/2017/02/24/what-we-found-when-we-tested-tools-on-the-worlds-least-accessible-webpage/>.

[4] Kelly Campbell Rawlings and Thomas Catlaw, “Democracy As a Way of Life,” in *Government Is Us 2.0*, ed. Cheryl Simrell King (New York: Routledge, 2011), 31.

[5] Pamela Herd and Donald P. Moynihan, *Administrative Burden: Policymaking by Other Means* (New York: Russell Sage Foundation, 2019).

[6] “What Is Section 508?” EPA.gov, last updated January 18, 2022, <https://www.epa.gov/accessibility/what-section-508>.

Appendix: Accessibility Tools and Resources

This technical appendix provides curated lists of screen readers, color tools, accessibility remediation firms and platforms, and other resources for accessibility guidelines. These lists are not comprehensive, and they are likely to change as more tools are developed and published. The lists do not constitute an endorsement of any particular tool or service. We hope these lists can help readers explore the tools available that will enable them to create better, accessible data and data visualization products.

Accessibility Guidelines

Chartability. Developed by Frank Elavsky (see Chapter 2), Chartability is a set of testable questions that seek to ensure data visualizations, systems, and interfaces are accessible. Chartability is organized into principles with testable criteria and focused on creating an inclusive data experience for people with disabilities. <https://chartability.fizz.studio/>

Section508.gov. Commonly referred to as simply “508 compliance,” Section 508 of the Rehabilitation Act of 1973 (29 U.S.C. § 794d), and amended by the Workforce Investment Act of 1998 (P.L. 105-220), requires federal agencies to develop, procure, maintain, and use information and communications technology that is accessible to people with disabilities. The US General Services Administration is the federal agency responsible for providing technical assistance to other agencies to ensure Section 508 compliance requirements are correctly put in place. <https://www.section508.gov/>

US Environmental Protection Agency. The EPA provides a succinct introduction to the guidelines and requirements set forth under Section 508. <https://www.epa.gov/accessibility/what-section-508>

Web Content Accessibility Guidelines (WCAG). The Web Accessibility Initiative hosts the WCAG guidelines, which explain how to make web content more accessible for people with disabilities. <https://www.w3.org/WAI/standards-guidelines/wcag/>

Screen Readers

JAWS. The Job Access With Speech (JAWS) screen reader provides speech and Braille output for many major computer programs including Microsoft Office, Google Docs, Chrome, Firefox, Edge, and more. JAWS only works on Windows computers. A single license for an individual user is currently \$95 a year. <https://support.freedomscientific.com/Products/Blindness/JAWS>

Narrator. Narrator is a screen reader tool built into the Windows 11 operating system. It has some limitations with some browser tools and web applications. <https://support.microsoft.com/en-us/windows/complete-guide-to-narrator-e4397a0d-ef4f-b386-d8ae-c172f109bdb1>

NVDA. NonVisual Desktop Access (NVDA) is a free and open-source screen reader that supports major software programs and is often compared with JAWS. As with JAWS, NVDA works only on computers running the Windows operating system. <https://www.nvaccess.org/download/>

VoiceOver on iOS. This screen reader is built into the iOS operating system on iPhones and iPads. <https://support.apple.com/guide/iphone/turn-on-and-practice-voiceover-iph3e2e415f/ios>

VoiceOver on macOS. This screen reader is built into the Mac operating system. It has a steeper learning curve than some of the other tools on this list, but it also has more than 30 languages available. https://www.apple.com/voiceover/info/guide/_1121.html

TalkBack. The internal accessibility tool for Android users, the TalkBack screen reader speaks text and image content on the screen. https://support.google.com/accessibility/android/topic/3529932?hl=en&ref_topic=9078845

Emacspeak The Complete Audio Desktop. This is a free screen reader tool that works in the emacs editor on Linux and OS X operating systems. Support is provided for interacting for many types of Internet media and is used by programmers and researchers. <https://github.com/tvraman/emacspeak>

Color Checkers

Accessibility Scanner. Accessibility Scanner is an Android app that suggests improvements such as enlarging small touch targets, increasing contrast, and providing content descriptions. <https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor&hl=en>

Adobe Color Contrast Checker. Associated with Adobe's Color tool (<https://color.adobe.com/create/color-wheel>), this contrast checker lets users input and adjust foreground and background colors and see contrast ratio scores per the Web Content Accessibility Guidelines. <https://color.adobe.com/create/color-contrast-analyzer>

Color Contrast Analyzer. A downloadable tool (available for Windows PCs and Macs) that uses the Web Content Accessibility Guidelines and a color vision deficiency simulator to check color contrast.

