

Web Accessibility

WCAG 2.2 made easy

2025 EDITION

Olga Revilla Muñoz
Olga Carreras Montoto

Prologue
Emmanuelle Gutiérrez y Restrepo

Web Accessibility

WCAG 2.2 made easy

Olga Revilla Muñoz

Olga Carreras Montoto

Prologue by Emmanuelle Gutiérrez y Restrepo



Itákora Press

Web Accessibility. WCAG 2.2 made easy.

Authors: Olga Revilla Muñoz and Olga Carreras Montoto

Copyright © 2025 Itákora Press. All rights reserved.

Printed by Amazon.

Edition: April 2025

ISBN: 9798316189311 · Depósito Legal : M-9105-2025

This book summarizes and cites content from the *Web Content Accessibility Guidelines 2.2*.

Copyright © [2023] World Wide Web Consortium (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University).

Editor: Olga Revilla Muñoz – Itákora.

Executive Editor: Olga C. Santos Martín.

Editorial design: Ana Matellanes García.

Prologue: Emmanuelle Gutiérrez y Restrepo.

PDF accessibility: Olga Carreras Montoto.

Images:

- Geralt'S cover mosaic on Pixabay.
- Man in the metro of Andrea Piacquadio on Pexels.
- Man Driving by Dan Gold on Unsplash.
- Personas by thisuserdoesnotexist.com.
- "Idea" by Ralf Schmitzer, "Stop" by Márcio Duarte, "Pointer gestures" by Jeff Portaro, and "Envelope with Arrow" by Andrey Vasiliev; all of them from *The Noun Project*. "Finger Drag Sliders" by Freepik

Coffee maker logo: Olga Revilla Muñoz

Web Accessibility. WCAG 2.2 made easy. / Revilla Muñoz, Olga; Carreras Montoto, Olga —Madrid: Itákora Press, 2025

All rights reserved. No part of this book may be reproduced or transmitted by any means or in any form (electronic, mechanical, photocopying, recording, or otherwise) without the publisher's prior written permission. To request information about reproduction or transmission rights, please contact info@itakora.com

Disclaimer:

This book aims to analyze and explain Web Accessibility topics for informational and educational purposes. However, the information provided does not constitute legal, technical, or professional advice.

The topics are subject to interpretation and periodic updates. Therefore, the content of this book may become outdated, be interpreted differently, or contain unintentional errors. The authors do not guarantee the information's accuracy, completeness, or applicability in all circumstances and disclaim any liability arising from the use or misuse of this material.

Readers are responsible for verifying any information before making decisions based on it and are encouraged to consult official sources or experts in the field for specific guidance.

If you find any mistake, please notify it to info@itakora.com to improve future editions.

*For those who consider compliance as the starting point,
and not just the goal.*

Olga Revilla

*For Jesús, Silvia, and Samuel,
on the other side of "Usable y accessible".*

Olga Carreras

Thank you 2.2

Olga Revilla
Itákora

The origins of this book date back to 2008, when the WCAG 2.0 guidelines were published. At that time, mobile phones were beginning to gain market share, thanks in particular to the iPhone, which had been launched a year earlier. Mobile phones and WCAG 2 were a significant improvement for people with disabilities, who gained two great allies in accessing information and digital functionalities.

Today, the trending topic is artificial intelligence: it describes images, generates captions for videos, rewrites complicated texts in plain language, detects the language, translates in real-time, develops apps just with natural text input... Again, it is a revolution for all those people with limited access to information who can enjoy the advantages of technology in their day-to-day lives.

WCAG 2.2 is already adapting to this technological revolution by revising one of its recommendations, as current technologies are capable of processing code, including invalid code. Will we see other recommendations disappear as technology continues to improve? It's pretty likely. In the meantime, the W3C will guide the path to make technologies more human.

This book is intended for everyone who wants to make the e-world a more accessible and better place. Its aim is not only to explain WCAG 2.2 but also to show its evolution and applicability to cross-cutting issues, such as organizational policies, design systems, web development, and plain language.

Thank you very much to my fellow partners: Olga Carreras for being a generous and inexhaustible source of knowledge; Emmanuelle Gutiérrez and Restrepo for her support at the Sidar Foundation; and Olga C. Santos for contributing her vision and academic expertise. Also, thanks to all the contributors of the W3C, who work tirelessly to make the web a place for everyone.

Finally, a big thank you to everyone who purchased our previous books and took the time to contact us to express their support. We hope that this book meets your expectations.

Preface

Olga Carreras
Usable y accesible

I want to thank you for the warm reception that the book "Web Accessibility: WCAG 2.1 made easy" received in 2018. This support has greatly motivated us to embark on this new edition. The current edition of the book not only expands its scope to address the new compliance criteria of WCAG 2.2, recommended since October 5, 2023, but is also offers as an updated edition—thoroughly revised and enriched with three new chapters: "Accessible Single Page Applications (SPA)", "Accessible Design Systems" and "Caring for words".

Our goal is to continue the dissemination and awareness efforts we started together seven years ago, with a commitment to making the web simple, accessible, and free for everyone.

Over these years, we have witnessed the publication of new standards, the approval of new laws, and the emergence of initiatives such as AccessibleEU. We are experiencing the technological revolution in artificial intelligence , and discussions about accessibility in virtual, augmented, and mixed-reality contexts are becoming more frequent.

You might expect that accessibility issues on websites and mobile apps would already be a thing of the past. However, I still encounter the same errors in today's accessibility audits as I did seven years ago: failure to meet many compliance criteria due to a lack of awareness or understanding; inaccessible documents; incorrect application—or misapplication— of the WAI-ARIA standard; and difficulties in managing and maintaining accessibility over time.

In my experience as a trainer, I have never met anyone who deliberately designs or develops inaccessible websites or applications; rather, inaccessibility arises from a lack of information. The vast majority, once they understand how people with disabilities access the web or mobile apps, the barriers they face, how easy it is often to remove those barriers, and how accessibility benefits everyone, quickly internalize best practices and apply them in their daily work. Some of the most common reactions are: "If I had known all this before...", "My eyes have been opened...", "I had never thought about this before...", and "I don't know anyone with a disability...".

Dissemination, awareness and training are the fundamental pillars of making technology truly inclusive, and I hope this book will serve as another building block in that effort.

It has been a pleasure to once again share this journey with the enthusiasm and dedication of Olga Revilla, as well as an honor to have the valuable support of Emmanuelle Gutiérrez y Restrepo, an outstanding reference for all those committed to digital inclusion.

Prologue

Emmanuelle Gutiérrez y Restrepo

Director of the Sidar Foundation – Universal Access

When I wrote the foreword for WCAG 2.0 Made Simple, I predicted that it wouldn't be the last time I would write a few words in support of its author. The success of that first edition, pioneering both in its purpose and in its accessible approach, has now found continuity in a new stage: this edition of Web Accessibility. WCAG 2.2 Made Simple, co-authored by Olga Revilla and Olga Carreras.

In this book, the authors do much more than just update the content to reflect the latest version of the Web Content Accessibility Guidelines. They offer a valuable and thoughtfully written resource that translates the complexity of the WCAG 2.2 into clear, understandable, and—most importantly—applicable guidance for professionals from a wide range of roles in digital content creation.

Those who work in web development, UX/UI design, content strategy, digital communication, or accessibility evaluation will find in these pages both clarity and inspiration. The book combines technical precision with ethical vision, making it an essential tool for those who understand that accessibility is not merely a legal requirement or a box to tick—it is a matter of human rights, dignity, and inclusion.

Just as in previous editions, this is not a translation or mere restatement of the WCAG 2.2. It is a curated synthesis of techniques and practical knowledge for implementing each guideline and success criterion. It is written in accessible language for information architects, interaction designers, developers, accessibility specialists, and anyone else involved in the creation of web content and applications.

This edition is being published in English again, collecting all the contributions that were made in the Spanish editions. It opens the doors to a wider international audience and extends the authors' clear and thoughtful guidance to those who may not have had access to these insights in their original Spanish-language editions. This English edition contributes significantly to the global conversation on digital inclusion, and I am certain it will help spread the principles of universal design further and more effectively.

Both authors are well known in the Spanish-speaking accessibility and UX communities, not only through their professional work but also through their commitment to knowledge-sharing via their blogs, Itákora and Usable y Accesible. As active and valued members of the Sidar Network, Olga Revilla and Olga Carreras represent the spirit of collaboration and mutual support on which our community is built.

I am proud to support this publication and confident that it will be warmly welcomed, not only because I wish both authors every success, but because it will be a sign —yet another— that accessibility is continuing to take root among those responsible for shaping our digital world. As someone who has spent more than two decades advocating for accessibility in the so-called “Information and Knowledge Society,” I know that the success of this book will be yet another sign that the seeds we have planted are beginning to bear fruit.

Contents

Introduction to Web Accessibility	1
Implementing Web Accessibility policies.....	21
The WCAG Guidelines	39
How to make an accessible website	55
Assessing Web Accessibility with WCAG-EM.....	71
Principle 1. Perceivable.....	83
Principle 2. Operable	135
Principle 3. Understandable.....	187
Principle 4. Robust	217
Accessible PDF documents	225
ARIA, the ally of accessible HTML.....	251
Accessible Single Page Applications	285
Accessible Design Systems	295
Caring for the words	309
Validation tools	333
Work tools	359
Summaries and Diagrams	369
Glossary.....	391
Global Index	395

Introduction to

Web

Accessibility

In this chapter, we introduce the concept of Web Accessibility, its importance, and how to apply it at every step of an interaction.

Moreover, we relate two concepts intricately linked to accessibility: usability and ethics in interaction design.

We emphasize the importance of evaluation at various levels and through different approaches.

We also recommend the minimum knowledge that professionals building websites should have.

Finally, we demonstrate how to implement and manage a Web Accessibility policy within an organization.

What is Web Accessibility?

Anyone working in the digital field will have a default requirement in their projects: "to be accessible," and many won't know where to start. Many people think that making a website or app accessible is so that blind people "can see" the content of the screens. However, Web Accessibility goes far beyond blind people or "being able to see" screens. And it is also much more than a mere functional requirement or an imposed rule.

Web Accessibility is an opportunity to ensure that as many people as possible can perceive, understand, and operate our websites or apps on different devices.

Watch the following two people using their mobile to receive directions to reach their destination. The first is a blind user with headphones who listens to the indications by manipulating the screen of their mobile phone, and the second is a user who drives a car and interacts with the mobile phone through voice. Both individuals have limitations in their vision and screen handling, one of which is permanent and the other situational; however, both benefit from an accessible design that allows them to access the content and functionalities necessary to achieve their purpose without relying on the visual sensory channel.

Limitations that affect people with disabilities permanently may affect others temporarily, or they may experience limitations because of their situation at a particular time. Table 1 (based on the [Microsoft Toolkit "Inclusive Design and Disabling Situations"¹](#)) shows some limitations and how the same solution can solve them all.



¹ <https://www.microsoft.com/design/inclusive/>

Table 1 Diversity as a source of inspiration

Limitation	Permanent	Temporary	Situational	Solution
No vision	Blindness	Cataracts	Not looking at the screen, like when driving a car	Facilitate voice interaction
Low vision	Myopia Magna	Conjunctivitis	Legal information in fine print	Allow screen zooming
Difficulty distinguishing colors	Color blindness	Glare from strong lights	Darkened screen due to low mobile phone battery	Sufficient contrast
Hearing loss	Deafness	Not understanding the language well	A noisy environment or a silent one, such as a library	Subtitling a video
Low mobility	Spasticity, loss of an arm, Parkinson's, osteoarthritis	Sprain, tendonitis, carpal tunnel syndrome	Using your mobile phone with one hand while the other is holding onto the bus	Interaction points of sufficient size
Difficulty fixing and maintaining attention and concentration	Autism, or attention deficit hyperactivity disorder	State of anxiety	Do another activity, such as looking at your phone when babysitting	Easy organization of content
Reading comprehension	Dyslexia, intellectual disability, deafness from birth	Low educational level, low language proficiency	A complex setting, like a bustling street	Simplicity of the texts

For example, the problems a deaf user experiences in understanding a video are the same as a listener experiences with the same video in a boisterous environment, or the difficulties that a user with Parkinson's has in aiming with a pointer on the screen are similar to those that an older adult can have when moving a mouse.

Disability is an opportunity to explore how technology creates limiting situations while also being able to solve them. The solutions to which people with disabilities are entitled (permanent or temporary) are the same ones that solve the situational

problem of many people without disabilities. However, the first argument should be more than enough to take Web Accessibility into account.

Using accessible design techniques, we can ensure that an extensive range of people in different situations can enjoy experiences that go beyond "reading" the web, such as being able to fill out a job application, laugh at a joke that is told in a video, or make a videoconference with the family. All this is regardless of whether the user accessing our experience has a disability, is unskilled, is in a complicated environment, or connects through a device with limited functionality.

Therefore, **accessibility is more than a mere requirement; it is the opportunity to generate a quality product that makes life easier for all the people who use it**: an accessibility problem is not only blocking for a user with a disability, but it is also a nuisance for many other people. With Web Accessibility, we ensure that all people exercise their right to be autonomous on the internet, that is, to be able to access content and functionalities regardless of their functional diversity (sensory, motor, intellectual, or mental) or the context of use (for example, technological or environmental conditions).



Web Accessibility is defined as the set of features that a website, mobile application or digital document must incorporate so that as many people as possible in as many circumstances as possible can perceive, use, and understand it.

Accessible interactions

An interaction is a process by which a user and a system (i.e., any device with a functionality) communicate through its interface.

Some ways of communicating **between people and systems** can be:

- Interacting by **voice**: for example, by asking Alexa, Siri, or Google for weather information.
- Moving, pointing, and selecting with a **pointer**, such as a mouse.
- Typing with a **physical keyboard**, such as on a laptop.
- Typing with an **on-screen keyboard**, such as on a mobile phone.
- By directly touching a **screen**, such as with a tablet.
- Inserting an object into a **slot**, such as a credit card at the ATM.
- Positioning the **body** in a specific way, such as in a virtual reality video game.

Some forms of communication between **systems and people** can be:

- **Audible**: for example, when emitting a beep indicating a success or an error.
- **Visual**: for example, by showing the effect of moving the pointer or entering text on the screen.
- **Haptic**: for example, controllers in a virtual reality game vibrate when another avatar is nearby.
- **Physical**: for example, by handing out a printed piece of paper when generating a parking ticket.
- **Thermal**: for example, changing the temperature when turning off the heating.
- **Olfactory**: for example, an air freshener emits a pleasant smell when the user enters the room.

According to the disabling situations that have been compiled in Table 1 we can consider how permanent, temporary, or situational limitations can affect this communication. For example, a user wearing winter gloves or with moderate Parkinson's disease will have difficulty inserting the card into the ATM if the slot requires precise alignment. Similarly, a deaf user and a user in a nightclub will not be able to hear a phone call.



In the chapter [Work Tools](#), you will find some simulators that allow you to put yourself in the shoes of people with limitations in their interaction with the systems.

To design an accessible interaction, we must think about the **five steps that make up the interaction** of people with systems and ask ourselves the following questions:

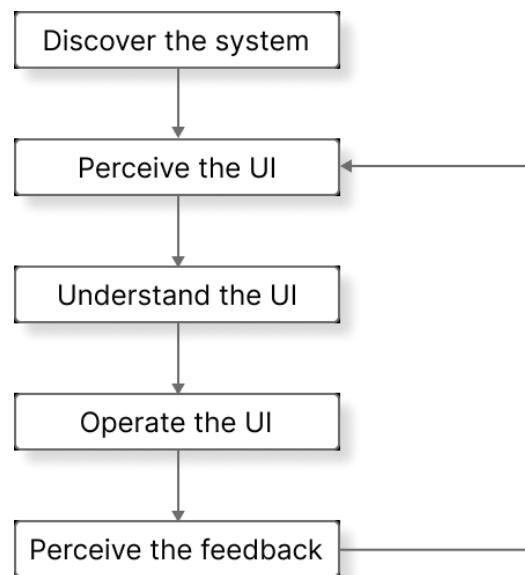


Illustration 1 The interaction sequence

- **Step 1. The user discover the existence of a system within their² environment.**
How do they find out the system? What barriers and limitations prevent them from discovering it?
- **Step 2. The user perceives the interface, its elements, states, and properties.**
How do they perceive the interface? What senses are involved? What happens if a user is unable to use that sense?
- **Step 3. The user understands the interface and how to communicate and manipulate the controls (affordances).**
Is the information shown sufficient? Is it excessive? Is it easy to understand? Is there enough time left for them to study how to use it?
- **Step 4. The user operates with the controls on the interface.**
Can they manipulate the controls easily? What if they can't use the input mechanism? What alternative do they have? Can they make a mistake?
- **Step 5. The user perceives and understands the feedback of the operation** and is willing to interact with the system again. Limitations of perception (step 2) and understanding (step 3) come into play again. How do users perceive the feedback? What if they can't use the implied sense? Is the information returned easy to understand?

² This book uses the “singular they” to achieve gender-neutral language.

When designing an interaction for a system, we must foresee the limitations (permanent, temporary, and situational) that may occur at each step and provide solutions to solve them if they arise.

These limitations can prevent, hinder, or reduce users' ability to discover, perceive, understand, or operate the interface.

Ethical aspects of an accessible interaction

Designing an interaction involves much more than establishing communication between humans and machines. In our opinion, the following ethical principles should be considered in every interaction so humans can control what happens, when, where, how, and why.

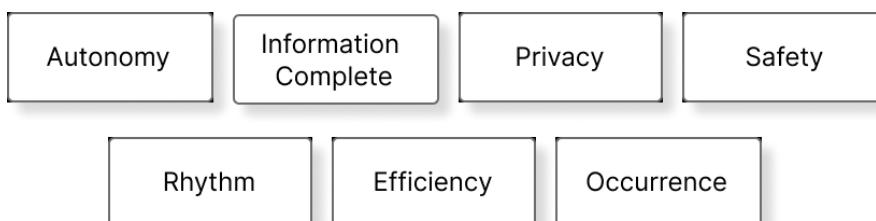


Illustration 2 Ethical principles of interaction control

- | | |
|-----------------------------|---|
| Autonomy | First, the user must act independently without the help of another person. User's autonomy is crucial to their self-esteem and includes the ability to make choices and decisions, and to assume the consequences of those choices. People need interfaces to interact with machines, which are more common in some cases (such as a screen) and less common in others (such as a screen reader). Designing and programming robustly will allow people to use the proper mechanism for their needs. |
| Complete Information | The system must provide sufficient and necessary information so that the user is aware of the actions to be carried out and their consequences. It is not required to include all the information in such a way that it overloads cognitive capacity and causes lethargy. It is about giving complete information to help make an informed decision. This is especially necessary on transactional sites where acts can have serious consequences, such as a wire transfer, a purchase in a store, or sending a resume to a job offer. |

Privacy	In addition, the system must guarantee the privacy of the user. Not only to ensure the rights established by the data protection law of your country (in the European Union, it is the "General Data Protection Regulation" or GDPR) but also that the contents of the conversation between the user and the system do not spread in the environment without user's permission . For example, if the credit card number is requested using a numeric keypad, the system should not repeat it out loud to confirm whether it is correct. Or, if a password is being entered, have a mechanism to mask it and prevent anyone from seeing it.
Safety	The principle of privacy is linked to the safety principle. The user must feel that their conversation is safe, that it is happening, that their actions have consequences, that if there is a risk, they are warned , and that if they make a mistake, they can recover from that mistake.
Rhythm	On the other hand, the interaction can be punctual, as in elementary mechanisms (for example, the doorbell). Nonetheless, it is usually sequential, and it can happen as many times as the user decides, so we must also consider the variable "time." The user must manage the pace of the conversation (as far as the technology allows). Does the user have enough time to carry out or understand the actions? If the system takes time to react, does it give the impression that something is wrong?
Efficiency	At the same time, the conversation must be efficient. There should be minimal clicks and dialogues but enough , so the user does not suffer at each step. If the user has to spend a lot of time or effort to complete a task, they will probably get frustrated and abandon the process.
Occurrence	In the same way, it should be the user who decides when that conversation occurs, when the interaction begins, and when it ends, and leaving when the user wants. We all have examples of operating system updates that block the use of our device at the most inopportune time.



In general, if we enhance the user's **internal control**, meaning the user perceives that events occur mainly as a result of their actions, they will be happier and react more positively to our system.

If, on the other hand, the user perceives that they do not have that control but that it is **external** to their actions, they will develop anxiety and will reject our system.

Usability aspects of an accessible interaction

Accessibility, in the end, is still the usability of interfaces extended to a broader range of people and situations. When we design accessible interfaces, we make them more usable. Below are the [ten principles that Jakob Nielsen](#)³ used to describe a usable interface and how it aligns with an accessible interaction, as described in the Illustration 1:

1. **Make the system's state visible:** give constant feedback to the user so that they always know what is happening. (Step 1 & 5)
2. **Use the same language as the user using the system,** with understandable messages, words, and terms familiar to them. (Step 3)
3. **Give control and freedom to the user using the system:** the system must adapt so that the user fully controls it. (Step 2 & 4)
4. **Be consistent and follow standards:** The user must understand the meaning of the system's words, actions, or situations according to the platform's standards. (Step 2 & 3)
5. **Prevent errors:** the system must be able to prevent and avoid mistakes before the user encounters a message indicating that something has failed. (Step 3 & 4)
6. **Minimize the user's memory load using the system:** the system must minimize this load through objects or images that make it easier for the user to recognize them. (Step 3)
7. **Be flexible and efficient in use:** the system must recognize the type of user who uses it and let them personalize their user experience. (Step 2 & 4)
8. **Offer aesthetic dialogues and a minimalist design:** the system must be able to provide the minimum relevant information for the user who uses it without losing sight of the essential content of the website. (Step 2)
9. **Help the user using the system recognize, diagnose, and recover from errors:** the system must be able to express errors in a language that the user knows, providing information about what has happened and proposing some solution. (Step 4 & 5)
10. **Provide help and documentation:** located in a place that is easily accessible to the user using the system, written in a language that the user knows, and, if possible, that is not exceedingly long. (Step 3)

³ <https://www.nngroup.com/articles/ten-usability-heuristics/>

What professionals should know about Web Accessibility

Traditionally, Web Accessibility has been linked to front-end developers and accessibility auditors. Thinking that only these profiles make a site accessible is as wrong as to believe that an accessible building is only a matter for the bricklayers. If the rest of the professionals are not involved and prepared, developers can only cover up errors in the best way possible.