Color Contrast App. A mobile accessibility checker designed for the iPad (iOS) that lets you test colors of apps, websites, or screenshots using an eyedropper tool. <https://apps.apple.com/na/app/color-contrast/id1095478187>

Color Contrast Checker. Color Contrast Checker is a free tool that can be used to test the contrast on your foreground and background colors. The user can type in hexadecimal color codes and the tool will generate a color contrast "score." <https://marijohannessen.github.io/color-contrast-checker/>

Color Oracle. Color Oracle is a free color blindness simulator for Windows, Macs, and Linux operating systems. It is a lightweight application that can show the user, in real time, what people with common color vision impairments will see. <http://colororacle.org/>

Contraste. A downloadable program for Mac computers only that enables the user to know if a combination of colors, for a text and a background, passes accessibility thresholds defined by the Web Content Accessibility Guidelines. <https://contrastapp.com/>

Leonardo. A relatively new project from Adobe for creating, managing, and sharing accessible color systems for user interface design and data visualization. Leonardo also has an open application programming interface that enables users to use colors in their development environment. <https://leonardocolor.io/#>

Material.io. This tool lets users create, share, and apply color palettes, as well as measure the accessibility level of any color combination. <https://material.io/resources/color/>

Monsido Contrast Checker. A basic contrast checking tool with multiple possible input methods and no download necessary. <https://monsido.com/tools/contrast-checker>

Sim Daltonism. Available through the iTunes store for Mac computers, this tool lets you visualize colors as they are perceived with various types of color blindness. <https://itunes.apple.com/us/app/sim-daltonism/id693112260>

Tanaguru Contrast Finder. RGB and hexadecimal color input models with options to adjust the contrast ratio. The tool doesn't have a full interactive color wheel like some others, but it gives suggestions on how to create a consistent palette. <https://contrast-finder.tanaguru.com/>

Vischeck. Vischeck simulates colorblind vision and shows the user what things look like to someone with color vision deficiency. <http://www.vischeck.com/>

WebAIM. A simple color contrast checker that requires the user to input hexadecimal colors for foreground and background objects, then returns a pass/fail score based on WCAG guidelines. <https://webaim.org/resources/contrastchecker/>

Accessibility Testing Tools and Companies

Chax Training & Consulting: <https://AccessibilityUnraveled.com>

Deque: <https://www.deque.com/>

Fable: <https://makeitfable.com/>

Intopia: <https://intopia.digital>

Knowability: <https://knowability.org/>

Tetralogical: <https://tetralogical.com/>

MEET THE AUTHORS



DAX CASTRO



AMY CESAL



FRANK ELAVSKY



ALICE FENG

DAX CASTRO

Dax Castro is an Adobe-certified PDF accessibility trainer. He is also an Accessible Document Specialist with more than 25 years of experience creating presentations for corporate communications. Castro was a contributing author for the Accessible Document Specialist certification through the International Association of Accessibility Professionals, he cohosts an accessibility podcast, and runs a document accessibility support group.

AMY CESAL

Amy Cesal is a data visualization designer and instructor and the senior design director of data visualization at Morning Consult. She specializes in working with subject matter experts to design accessible and legible visualizations of complex data across domains. She is also a leader in the use of data visualization style guidelines, writing and speaking on the topic. Cesal cofounded the Data Visualization Society, where she serves as an advisory committee member. Cesal's innovative and unusual data visualization work has garnered her three Information is Beautiful awards. She holds an MS in information visualization from the Maryland Institute College of Art, where she is the subject matter expert for the Data Analytics and Visualization program. Find her at <https://www.amycesal.com/>.

FRANK ELAVSKY

Frank Elavsky is a design engineer turned researcher who is currently pursuing a PhD studying the intersection of data interaction and accessibility at Carnegie Mellon University's Human-Computer Interaction Institute. He is also the author of *Chartability*, a set of heuristics for evaluating the accessibility of data experiences. Before pursuing his PhD, Elavsky was a lead contributor to *Visa Chart Components*, a design system component library of charts and graphs built with a focus on accessibility.