Implementing an accessible website is the daily life of many digital professionals, especially **content creators**, **UX/UI designers**, and **front-end developers**, who are on the front line of the battle with the content, the interface, and the code: any change introduced, no matter how small, can mean achieving or losing the accessibility of a process or a set of pages. We have dedicated a specific chapter to them.

However, **all the profiles involved** influence the final result of accessibility and must, to a greater or lesser degree, have some knowledge of what their decisions entail. Accessibility is achieved through a chain of actions that go from the beginning of the concept to the final delivery to the customer who has contracted it:

- **Project managers** must motivate and train their team, plan and allocate resources, and hire external auditors if necessary;
- **Information architects** must integrate a simple structure and navigation with alternatives;
- **User experience researchers** must include people with disabilities, as well as limiting situations, in their tests and in the application of the *Persona* method to guarantee the diversity of profiles;
- **SEO specialists** must push to make pages accessible because they know that search engines love them, and manage to better position them;
- **Service designers** must think about extreme situations of use of their business idea to expand the sales target and facilitate its use for all people;
- **Analytics consultants** must detect accessibility issues (e.g., filtering by "age" or "device" can find significant differences in some groups of people);
- **Video and audio creators** must consider subtitling, intense changes in luminosity or audio contrast, among other things;
- **Quality reviewers** must not only internally assess compliance with the guidelines using automated tools, but they also must incorporate the tools used by people with disabilities into their manual checks and ensure they work correctly;
- **Back-end developers and database and systems administrators** must provide support in configuring services, data management, or developing the APIs needed in each case.

However, not only the professionals in direct contact with the project influence the result. The rest of the organization must also know the importance of accessibility:

- **Purchasing managers** who decide to hire content management tools must be aware of the accessibility implications of this software, as well as what barriers or facilities they will pose for the team that is going to use it;
- **IT managers** must establish the necessary policies to plan and manage the accessibility of the website under their responsibility in the long term;
- **Human resources managers** must incorporate this skill and sensitivity into their recruitment processes and employee training plans;
- **Directors of the legal and financial areas** must be aware of the regulations, as well as possible litigation and fines for non-compliance with the legislation;
- **Directors of marketing, communication, and public relations** must report on accessibility support, as they probably want the company to have a good reputation in the eyes of its customers and shareholders;
- **Commercial directors** may apply for calls for projects where the company's quality processes are considered, including Web Accessibility;
- finally, **CEOs** must be the one who leads the philosophy of accessibility throughout the company, not only on the web but in all aspects of the company.

Accessibility is a team effort. Hence we all have a role: providing suitable means, detecting problems, integrating techniques, reviewing code... Delegating all responsibility to copywriters, UX/UI designers, and developers is a common mistake that we must banish and start taking responsibility for the accessibility part that falls to us.



The **W3C** (World Wide Web Consortium) has established a series of responsibilities for each professional profile: [Accessibility Responsibility Breakdown](#).⁴

⁴ https://www.w3.org/community/wai-engage/wiki/Accessibility_Responsibility_Breakdown

What you need to know according to your profession

Once the need to involve all profiles in Web Accessibility has been conveyed, we will focus on content creators, UX/UI designers, and front-end developers, who are the primary recipients of accessibility techniques. For this reason, we summarize and explain the concepts that affect them in the following pages.

This does not mean that each profile should adhere strictly to these concepts, but rather that they should at least review what falls within their scope of action and lend a hand to the rest of the professionals. Additionally, on many occasions, they share criteria, as collaboration between professionals is necessary to ensure the site's accessibility.

Depending on the project, a blurred line separates each profile's responsibilities: sometimes, the UX/UI designer enters the content, the content creator touches the code, or the developer creates audiovisual content using the code.

Below, we delimit the functions of each professional, although in reality, a user can assume several roles according to the requirements of the project:

- When we refer to content creators, we mean individuals who create texts, images, videos, or audio content and publish them on web pages, typically through a content management system.
- **UX/UI designers** are the individuals who determine the graphic aspect, the organization of elements, and their interaction within the website.
- Finally, **front-end developers** are responsible for HTML and CSS layout, with the complementary implementation of ARIA, plus the behavior in JavaScript (with Angular, React, etc.).



In the chapter [Summaries and Diagrams](#), there are tables that you can use as a checklist.

What you need to know if you are a content creator

Text writers, illustrators, graphic designers, photographers, audio and video editors, and content managers are responsible for ensuring that website users perceive and understand the information intended for them.

Content creators must seek **simplicity** in all its forms. To facilitate the **readability** and understanding of the texts, they should be structured into sections preceded by clear and unique headings on the page, and written in simple language that explains technical terms or abbreviations to eliminate any doubts. Other mechanisms that help are lists, a brief introduction with a summary of the content, or the inclusion of "Easy Reading" or audio versions.

Texts, including instructions and aids, should be sufficiently explanatory and informative, **avoiding** instructions based solely on sensory characteristics such as the component's color, shape, or location.

Clarity is essential in the wording of the **page title and each link; in both cases, they must undoubtedly convey the content that can be found**. Links must be consistent so that two identical links navigate to the same resource. On the other hand, the page's title should be short, concise, and unique on the website.

Always indicate the **language** of the page and content so that screen readers can pronounce them correctly and machine translators can work more efficiently.

Suppose audiovisual resources such as videos and audio are used on the website. In that case, the creators must provide **alternatives** for people who cannot perceive them in their original form: for example, subtitles, sign language, or verbatim transcription of the content. This content should not start playing by default, and in the case of sound, there should be enough contrast between the main track and the background sound to be well understood.

Certain flashes and movements can cause **epileptic seizures and dizziness**, so content creators must be careful when making them and check them before including them.

A concise description should be included for **images** that convey the same information as the image unless the image is decorative, and text images should be avoided.

Color is essential; the **color contrast** must be considered, and information must not be transmitted solely by color.

It is essential that content editors correctly use the **options offered by the content management editor**: that they do not paste content directly from Word nor simulate HTML elements, but include headings, lists, or tables with the editor's options. In many cases, the editor will also allow to mark citations, abbreviations, the language of the contents, the header cells, or include summaries in the tables.

Finally, those responsible for the **documents** attached to the website (PDFs, text documents, spreadsheets, etc.), should know that these must also be accessible. We dedicate an entire chapter in the book about this.

What you need to know if you are a UX/UI designer

UX/UI designers decide the graphic look, the organization of the elements, and their interaction within the website.

First of all, probably the most well-known thing for all UX/UI designers is that they must **ensure the contrast** between the color of the text and the background to ensure that it can be read. They should also apply this revision to non-text elements that convey information, such as graphics and icons, and to the edge of form fields or the focus indicator.

Color can be used with other assets (sizes, fonts, texts, icons, etc.) to **create easy-to-identify elements**. Think about people who can't distinguish colors: if you use only color to convey information, especially in charts, links, or calendars, they won't understand them.

Color is not the only design aspect that makes **texts more accessible** to read; there are other requirements or recommendations related to alignment, margins, font type, or size, such as not justifying texts, or paragraph spacing being more significant than line spacing.

Additionally, one of the key principles that UX/UI designers must adhere to is **consistency**. They should create a consistent structure and navigation system throughout the entire site. Users must be informed where they are and where they can go, providing them with several ways to reach the same content (through menus, related links, a web map, a search engine, etc.). One of the novelties of WCAG 2.2 is precisely the homogeneity of the location with the help of the forms of contact so that people can find them quickly.

Another maxim is **clarity**; in case of doubt, always choose the most straightforward option. Clarity is critical, for example, in links. As we already mentioned in the section dedicated to content creators, the user cannot have any doubt about where a link will take them; therefore, for consistency, two identical links should always have the same destination. Although developers will be able to clarify the destination later, it is essential to consider it from the definition of the information architecture.

In terms of the **organization and presentation of the content**, in addition to being consistent throughout the pages, the UX/UI designer must use headings to divide the various parts of the page and the content into coherent blocks.

Although we have tried to create a straightforward design, people who will later access the website can **customize** it with the options of the browsers, with extensions, or with support products to adapt it to their needs. Additionally, we may include customization options on the site, such as controls to increase text size or switch to a high-contrast version.

Regarding the arrangement of elements on the screen, UX/UI designers should keep in mind that other elements of the design should not obscure elements that can receive **focus**. It is typical for headers and fixed footers, or other adhesive layers, to cover the keyboard focus and cause problems when viewing pages when zoomed in.

Clarity should also be a maxim in the **design of forms** in all their phases and statuses (unsent, sent, with errors, etc.). On the one hand, field labels and instructions should make it easier to enter information and prevent the user from making mistakes. If you make a mistake, identify the errors, and offer suggestions to recover from the error. On the other hand, data entry should be facilitated, especially when it has already been entered. Special attention must be paid to the authentication process to avoid involving a cognitive effort that prevents some people from carrying it out.

The variety of **screen sizes** is enormous, which UX/UI designers must consider when designing page versions in different screen resolutions. They must also respect a minimum recommended size for interactive areas of 44x44 pixels or, at least, 24x24 pixels.

When designing, images, videos, audio, animations, flashes, transitions, and effects are also used to enrich the message. **Audiovisual resources** must have an alternative for people who cannot perceive them in their original form, and, in addition, there must be a way that allows people to control their reproduction, so the necessary mechanisms must be designed for this.

As content creators, UX/UI designers should know that certain **flashes** and movements can cause epileptic seizures and dizziness and be careful when designing them and verify them before including them.

Moreover, when designing the interaction, it must be considered that the user must be able to confirm or review their actions and **not be bothered by unexpected interruptions or asked to re-authenticate themselves repeatedly** in the system. They should be allowed to make **decisions autonomously**, such as when to submit a form or navigate from one web page to another.

At the interaction level, UX/UI designers must ensure everyone can easily use the website, especially those with difficulty using the mouse or phone. In complex touch or mouse interactions (such as drag, swipe or multi-finger gestures), they must ensure keyboard interaction and simple gestures as well. WCAG 2.2 requires dragging interactions, such as in sliders or drag-and-drop sortable panels or lists, to have a pointer alternative.

Finally, another way to make life easier for people who are less able to respond quickly or are less skilled with technology is to **manage the time** they have to carry out interactions without being overwhelmed, thus preventing them from making bad decisions in unnecessary haste.

What you need to know if you are a front-end developer

This section includes the profiles responsible for the pages' code, such as layout or defining behavior.

In the past, assistive products such as screen readers interpreted a web page directly from its code. Currently, most of them do so through the Accessibility API. For this reason, WCAG 2.2 has decided that it is no longer required to validate HTML or CSS code since, if an error affects accessibility, it will already become known in other criteria. However, it is still a good practice to **validate the code** to verify that it correctly follows the standard, as it facilitates not only the interpretation of the code by different technologies but also by other developers who analyze that code later.

The primary responsibility of developers is that the code is **semantically** appropriate to the information, relationships, sequences, and controls it generates. To do this, standard controls, such as the WAI-ARIA standard, should be used whenever possible, to which we dedicate an entire chapter. This standard allows you to add semantic information to page elements and helps to make dynamic elements (alerts, drop-down menus, carousels, etc.) accessible.

Developers must help people understand the interface by correctly **identifying the controls and their purpose**. This is especially important for links and form fields, which are traditionally complicated for users of assistive products.

Developers should help fill out **forms** by clearly and consistently identifying fields, help instructions, and errors so that they are noticeable, operable, and understandable to everyone who will fill them out. They must take particular care with authentication processes not to force people to remember or transcribe information. Allow them to copy their login details into text fields, and don't include complicated tests, such as puzzles or math calculations, during the process.

Although it is already widespread to use HTML, CSS, and JavaScript files for structure, presentation, and functionality respectively, developers must **separate the content from the presentation**, not only in the general layout of the pages but also in the images, avoiding text images, and defining the decorative images in the CSS. The content creators are responsible for providing many of the images on the website with a suitable **alternative**.

Moreover, developers will need to select an **accessible media player** that supports the different alternatives available for each video or audio content.

Developers must also ensure a proper **reading order** in any context, and it is a good practice that the visual order of the content matches its order in the code or the DOM.

In addition, fields, links, and other interaction or interface elements, such as modal windows, must be available to be used by the **keyboard** in a logical order, identifying at all times where the focus is and without it being trapped in a component. If **keyboard shortcuts are implemented**, developers must ensure that they do not conflict with those of the browser and that they are not associated with a single key.

Furthermore, developers must ensure that people browsing the website, including those who access it with a screen reader, get the necessary **feedback** on the consequences of their actions and are **in control** at all times, avoiding unexpected or unsolicited behavior such as moving the focus or navigating to another page without having requested it.

It also depends on the developers' implementation that the page is displayed correctly on **different devices, configurations, and screen sizes**, that it adapts when the screen rotates, that the text can be enlarged, and that it is possible to zoom without problems so that the elements are readjusted without the need to scroll horizontally.

Another way to make people's lives more accessible is to **avoid** repetitive navigation blocks between pages by providing mechanisms that allow them to skip them and to avoid automatic page refresh or data loss if the session expires.

Like content creators, they must indicate the **language** of the page and its different content so that screen readers can pronounce it correctly.

They may also be responsible for the **titles of the pages** or the pattern from which they are generated, so they should bear in mind that they have to be short, concise, and unique on the website and identify the page and the website itself.

Evaluation in Web Accessibility

The concept of evaluation is critical and transversal in Web Accessibility to know the starting situation and to measure the impact of the changes made. Throughout the book, we will show different approaches to understanding accessibility evaluation.

To help the reader get a clear picture of each of them, we anticipate them here:

- **Within your organization:**

- **Accessibility Maturity Model:** Measure the maturity of your organization regarding accessibility in terms of proof points according to several dimensions (communication, knowledge and skills...). The goal is not only to ensure Web Accessibility but also to encompass all organizational processes, which reflects the organization's engagement with Web Accessibility.
 - **Web Accessibility Implementation and Management Cycle:** quantify in a dashboard how the web accessibility is progressing within the organization (e.g., number of people trained, number of PDFs verified, number of pages with errors since last audit...).

- **Within the website:**

- **Assign a conformance criteria:** check that specific technical criteria are met. The outcomes are to be provided in a predefined way called "declaration of conformance", which depends on the regulation that the website needs to adhere (the most common is the W3C WCAG, but for instance, in the USA federal agencies much follow the format defined by Section 508). But even if the WCAG are used, specific statements model have to be provided in different regions (e.g. the Implementing Decision (EU) 2018/1523 of the European Commission).
 - If **specific methodologies** are used to validate the websites (e.g., WCAG-EM methodology), a report of the resulting declaration of conformance of the results obtained after applying the methodology have also to be provided.
 - There exist **validation tools** that can help reviewing the website automatically or manually, and in general or for specific aspects.

- **Within documents:**

- **Automatic accessibility validators and Manual validation** using assistive tools must be performed to detect accessibility problems within specific content (e.g. pages, processes, PDF files) or formats (ARIA).

However, when possible, **interview users with disabilities** to gather their firsthand experience using your website.

Thus, as will be seen, it is essential not only to evaluate accessibility but also to communicate the evaluation outcomes in a clear and solid manner.

Implementing Web Accessibility policies

In this chapter, we introduce essential concepts for organizations.

First, we show the most common accessibility challenges and then how to convince organizations to cope with them.

Next, we present the Accessibility Maturity Model, proposed by the W3C, still on draft, to give a comprehensive overview of the organization's accessibility efforts.

Finally, we explain how to integrate Web Accessibility into an organization.

Why do websites have accessibility problems?

According to the [ContentSquare Foundation](#)⁵, 70% of digital content is not accessible to people with disabilities. The [Digital Accessibility Barometer](#) published by this foundation in 2024 -based on audits of 100 websites in six countries in different sectors- concluded that only 7% of the audited websites show consistent effort in prioritizing accessibility, while 40% of websites show little or no effort. Other study made by [WebAIM](#)⁶ detects an average of 51 errors per page with their automatic tool across the one million home pages they have analyzed.

Why does this happen? Nobody wants to create an inaccessible website by default, but accessibility problems occur. There are several reasons.

First, because **Web Accessibility is not so easy to learn**. Many basic training courses just cover the main concepts, and when professionals have to apply what they have learned, they realize they still have to learn many new things by themselves. Besides, most of the courses are generic, and different profiles should be trained differently depending on their focus: a web developer's needs differ from those of a quality tester, a PDF creator, or a project manager.

Second, because **Web Accessibility is not easy to apply**. Creating and maintaining an accessible website is a team effort sustained over time. A small wrong decision, most probably inadvertently, by just one of the team members can make a complete website inaccessible, and that error may go unnoticed by the rest of the team for a long time. Imagine many people in a rush doing many actions on the website. Eventually, errors will happen even in organizations with strong engagement,

⁵ <https://contentsquare-foundation.org/learn/>

⁶ <https://webaim.org/projects/million/>

training, dedicated accessibility professionals, and quality processes, and they won't be disclosed until an audit is performed, or a user complains.

Creating and maintaining an accessible website **cannot show a direct, measurable, impactful ROI (Return On Investment)** as other disciplines can, like SEO, SEM, CRO, Automation, or Growth Hacking. This makes organizations usually prioritize their resources and budget toward them instead of an accessibility specialist who supervises continuously, coordinates efforts, democratizes knowledge, designs processes, and suggests improvements. For this same reason, known accessibility issues are not always prioritized in the backlog.

Most websites are **living products that change frequently**. Newspapers, for example, are made by trained professionals who are under pressure to publish at the right time. Spending some minutes adding alternatives to images or subtitles to videos will erode their competitiveness against other media.

Other websites have a **short life**, like promotional landing pages, pages being A/B tested, or just the MVP for a concept. In these cases, Web Accessibility is not usually considered because '*why putting effort into something so ephemeral?*'. However, many of them become permanent in the end, and drag their accessibility problems during their whole lifetime.

Sometimes, **changes are not even supervised** nor verified. For example, social networks change every second by including end-user-generated content. In this case, organizations can use partial declaration of conformance for content from uncontrolled sources.

Other organizations believe that accessibility restricts their **creative freedom** or forces them to use less visually appealing interfaces, so they literally just don't care, don't review, or don't fix anything. On the contrary, Web Accessibility provides techniques for being as creative as you want, but learning and applying them takes time and effort.

Some organizations use **third-party tools** like Content Management Systems (CMS), plugins, or frameworks with accessibility flaws that organizations cannot fix. This excuse is similar to the **legacy code**: pages are so old that they are not worth it, or it is very expensive to improve them. There is a concept in some regulations regarding "[disproportionate burden](#)"⁷. That clause allows organizations to avoid full compliance without penalty if the organization proves that solving problems costs more than the organization has funds for, it is not significantly impactful for disabled users, or it is a wasted effort because they are actively working on a better version.

Finally, some organizations use [overlays](#)⁸ to fix known problems by injecting third-party code to change the website's front-end code. They don't guarantee fixing all the problems and they even can create new ones for users with disabilities, so it is pretty risky to use them as an easy solution for accessibility requirements.

⁷ <https://www.makethingsaccessible.com/guides/understanding-disproportionate-burden/>

⁸ <https://overlayfactsheet.com/en/>

Arguments for investing in Web Accessibility

Advocating and working toward Web Accessibility in some organizations can be complicated and exhausting without the support of corporate management. Let's consider some arguments for incorporating web accessibility into organizations.

First, because access to web content and functionalities is a [Human Right](#)⁹ ratified by [99% of the countries](#)¹⁰. Individuals with disabilities encounter daily discrimination and obstacles that hinder their ability to engage with society on an equal basis with others. For instance, they often face denial of their rights to participate in education and employment, to request medical treatment, or to open a bank account, just because the website they have to use has accessibility problems. As web makers, we are due to solve these issues and provide the best experience possible to all.

Legislation is the main pusher for institutions to implement accessibility in their digital projects, including websites and mobile applications. [Most of the countries](#)¹¹ have laws and policies related to Web Accessibility like the [Directive 2016/2102 \(EAA\) in the European Union](#)¹², or the [Rehabilitation Act \(Section 508\) in the USA](#)¹³.

Regulations can establish **severe sanctions and fines**. For example, in [Ireland](#)¹⁴, CEOs of non-accessible websites can face fines ranging up to €60.000 and 18 months in prison if their websites are inaccessible by June 2025. In [California](#)¹⁵ (USA), lawsuits had been set for up to \$2.5 million.

However, there are **other benefits** that organizations must consider as well for implementing Web Accessibility:

- By making websites accessible, the target expands, with the opportunity to reach a larger audience and increase sales and profits. According to the ContentSquare Foundation, "there are 1.85 billion people with a disability, which is larger than the population in China, with \$1.9 trillion annual disposable income. 71% of them will leave a website they find difficult to use." This argument alone should be eye-opening to stakeholders. But think now in the elderly or in situational disabilities, and the number just increases.
- Web Accessibility compliance is a competitive advantage in procurement processes. Most websites are not just informative, but transactional. An e-

⁹ <https://www.ohchr.org/en/instruments-mechanisms/instruments/convention-rights-persons-disabilities>

¹⁰ <https://indicators.ohchr.org/>

¹¹ <https://www.w3.org/WAI/policies/>

¹² <https://web-directive.eu>

¹³ <https://www.section508.gov>

¹⁴ <https://www.forbes.com/councils/forbesbusinesscouncil/2024/05/30/ireland-web-accessibility-everything-you-need-to-know/>

¹⁵ <https://adasitecompliance.com/ab-1757-california-accessible-website-legal-guide/>

- commerce that wants to sell its products to a town council, a bank that is used to collect taxes, or a tool that enables civil servants to do their jobs are just three examples of websites that must comply with web accessibility standards to be considered. [New Zealand Government](#)¹⁶ performs 24 actions to embed Web Accessibility in their procurement process when purchasing web-based products, services, or development work.
- Web Accessibility improves **search engine positioning**. Although [Google](#)¹⁷ has not recognized it officially as a ranking factor, many of the accessibility practices have a strong impact on search performance. In turn, [TikTok](#)¹⁸ uses the content of the captions in their algorithm.
 - SEO bots are not the only **artificial intelligence (AI) bots** that can use your web content and functionalities. **Virtual agents** can read, summarize, and present contents to users who ask them complex questions, so the simpler the contents are, the better answers the Generative AI tools will elaborate (and less dedicated customer support staff will be needed).