ALICE FENG

Alice Feng is a data visualization developer based in the Washington, DC, area. She is passionate about using design to make data and information more accessible to broader audiences and recently has explored ways to bring more diversity, equity, and inclusion into how we visualize data. Her work has appeared in the *Parametric Press* and the *Pudding*. Previously, Alice worked as a data visualization developer at the Urban Institute, where she built interactive and static data-based features and tools to communicate public policy research. Alice is currently embarking on a new adventure at Natera. She is on Twitter at @fleecealeece.



SARAH FOSSHEIM

SARAH FOSSHEIM

Sarah Fossheim is a multidisciplinary developer, designer, and accessibility specialist. They have a strong focus on data visualization accessibility and usability, having worked as a product developer and designer on data-heavy products in the cancer research and educational sector. Currently, Sarah is an independent consultant, educator, and advisor who helps companies create more accessible and inclusive solutions.



ELIZABETH HARE

ELIZABETH HARE

Liz Hare is a quantitative geneticist researching working on dog behavior and health. She works with breeding programs to implement data-driven selective breeding that can improve canine welfare, health, and work-related behavior. As a blind member of the R statistical programming community, she advocates for broadly inclusive conferences and training as well as for accessible software, documentation, and reporting that includes meaningful alt text for data visualization. She holds a PhD in genetics from the George Washington University.



LARENE LE GASSICK

LARENE LE GASSICK

Larene Le Gassick is the head of technology at Intopia, a Microsoft MVP in Developer Technologies focusing on accessibility, and community manager of Tech Leading Ladies Australia. She specializes in building full-stack applications with accessibility and inclusion at front of mind. She is also an international conference speaker and guest university lecturer on digital accessibility and software architecture. Le Gassick is one of the core members of the DatavizA11y group, where she and other accessibility advocates and experts come together to raise awareness of and provide guidelines and solutions for a lack of resources in making data visualization inclusive of people with various disabilities. When she's not building software or talking about accessibility and inclusion, Le Gassick spends her time cycling or scrolling through Twitter.



MELANIE MAZANEC

MELANIE MAZANEC

Melanie Mazanec is a public interest technologist located in Philadelphia. Most recently, she worked on knowledge management and best practices for Code for America as the associate director of Human-Centered Government. Mazanec has also volunteered in the Code for America network and worked as a software engineer for public-interest tech consultancy Bloom Works Digital; the City of Asheville, North Carolina; and a cybersecurity startup. Mazanec is most proud of her work for many projects to help make data and data visualizations transparent and open source. Mazanec worked in music and education before getting into tech. When she's not busy emailing government software vendors to inquire about their accessibility roadmaps, she takes thousand-mile bicycle adventures and plays the trumpet.



SUE POPKIN



DOUG SCHEPERS

SUE POPKIN

Susan J. Popkin is an institute fellow in the Metropolitan Housing and Communities Policy Center at the Urban Institute and program director for Urban's Disability Equity Policy Initiative. A nationally recognized expert on public and assisted housing programs and policy, Popkin also leads Urban's Future of Public Housing work and is currently the co-principal investigator for the evaluation of the US Department of Housing and Urban Development's Rental Assistance Demonstration. Popkin has led many mixed-methods studies of the impact of housing programs on resident outcomes, including Chicago's Plan for Transformation; HOPE VI; and Urban's Housing Opportunities and Services Together demonstration, which uses community engagement and community-based participatory approaches to explore new strategies for improving outcomes for families in public and assisted housing. Popkin is the author or coauthor of multiple books, including *No Simple Solutions: Transforming Public Housing in Chicago*, *Moving to Opportunity: The Story of an American Experiment to Fight Ghetto Poverty*, and *The Hidden War: Crime and the Tragedy of Public Housing in Chicago*. Popkin holds a PhD in human development and social policy from Northwestern University.

DOUG SCHEPERS

Doug Schepers is the founder and director of Fizz Studio, a startup focusing on making data visualizations accessible to people with disabilities. While he was working as a project manager for the World Wide Web Consortium, organizing and authoring web standards for over a decade, Schepers recognized a need for accessible data visualization, so he dedicated his time to consulting, software development, and research to make access to data universal and equitable. Schepers is widely acknowledged as a leader in data visualization accessibility and is a popular speaker at conferences, workshops, and seminars. He uses his adult-diagnosed ADHD as a tool to help synthesize ideas from across different domains and as an intuitive gauge for cognitive load.