In the same direction, **AI agents** can perform actions easier on your website if the functionalities are perceivable and operable following accessibility guidelines because they are programmatically determined. This can lead to automation that will use your website to perform an action repeatedly (for example, buying a product recursively).



Jakob Nielsen has written an interesting article about how [AI Agents will revolutionize UI Design and Accessibility](#).¹⁹

- Web Accessibility also improves **satisfaction rates and user experience** for all. Elements such as intuitive navigation, organized content, and easy-to-read fonts enhance the experience of all users. As commented before, there is a close relationship between usability and accessibility. So, typical User experience KPIs like Net Promoter Score (NPS) or Customer Effort Score (CES) are affected by improving Web Accessibility.

¹⁶ <https://govtnz.github.io/web-a11y-guidance/ka/embedding-a11y-in-your-organisation/a11y-in-web-procurement/include-a11y-in-the-procurement-process.html>

¹⁷ <https://developers.google.com/search/docs/essentials>

¹⁸ <https://sproutsocial.com/insights/tiktok-algorithm/>

¹⁹ <https://www.uxtigers.com/post/ai-agents>

- Web Accessibility positively impacts **the brand as it reflects a high degree of social responsibility**. Improvements in organizational reputation usually lead to increased trust among stakeholders (current and potential employees, partners, politicians, etc.), higher customer loyalty, or positive media coverage.

Although Web Accessibility's direct ROI is complicated to calculate, there are benefits that can be quantified. A report from [Forrester Research](#)²⁰ estimated that accessibility and user improvements bring back \$100 for every \$1 invested.

²⁰ <https://www.boia.org/blog/roi-of-web-accessibility>

Accessibility Maturity Model

Integrating Web Accessibility into an organization's daily processes is not a one-person, one-time effort but the coordination of all the professionals involved in creating the website during the lifetime of that product.

It involves allocating a series of professionals, time, tools, and other resources, establishing activities, and actively involving all those who make decisions that affect the digital product. So, integrating Web Accessibility into an organization's workflow and quality governance can be challenging. Some organizations have dedicated personnel to promote accessibility, but many have never heard of it and have no plans to integrate it. This oversight can hinder their capacity to deliver accessible products and services.

To solve this, the [W3C²¹](#) is working towards a framework for measuring and assessing accessibility maturity, linking teams toward common goals and objectives. The Accessibility Maturity Model differs from accessibility conformance testing:

- **Conformance testing** assesses how well a specific product meets accessibility standards. The outcomes of this testing offer insights into the accessibility level of that particular version of the product.
- In contrast, **maturity modeling** evaluates an organization's long-term capability to create accessible products. The results of a maturity modeling assessment give a comprehensive overview of the organization's accessibility efforts, highlighting areas of strength and identifying where enhancements can be made to eliminate barriers.

This model's goal is not only for Web Accessibility but for all the processes inside the organization, although it reflects the organizational engagement with Web Accessibility.

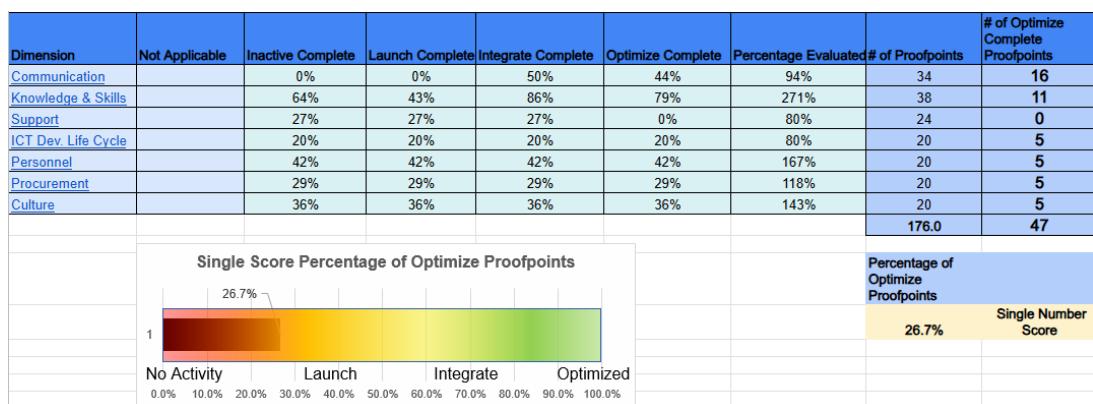
The model is structured around **seven key dimensions**:

1. **Communications:** Information related to an organization's accessibility and the accessibility of all internal and external communications.
2. **Knowledge and Skills:** Training to address gaps in accessibility operations.
3. **Support:** Assistance with accessibility provided to both internal employees and external customers with disabilities.
4. **ICT (Information and Communication Technology) Development Life Cycle:** The integration of accessibility considerations from initial concept to design, development, testing, Accessibility Conformance Reports (ACR) production, user research, maintenance, and eventual obsolescence.
5. **Personnel:** Job descriptions, recruitment practices, and disability-related employee resource groups.

²¹ <https://www.w3.org/TR/maturity-model/>

6. **Procurement:** identifying and acquiring the accessible products required by an organization, involving activities such as sourcing, negotiation, and selection of goods and services.
7. **Culture:** The attitudes, awareness, and behaviors surrounding accessibility, encompassing internal interactions, perceptions, and decision-making processes.

Each dimension has to be evaluated with several proof points. For example, an item in Knowledge and Skills is “Accessibility training when onboarding all new employees”. Other example would be in ICT Development Life Cycle: “Testing process documents steps for manual accessibility testing, utilizing assistive technology”. An [Excel Spreadsheet template](#)²² is provided by the W3C to help track and compute the proof points.



Screenshot 1 Accessibility Maturity Model Excel spreadsheet

Finally, depending on the number of points computed, the organization is categorized into 4 stages:

- **Inactive:** The organization has no awareness and recognition of need of Web Accessibility.
- **Launch:** The organization has recognized the need of Web Accessibility organization-wide. Planning was initiated, but activities are not well organized.
- **Integrate:** The roadmap is in place, and the overall organizational approach for Web Accessibility is defined and well organized.
- **Optimize:** The organization has incorporated Web Accessibility into the whole organization, consistently evaluated, and actions taken on assessment outcomes.

²² <https://www.w3.org/TR/maturity-model/A11yMaturityTemplate.xlsx>

Steps to integrate Web Accessibility into an organization

Depending on the stage of maturity of the accessibility in the organizations, the needs are different, but the ways to integrate Web Accessibility are similar.

It should be noted that **the steps do not have to be sequential, but there may be overlapping situations**. For example, there may be daily work to implement improvements before defining objectives. An intense empathy phase may be needed after setting specific goals to convince those implementing the changes. The idea is to have the most excellent control of what happens but to be flexible enough to manage it most appropriately if the situation demands it.

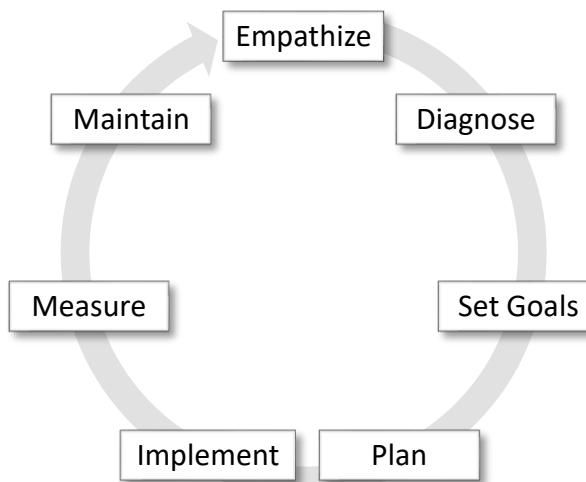


Illustration 3 Web Accessibility Implementation and Management Cycle

Empathize

First, the organization must **become aware** of its digital products' turning situations on and off. First, a maturity evaluation is recommended, to establish the starting point. Then, **involve the rest of the teams** through demonstrations, training, or experiencing them in their skin. In the chapter *Work tools*, you can find some [disability simulators](#) to feel how people perceive, understand, and operate your website. However, nothing will impact more than watching real users with disabilities using your website -or not being able to use your website. If possible, people with disabilities should be incorporated into the team and user testing sessions.



A good practice is to use the Personas technique to create permanent, temporary, or situational profiles with limitations. The following section shows several examples created for an online university.

Diagnose

Beyond the maturity stage, the next step is **to diagnose** the organization's state regarding Web Accessibility. Some questions the organization must answer are:

- Which are the **organization's** digital assets?
- Who is **responsible** for maintaining these assets?
- What **accessibility training** does the organization have?
- What policies and processes are **currently in place**?
- Which Web Accessibility **laws** apply to it? Does the organization have a minimum level that it must achieve?
- Have you carried out any **previous audits**?

In this step an accessibility audit is needed to understand the starting point and to discover the key issues. The [Chapter Assessing Accessibility with WCAG-EM](#) shows a reliable way to audit your organization's website's accessibility.

Set goals

With the diagnosis in our hands, we can start building and setting specific, measurable, and achievable objectives in a certain period of time.

Legislation will probably require reaching WCAG 2.2 level AA (we will explain the levels in the next chapter) by a specific date. This is an excellent goal to start working on. Considering the organization's maturity stage and nature, reaching levels above or below should also be considered.

All people must share this goal in the organization to enable everyone to row in the same direction.

Plan

Depending on the previous diagnosis and the objective we set, it is necessary to draw up a **work plan** that considers the needs and characteristics of each professional profile involved. Keep in mind that time and money should be scheduled for, at a minimum, team training, website review, and implementation of the necessary changes. In addition, there must be specific and committed people who take responsibility for each task. And all this within a particular period.

The work plan could include the following tasks:

- Create a **backlog** of the detected accessibility errors.
- Document the **processes and flows** of content publishing and website maintenance.
- Identify the necessary **accessibility control points** within these processes and flows.
- Define a **review plan** with clear managers, specifying the review frequency, method, and scope.
- Train and transmit **knowledge** to the people involved in the publication and maintenance of the website.
- Include **feedback** from people who access your website through a specific contact mechanism that allows you to collect their complaints and suggestions. The analysis and monitoring of this feedback helps to define corrective actions, to adapt websites and web applications to the real needs of people, and to continuous improvement.

Implement

Once each organization's profile is clear about what it needs to do, it is time to **implement** the necessary changes. Progressive changes should be made and continuously evaluated to achieve a greater sense of progress in pursuing the common goal and ensure its correct implementation and the involvement of the entire organization.

A way to prioritize the backlog can be the following:

- 1) **Blocking** issues (read [The 5 requirements of accessibility](#)).
- 2) Issues **reported by users** with disabilities.
- 3) Issues **found in the audit**, from lower to higher levels.
- 4) Issues found in **key tasks** (e.g., checkout, registration).
- 5) Issues found in **key content** (e.g., home page, contact page).
- 6) Issues found in the **frequently accessed content**.
- 7) **Common** issues. According to [WebAim](#)²³, 96% of all the automatic errors in homepages fall into these six categories, and addressing just these issues would significantly improve accessibility across the web:

²³ <https://webaim.org/projects/million/#errors>

Table 2. Most common WCAG 2 failures in homepages

WCAG Failure Type	% of pages
Low contrast text	79.1%
Missing alternative text for images	55.5%
Missing form input labels	48.2%
Empty links	45.4%
Empty buttons	29.6%
Missing document language	15.8%

Besides repairing what is broken, it is time to implement the long-term organizational policies you defined: making the design system accessible, training personnel, changing quality assurance procedures, including people with disabilities in user testing, ...