JONATHAN
SCHWABISH



LÉONIE WATSON

JONATHAN SCHWABISH

Jonathan Schwabish is a senior fellow in Income and Benefits Policy Center at the Urban Institute, where he studies disability insurance, retirement security, and nutrition policy; he is also a member of Urban's communications team, where he specializes in data visualization and presentation design. Schwabish is a leading voice for clarity and accessibility in research and has written on how to best visualize data, including on technical aspects of creation, best practices in design, and how to communicate social science research in more accessible ways. Through his work, Schwabish helps nonprofits, research institutions, and governments at all levels improve how they communicate their research and findings to partners, constituents, and citizens. He teaches data visualization and presentation skills at Georgetown University and at American University, and he founded PolicyViz, a consulting firm that helps clients improve how they communicate data and analysis.

LÉONIE WATSON

Léonie Watson is the director of TetraLogical, a member of the World Wide Web Consortium Advisory Board, co-chair of the World Wide Web Consortium Web Applications Working Group, and a member of the BIMA Inclusive Design Council. Watson also is co-organizer of the Inclusive Design 24 conference, coauthor of the Inclusive Design Principles, and a mentor to young people interested in the fields of accessibility and inclusive design. Watson is often found at conferences talking about web standards or accessibility mechanics and pushing the boundaries of inclusive design. She has also been published in Smashing magazine, SitePoint.com, and Net magazine, as well as on her own site, tink.uk. In her spare time, Watson likes reading, cooking, drinking tequila, and dancing (although not necessarily in that order)!

ADVISORY BOARD



AMANDA BOTTICELLO



TREVOR
DYSON-HUDSON



DOMINIK MORITZ

AMANDA BOTTICELLO

Amanda Botticello is the assistant director of the Center for Spinal Cord Injury Research and the Center for Outcomes and Assessment Research at the Kessler Foundation in West Orange, New Jersey. Botticello has a faculty appointment in the Department of Physical Medicine and Rehabilitation at Rutgers New Jersey Medical School, where she is research associate professor and vice chair of research education. She conducts research on the social determinants of health and disability, focusing on environmental factors and addressing barriers to community reintegration for adults with spinal cord injury, children with special health care needs, and other populations with chronic health conditions. Botticello trained in social epidemiology at the Fielding School of Public Health at the University of California, Los Angeles, where she received her MPH and PhD. She holds a BA in psychology from Amherst College.

TREVOR DYSON-HUDSON

Trevor Dyson-Hudson is the director of the Center for Spinal Cord Injury Research and the Center for Outcomes and Assessment Research at the Kessler Foundation in West Orange, New Jersey. Dyson-Hudson is also a research associate professor in the Department of Physical Medicine and Rehabilitation at Rutgers New Jersey Medical School. His research interests include the prevention and treatment of common secondary medical complications affecting people with spinal cord injuries. Dyson-Hudson has benefited from his lived experiences with spinal cord injury since 1992. He received his BA in physiology and cell biology from the University of California, Santa Barbara, and his MD from the Albert Einstein College of Medicine. Dyson-Hudson also completed a research fellowship in rehabilitation research and complementary and alternative medicine at Rutgers New Jersey Medical School.

DOMINIK MORITZ

Dominik Moritz is on the faculty at Carnegie Mellon University and a researcher at Apple. At Carnegie Mellon, he coleads the Data Interaction Group at the Human-Computer Interaction Institute. His group develops interactive systems that empower everyone to effectively analyze and communicate data. Moritz's systems (Vega-Lite, Falcon, Draco, Voyager, and others) have won awards and are widely used in industry and by the Python and JavaScript data science communities. Moritz received his PhD from the Paul G. Allen School at the University of Washington, where he was advised by Jeff Heer and Bill Howe.



SUSAN ROBINSON

SUSAN ROBINSON

Susan Robinson masterfully blends over 25 years of multisector leadership with her experiences being legally blind to shift thinking, elevate potential, and inspire action regarding accessibility. In her TED Talk, “How I Fail at Being Disabled,” she flips generally accepted ideas upside down to uncover, challenge, and redefine preconceived obstacles. In her role as an advisor and consultant, Robinson hosts leadership workshops as well as business strategy and transformation planning. She received her MPA in health policy and management from New York University’s Robert F. Wagner Graduate School of Public Service and her BS in health policy and administration from Pennsylvania State University. In her free time, Robinson is a tango dancer, yoga practitioner, tandem cyclist, and world traveler.