Measure

To monitor, track, and share improvements, you need a tool that helps **quantify** the website's quality. We recommend creating a dashboard with the following data:

1. **Goals**
2. **Global level** achieved in the audit.
3. **Number of issues found** in the audit detected & solved with their level.
4. **Complaints** or legal requests received and resolved regarding the inaccessibility of a website.
5. Number of **pages** with and without issues.
6. Number of **documents** (PDF, videos) verified.
7. Number of **testing procedures** before deployment into production.
8. Number of **accessible components** of the design system implemented.
9. Number of **people trained** in Web Accessibility.
10. Number of users with disabilities included in **user testing**.
11. Audit and reviews calendar

	Last Audit	Target	Target Date	Days to target	
Level	-	AA	28 June 2025	45	
	Last Audit	Target	Solved	In progress	Left
Errors A	4	0	2	2	2
Errors AA	25	0	10	1	15
Errors AAA	-	-	-	-	-
	Last Audit	Target	Solved	In progress	Left
Pages with Errors A	40	0	20	4	20
Pages with Errors AA	97	0	43	4	54
Pages with Errors AAA	-	-	-	-	-
	Non-verified	Target	Verified AA	In progress	Left
PDFs	146	180	120	2	60
Videos	21	21	2	5	19
Other media	5	5	0	0	5
	Received	Target	Solved	In progress	Left
Complains	20	0	5	1	15
Legal Requests	0	0	0	0	0
	Total	Target	Done	In progress	Left
Design System Components	300	300	20	5	280
People trained	50	50	10	0	40
Testing Procedures		Usertest w/d			
10		1			

Illustration 4 Example of a Web Accessibility dashboard in Excel



Illustration 5 Example of a Web Accessibility dashboard in PowerBI

Maintain

Once we achieve our goal, it is time **not to let our guard down**. Websites are constantly changing entities, and any of them can mean raising a barrier again. It is necessary to continue working to maintain the objective pursued and achieve new achievements. To do this, training is still required, being aware of new legislation, reevaluating the website, implementing improvements, and reviewing the organization's accessibility policy.



The W3C has produced a guide with tips and resources to help organizations [Planning and Managing Web Accessibility.](#)²⁴

²⁴ <http://www.w3.org/WAI/planning-and-managing>

Examples of "Personas" for an online university

Professor – Peter García

Male, 32 years old | Professor of Politics



RELEVANT DATA Doctor with Honors. Has ADHD (Attention Deficit Hyperactivity Disorder)

TECHNOLOGICAL PROFILE He uses his computer and mobile phone independently and effectively. He has an old, slow computer.

GOALS Teaching the main theories to first-year students.

CHALLENGES He has to prepare the material because it changes every year.

OBSTACLES It is difficult for him to focus on a single issue.

First time student – Louise Grant

Female, 19 years old | Student of Literature



RELEVANT FACTS It's her first year at university; she's a little out of place.

TECHNOLOGICAL PROFILE She uses her mobile phone well but hardly knows how to use a computer, and she doesn't have one at home.

GOALS Getting her degree and becoming a teacher.

CHALLENGES She has to do much practical work.

OBSTACLES She receives numerous notifications, emails, and notes, and doesn't have time to keep up. She goes slow with the keyboard.

Visiting student - Said Ahman

Male, 23 years old | Student of Civil Engineering



RELEVANT FACTS An exchange student, he is learning the local language.

TECHNOLOGICAL PROFILE He can use his computer and mobile phone perfectly with autonomy. However, his computer's keyboard is not adapted to the local language, so he misses some important keys.

GOALS Getting to know the local culture.

CHALLENGES He has to pass at least four subjects to renew the scholarship for the next year.

OBSTACLES He barely understands the language and knows no one. The class videos do not have subtitles, making it difficult for him to follow the explanations.

Usual student – Liz Partel

Female, 23 years old | Student of Economics



RELEVANT FACTS	100% hearing loss. She can read lips and uses sign language.
TECHNOLOGICAL PROFILE	She uses her computer and mobile phone perfectly with autonomy. It has apps that transcribe text.
GOALS	Finishing university and doing an internship in a company.
CHALLENGES	Understanding complex economic and statistics theories.
OBSTACLES	Books have an overly complex language. The class videos do not have subtitles, making it difficult for him to follow the explanations.

Student as second activity – Emily Trent

Female, 45 years old | Student of Pedagogy



RELEVANT FACTS	She works part-time and spends the rest of the time raising her two daughters, aged 7 and 5.
TECHNOLOGICAL PROFILE	She uses her mobile phone perfectly with autonomy but does not have a computer at home.
GOALS	Learning how to manage her daughters.
CHALLENGES	Making time to study and do homework.
OBSTACLES	She receives a lot of messages from forums that are not interesting for her, and she needs to be told clearly what to do.

College Staff - Eric del Valle

Male, 43 years old | Administrative Assistant



RELEVANT FACTS	He has just become a father; he sleeps little.
TECHNOLOGICAL PROFILE	He only uses the computer throughout the day.
GOALS	Processing and verifying student tuition and payments.
CHALLENGES	Learning how to use the new intranet.
OBSTACLES	When the intranet was implanted, he was on parental leave, and now he has to learn on his own through the manual.

The WCAG Guidelines

In this chapter, we discuss the well-known recommendations from the W3C consortium, encompassing their historical evolution, structure, and documentation.

We end the chapter by explaining its usefulness and importance.

What are the WCAG Guidelines?

The Web Content Accessibility Guidelines (WCAG) include recommendations for making web content more accessible, especially to people with disabilities.

Published by the W3C, the world's leading internet standards organization, WCAG are currently the most recognized, followed, and required international guidelines for Web Accessibility, to the point of being integrated into most [national and regional legislation.](#)²⁵

WCAG [were born in its version 1.0](#) in 1999 with 14 guidelines and 65 checkpoints²⁶. In 2008, the W3C published WCAG 2.0, an evolution of the previous ones that had become obsolete due to technological advances. This latest version had 12 guidelines that grouped 61 success criteria with more than 500 associated techniques.

In 2018, the W3C approved WCAG 2.1, an evolution of the guidelines that addressed a greater diversity of technologies, such as mobile phone touch screens. It gave more significant support to groups with low vision, cognitive disabilities, or learning difficulties. WCAG 2.1 introduced a new guideline (*Guideline 2.5 Input Modalities*) and 17 new success criteria (5 Level A, 7 Level AA, and 5 Level AAA).

In 2023, the W3C approved WCAG 2.2, an evolution that continues the work of WCAG 2.1 and responds to its exact needs. This recent version adds nine success criteria (2 level A, 4 level AA, and 3 level AAA), and an obsolete criterion is eliminated.

²⁵ <https://www.w3.org/WAI/policies/>

²⁶ <https://www.w3.org/TR/WCAG10/>

Table 3. Comparative summary of the different versions of the WCAG

Year	Version	Guidelines	Organized in
1999	WCAG 1.0	14 guidelines	65 checkpoints
2008	WCAG 2.0	12 guidelines	61 success criteria
2018	WCAG 2.1	13 guidelines	78 success criteria (17 new criteria)
2023	WCAG 2.2	13 guidelines	86 success criteria (9 new criteria, 1 criterion removed)
?	WCAG 3.0 (1 st draft in 2021)	12 guidelines	46 groups of foundational requirements, supplemental requirements, and assertions ²⁷

In parallel, the first draft of WCAG 3.0 was published in January 2021 and has been updated several times since then. It should be noted that the development and publication of WCAG 3.0 will take years to complete and parallel the evolution of WCAG 2. Read the section “[A sneak peek into the future: WCAG 3](#)” at the end of this chapter to learn more about the evolution of the WCAG.

²⁷ According to the last draft published in December 2024.

What was new in WCAG 2.1

The 2018 WCAG 2.1 guidelines incorporated 17 new criteria:

- 1.3.4 Orientation (AA)
- 1.3.5 Identify Input Purpose (AA)
- 1.3.6 Identify Purpose (AAA)
- 1.4.10 Reflow (AA)
- 1.4.11 Non-Text Contrast (AA)
- 1.4.12 Text Spacing (AA)
- 1.4.13 Content on Hover or Focus (AA)
- 2.1.4 Character Key Shortcuts (A)
- 2.2.6 Timeouts (AAA)
- 2.3.3 Animation from Interactions (AAA)
- 2.5.1 Pointer Gestures (A)
- 2.5.2 Pointer Cancellation (A)
- 2.5.3 Label in Name (A)
- 2.5.4 Motion Actuation (A)
- 2.5.5 Target Size (AAA)
- 2.5.6 Concurrent Input Mechanisms (AAA)
- 4.1.3 Status Messages (AA)

What was new in WCAG 2.2

The 2023 WCAG 2.2 guidelines removed the 4.1.1 "Parsing" criteria and incorporated nine new ones:

- 2.4.11 Focus Not Obscured (Minimum) (AA)
- 2.4.12 Focus Not Obscured (Enhanced) (AAA)
- 2.4.13 Focus Appearance (AAA)
- 2.5.7 Dragging Movements (AA)
- 2.5.8 Target Size (Minimum) (AA)
- 3.2.6 Consistent help (A)
- 3.3.7 Redundant entry (A)
- 3.3.8 Accessible Authentication (Minimum) (AA)
- 3.3.9 Accessible Authentication (Enhanced) (AAA)

WCAG Guidelines 2 Documents

WCAG 2 consists of **five interconnected documents**.

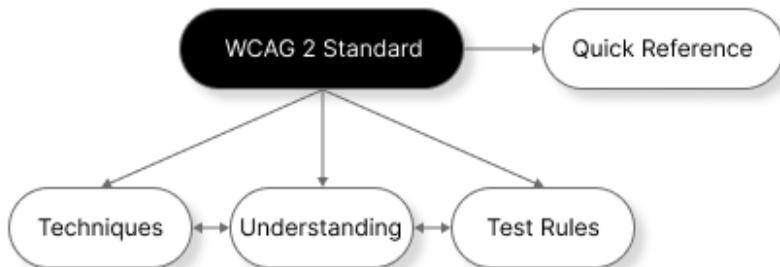


Illustration 6 WCAG 2 Composition

On the one hand, the "**Guideline**" itself, which is normative and does not undergo changes.

On the other hand, four informative documents that are updated from time to time:

- **How to Meet WCAG 2:** This is a quick reference guide and the primary tool that includes requirements (success criteria and techniques).
- **WCAG 2 techniques:** Provide specific information, such as snippets of accessible HTML code. These techniques are informative, they are not mandatory. WCAG 2 compliance is based on compliance criteria, not techniques.
- **Understanding WCAG 2:** This is an additional reference for learning how to implement WCAG 2 and is intended for people who want to understand the guidelines and compliance criteria in depth.
- The **Accessibility Conformance Testing (ACT) Rules** outline the methods for assessing compliance with WCAG success criteria. Their primary purpose is to aid in the creation of evaluation tools and testing strategies.

In this book, we have simplified the organization and texts of these documents. To conduct an audit, reference and always use the original documentation.

How WCAG 2.2 are organized

WCAG 2.2 are organized into four levels:

4 principles

They are the basis of Web Accessibility: every website must be perceivable, operable, understandable, and robust. Principles are not assessable.

13 guidelines

The principles are divided into guidelines that logically group the success criteria. The guidelines are not assessable.

86 success criteria

The success criteria are **assessable**. Depending on several factors that will be discussed later, they are categorized into three levels: A (the lowest), AA (the middle level), and AAA (the highest).

Around 600 techniques to follow and mistakes to avoid

Each success criterion proposes a series of techniques to follow and documents a list of errors to avoid to achieve compliance. Additionally, each method and error has its test procedures.

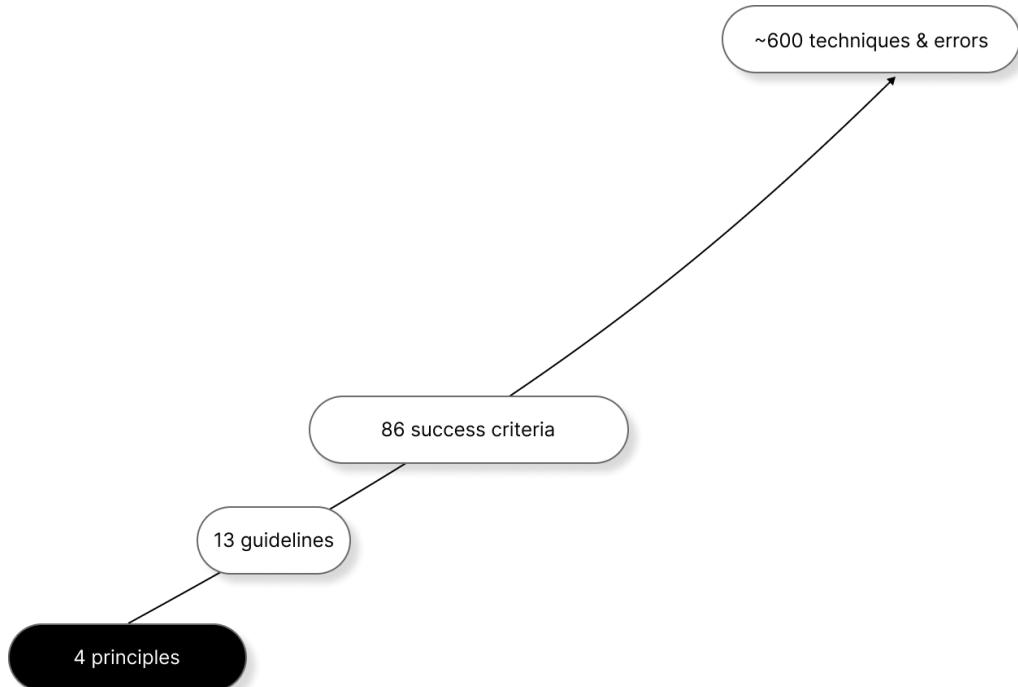


Illustration 7 WCAG 2.2 Organization

The Four Principles and Their Guidelines

WCAG 2.2 are organized in principles, and these in guidelines. The objective is to overcome the four principles, for which we must comply with all their guidelines.

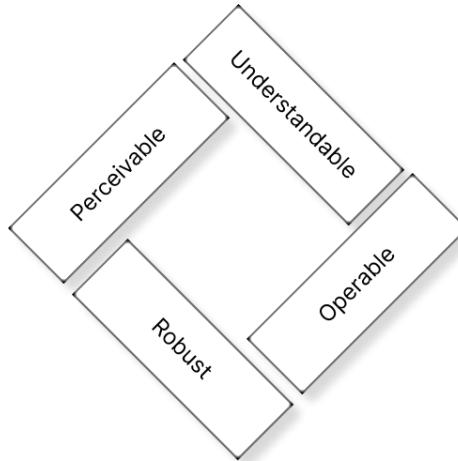


Illustration 8 The Four Principles

The website must be "Perceivable"

Our website can be visited by people with different preferences and abilities and by robots (search engines, translators...). Our information and interface must take this need into account and, therefore, we must offer alternatives to people who cannot use any of their senses. The guidelines of this principle are:

Guideline 1.1 *Text Alternatives to non-text content*

Guideline 1.2 *Alternatives to Time-based media, i.e., Video and Audio*

Guideline 1.3 *Content is adaptable to different forms of presentation*

Guideline 1.4 *Distinguishable: Content is easy to see and hear*

The website must be "Operable"

UX/UI designers and programmers must provide interaction and navigation elements that people with different abilities can handle. The guidelines of this principle are:

Guideline 2.1 *Keyboard-accessible functionality*

Guideline 2.2 *People have enough time to read and use the content*

Guideline 2.3 *Content does not cause seizures or physical reactions*

Guideline 2.4 *Navigable: People can browse, find content, and know where they are at all times*

Guideline 2.5 *Support multiple input modalities*

The website must be "Understandable"

If the user accessing our website doesn't understand what we're talking about or we make them feel lost, we have a problem. The guidelines of this principle are:

Guideline 3.1 Readable: Content is easy to read and understand

Guideline 3.2 Content appears and is handled in a *predictable* way

Guideline 3.3 Input assistance: Help in data entry to avoid and correct errors

The website must be "Robust"

This is the principle most dependent on technology. It is based on the website's ability to be transmitted and interpreted by different user agents.

User agents are all programs that display internet content, such as browsers (Chrome, Firefox, Edge, Safari, Internet Explorer...), media players (QuickTime, Real Player, Windows Media Player...), plugins (Java...), and other programs and devices, such as assistive products.

Assistive **products**, also known as *technical aids*, are programs or devices that provide functionality beyond the commonly used applications to meet the needs of people with disabilities. They support products such as a screen magnifier, an on-screen keyboard, voice recognition software, and a screen reader.

Our website must interact with this technology and their future versions to make the content accessible even if technologies and user agents evolve.

The only guideline on this principle is:

Guideline 4.1 Content is *compatible* with current and future user tools

The success criteria

As we mentioned earlier, the guidelines encompass the success criteria, which are the specific technical requirements that must be met when creating a website. These requirements are stated in a way that can be checked to see whether they are met.

The criteria are classified into three levels according to a series of variables.

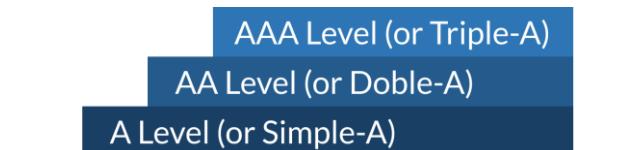


Illustration 9 Levels of conformance criteria

- **A (or Simple-A)**: the lowest.
- **AA (or Double-A)**: medium level.
- **AAA (or Triple-A)**: the highest.

To decide the level that each criterion should have, the W3C assesses:

- **Essentiality**: If this criterion is not met, can user agents (including assistive products) reproduce the website or not?
- **Universality**: Is this compliance criterion applicable to all websites, regardless of their theme, type of content, technology used, etc.?
- **Learning**: Can content creators learn in under a week to meet this compliance criterion?
- **Freedom**: How would this criterion of conformance limit the appearance, functionality, form of presentation, freedom of expression, design, and aesthetics of the page?
- **Alternatives**: Is there another way to access the content if the compliance criterion is unmet?

All compliance criteria are essential: the lowest ensure that the content of the web pages is accessible, and the higher ones help to improve their usability and reach a more substantial number of people.

When websites are evaluated, each criteria is reviewed. The next chapter explains how to do this.



At the end of the book, you will find all the conformance criteria organized by level.

Techniques and mistakes

Each success criterion *recommends* several known, documented, and widely used techniques. Each technique has a test procedure to determine whether we apply it correctly.

Note that we have used the verb "**recommend**" because no technique is mandatory since WCAG 2.2 allows you to create alternative mechanisms to overcome the criterion. In other words, apart from the methods documented by the W3C, it admits that there may be other ways of meeting the conformance criteria, which will be accepted as long as:

- Enable the objectives and *raison d'être* of the success criterion to be achieved;
- They meet – or do not fail – the rest of the criteria.

There are two categories of **techniques**: **sufficient** and **advisable**. The former helps to exceed or meet the criterion. The second are recommendations to improve the website's accessibility and usability.

The techniques can be general or refer to a specific technology (HTML, CSS, SMIL, ARIA, PDF...).

The **errors** documented by WCAG 2.2 are practices that hinder or prevent page accessibility. Like techniques, they have testing procedures to verify that we are not running into the same problems again.

Both techniques and errors can be associated with more than one success criterion.



When explaining each criterion in this book, we identify:

- **sufficient** techniques, as "Techniques to Meet the Criterion,"
- **advisable** techniques, as "Techniques to Improve Usability," and
- **mistakes** to avoid, as "Don't do it!"

What technologies do WCAG apply to?

WCAG 2.2 is technology-neutral, meaning you can build a website with any accessibility-compatible technology.

A technology, such as an HTML page or PDF document, is considered **accessibility-compatible**, when people using the assistive products can work with that technology; the accessibility features of the significant and mainstream technologies work with that technology. The key is how technologies include the necessary mechanisms so that user agents and assistive products can understand and handle them; that is, they can access the content, present it to the user, and the user perceive and manipulate it. It may even be necessary to modify user agents or supporting products to support that technology, for example, by downloading a *plugin*, as was the case with Adobe Flash.

Technological dependence refers to the fact that when a web page uses a specific technology to display content, if the user agent does not recognize or support that technology or if it is disabled, the content will no longer be compliant. For example, a SPA (*Single Page Application*) is a web page whose content is dynamically modified by JavaScript asynchronously, often using frameworks such as Angular or libraries with React. These applications can be accessible thanks mainly to the WAI-ARIA standard, which is why JavaScript is considered a technology that supports accessibility, but it creates a technological dependency.

While the W3C takes a neutral stance on technology, it does not take a neutral stance on its use: **how we use technology determines whether technology is accessible**. Therefore, there is no official list of technologies that support accessibility, but HTML, PDF, or JavaScript are considered to be the best. Using these technologies in an "accessibility-compatible way" means that they can be used seamlessly by assistive products, such as a screen reader. The [chapter "WAI-ARIA"](#) explains how to do it in the case of JavaScript, and in the chapter "[Accessible PDF documents](#)", how to do it in the case of PDFs.

Suppose we use technologies that don't support accessibility, such as the HTML "canvas" element, or we use accessibility-enabled technologies, such as PDF, HTML, or SVG, but use them unsupported. In that case, we'll need to provide an alternate version.

- For example, if we include a non-accessible PDF, we must provide an accessible alternative in accessible HTML.
- But if our PDF is natively accessible, we don't need to include an accessible HTML version.

Why following WCAG 2 and upgrading to 2.2

Here we propose several benefits of the current guidelines:

1. Because they are the official recommendation of the **W3C**, the world's most recognized web standards development organization.
2. They are the outcome from many **years of work** and the consensus of an international community of accessibility specialists.
3. Because WCAG 2 have been **recognized by the leading standardization organizations**, such as ISO at the international level, the CEN/ETSI at the European level, creating the standards [ISO 40500](#)²⁸ (WCAG 2.0), [EN 301 549](#)²⁹, (WCAG 2.1) respectively. It's a matter of time before they upgrade to 2.2.
4. Because WCAG 2.1 are **legally mandatory** in many countries, such as the member states of the [European Union](#)³⁰ or the [USA](#)³¹, while others like [UK](#)³² have already adopted WCAG 2.2. The W3C-WAI site provides an overview of [web accessibility legislation worldwide](#).³³
5. Because they are **summative**: complying with WCAG 2.2 also complies with WCAG 2.1 and WCAG 2.0. Since this update is cumulative, it does not alter the previous criteria.
6. Because its compliance **can be verified**, it has its **official evaluation methodology**, which perfectly applies to WCAG 2.2. This methodology is explained in the chapter "[Assessing Accessibility with WCAG-EM](#)".
7. Because some **automatic validators** and tools facilitate evaluation. WCAG 2.0 or WCAG 2.1 validators are compatible with WCAG 2.2.
8. They work with all web technologies and are designed to remain current and not become obsolete.
9. Because they have **clear rules** for assessing the conformance of a page.
10. Because **they show** who is interested in accessibility.
11. Because WCAG 2.2 **improve usability** for people who access through mobile devices and takes more into account the needs of people with low vision and cognitive disabilities.

²⁸ <https://www.iso.org/standard/58625.html>

²⁹ https://accessible-eu-centre.ec.europa.eu/content-corner/digital-library/en-3015492021-accessibility-requirements-ict-products-and-services_en

³⁰ <https://digital-strategy.ec.europa.eu/en/policies/web-accessibility>

³¹ <https://www.ada.gov/resources/2024-03-08-web-rule/>

³² <https://www.gov.uk/guidance/accessibility-requirements-for-public-sector-websites-and-apps>

³³ <https://www.w3.org/WAI/policies/>

A sneak peek into the future: WCAG 3

The next version of the WCAG is currently a draft and is not expected to be a completed W3C standard for a few more years. The final draft released before the publication of this book was in December 2024. Significant changes may occur when the definitive version is published, so always verify the official recommendations.

Like WCAG 1 and WCAG 2, WCAG 3 aims to make the web more accessible to people with disabilities, especially those with **cognitive disabilities**.

Another novelty of WCAG 3 is flexibility in addressing diverse types of web content, apps, tools, publishing, and emerging technologies on the web and in organizations. Even the name of WCAG 3 is different from the previous version:

- WCAG 1 & 2 is Web Content Accessibility Guidelines
- WCAG 3 is W3C Accessibility Guidelines

Finally, WCAG 3 tries to be **easier to understand**. The W3C is conscious of the complexity of WCAG 2 and wants to improve the popularity of its recommendation.

WCAG 3 is **similar** to WCAG 2 in providing fundamental and specific accessibility requirements, but it is grouped in a different way and differs in the structure, conformance model, and evaluation.

Regarding the **structure**, the last WCAG 3 draft has 12 “**Guidelines**” that are more granular than the guidelines in WCAG 2.

- 1) Image and media alternatives
- 2) Text and wording
- 3) Interactive components
- 4) Input/operation
- 5) Error handling
- 6) Animation and movement
- 7) Layout
- 8) Consistency across views
- 9) Process and task completion
- 10) Policy and protection
- 11) Help and feedback
- 12) User control

Each guideline has foundational requirements, supplemental requirements, and assertions; all are normative. These are some examples for the “Image and media alternatives” guideline:

- **Foundational requirement:** *Decorative image is programmatically hidden.*
- **Supplemental requirement:** *The role and importance of the image is programmatically indicated.*
- **Assertion:** Text alternatives follow an organizational style guide.

Another significant difference is the **Conformance Model**, which is intended to be more flexible. The most basic level of conformance will require meeting all of the foundational requirements (somewhat comparable to WCAG 2.2 Level AA). Higher levels of conformance will be achieved through the use of supplemental requirements and assertions.

Instead of requiring perfect accessibility, percentages or points would be used to assess compliance. Content accessibility would be categorized as:

- bronze (minimum),
- silver (improved), or
- gold (exemplary role models).

Finally, WCAG 3 proposes two ways of **evaluating**: **quantifiable tests** (where there is a high degree of reliability between test results from different testers) and **qualitative tests** (results may vary between testers who reviews the criteria).

So, as you can see, there are numerous changes compared to WCAG 2. The new standard will introduce significant changes in various aspects, including philosophy, scope, structure, content, and compliance model. Currently, WCAG 3 is a work in progress that will undergo significant changes before being released, so we will have to wait for its final publication before applying it. In the meantime, we must apply WCAG 2.2, which is validated and approved.



You can read the [last draft of the WCAG 3](#)³⁴ on the official W3C site.

³⁴ <https://www.w3.org/TR/wcag-3.0/>

How to make an accessible website

This chapter outlines the minimum requirements a website or web page must meet to be considered compliant to accessibility.

Implementing WCAG is the foundation of accessibility but not its ceiling. We can always do more than the guidelines to achieve greater accessibility.

In addition, we explain how to communicate the level of accessibility achieved through declarations of conformance.

The 5 requirements of Web Accessibility

Five mandatory requirements must be met for every website to be considered accessible or "compliant" with WCAG 2.2.

1) Achieving one of the three levels of conformance

The success criteria are categorized into A, AA, and AAA. The following section provides an in-depth explanation.

2) Applying to full web pages

WCAG 2.2 considers a web page as a whole, including content, functionalities, structure, styles, videos, subtitles, sounds, etc. Conformance, *partial conformance*, or *non-conformance* refers to the **entire website**.

Different page variations, such as those automatically generated for various screen sizes and resolutions, are also part of that page and must all conform.

Similarly, suppose the website has information or functionality that is not accessible but has an accessible alternative version linked to it, which can be understood and operated by user agents and supporting products. In that case, that alternative version is considered part of the page. Then, we can consider the page to be accessible.

3) Applying to complete processes

If the website is part of a process (for example, a purchase form divided into several pages), we must consider the accessibility of the **entire process**. We cannot claim that the order confirmation page is accessible when the previous order request page is not.

4) Using accessibility-compatible technologies.

Alternative versions are not needed if the technology is natively accessible. Remember that **we only check pages** as a whole, not technologies.

5) Not having blocking problems or interferences

We must ensure that certain content **does not prevent access to the rest of the page**. Therefore, even if they have an alternative, these four compliance criteria must always be met:

- 1.4.2 Audio control (A).
- 2.1.2 No keyboard Trap (A).
- 2.2.2 Pause, stop, hide (A).
- 2.3.1 Three Flashes or Below Threshold (A).

Web page compliance levels

Depending on the compliance criteria passed and their levels, we can certify that a website has reached one of these three levels:

- **Level A:** The page (or its alternative) meets all 31 Level A success criteria.
- **Level AA:** The page (or its alternative) meets 55 success criteria:
31 Level A + 24 Level AA.
- **Level AAA:** The page (or its alternative) meets all 86 success criteria:
31 Level A + 24 Level AA + 31 Level AAA.

Remember that the lowest level (level A) makes the content easier to access, and the highest level (level AAA) improves its usability and enlarge the number of people who can access it.

The decision to take our pages to one level or another is made for the following reasons:

- **Suitability for the target audience:** if our website is going to be used by people with functional diversity, it is logical and necessary that it is adapted to their characteristics. The higher the level, the better we will please our audience.
- **Legal obligation:** For example, in Europe, the USA, and many other countries, the minimum level required is AA.
- **Political and personal decisions:** In many organizations, sensitivity to social issues involves achieving higher levels of accessibility than those legally required.
- **Available resources:** Creating or improving a website entails a series of human, technical, and economic resources that not all organizations can afford. Staff training, the tools at our disposal, and the time needed are crucial elements that determine how far we can go. These limitations should not be used as an excuse but as indicators to improve our processes and increase the accessibility of the sites we create.

As seen in the previous sections, the key to achieving an elevated level of accessibility is training the different profiles involved in the site's development and maintenance and the correct planning of the computer systems, design, programming, and content departments.

Usually, it is much **more complicated to repair websites to make them accessible than to create them accessible from scratch with good planning**. Therefore, preventing and identifying accessibility issues early significantly reduces development costs and time. To achieve this, we must integrate accessibility into all phases of development, considering the needs of people with disabilities from the project's inception, not just at its conclusion.

Declarations of conformance

Once you've verified that your website is accessible, you likely want to communicate that you care about it (or have a legal obligation to do so).

The declaration of conformance indicates the level of accessibility and other relevant information and is usually included in an "Accessibility" section in the webpage's footer, the help menu, the 'about' page, or the sitemap.

There are several ways to declare conformance. Conformance claims are more technical and specific, focusing on compliance with standards, while accessibility statements are broader and more user-oriented, highlighting the organization's overall commitment to accessibility.

We have included a section explaining each one with examples and templates.

- The first way is through **the technical WCAG Conformance Claims** and **W3C Accessibility Statements**.
- In **European Union member states**, public sector websites and mobile applications (or those that receive public funding) must include an **accessibility statement** with a particular structure and content specified by the European Commission.
- In the **USA**, **federal agencies** must also implement an **accessibility statement**, but in a different format, following Section 508.
- For the **private sector**, some countries such as **the USA** use the **ACR (Accessibility Conformance Report)** and the **VPAT® (Voluntary Product Accessibility Template)** declarations of conformance.

If an assessment has been performed using the WCAG-EM methodology, we explain how to report it in the next chapter.

WCAG Conformance Claims

Conformance claims outline a website's adherence to specific standards.

On 18 April 2025, all content available on the server at <http://www.example.com> conforms to Web Content Accessibility Guidelines 2.2 at <https://www.w3.org/TR/WCAG22/>.
Single-A conformance.

- The technology that this content "**relies upon**" is: HTML 5.
- The technologies that this content "**uses but does not rely upon**" are: CSS3, and WEBP.
- This content was tested using the following user agents and assistive technologies:
 - Firefox X.X on Windows 10 with Screenreader X
 - Chrome X.X on Windows 11 with Screenreader Y
 - Edge X.X on Windows 10 with Screenreader Z
 - Safari X.X with OS X.X with Screenreader W

Illustration 10 Example of a WCAG Conformance Claim

WCAG conformance claims include mandatory and optional information, in natural text format and optionally with [machine-readable metadata](#).³⁵

Table 4 Information to be included in the WCAG 2.2 Conformance Claims

Required information	Optional Information
<ul style="list-style-type: none">- Date of the declaration of conformance.- The guidelines have been followed: the title, the version, and the URI.- The level of conformance achieved: A, AA, or AAA.- The scope of the declaration, i.e., the pages that comply with the conformance. The next examples explain how to list pages.- The web content technologies relied upon are listed on the declaration or on a separate page.	<ul style="list-style-type: none">- Success criteria met above the declared level: "<i>The following conformance criteria are also met ... level...</i>"- Web content technologies used but not relied upon.- User agents and supporting products that pages have been tested against: "[browser+version] on [operating system+version] with [plug-in+version] and with [supporting product+version]."- Steps taken beyond what is indicated by WCAG 2.2, e.g., usability testing with people with disabilities.- List of content-specific accessibility features.

³⁵ <https://www.w3.org/WAI/WCAG22/Understanding/understanding-metadata>

There are different ways to specify the scope of the claim:

Table 5 Templates for including the scope in the WCAG Conformance Claims

Entire site	Single page	Multiple pages ³⁶
"On [date], all web pages on [domain url] comply with Web Content Accessibility Guidelines 2.2 (https://www.w3.org/TR/WCAG22) Conformance Level [A, AA, AAA]. "	"On [date], the web page [page title] on [page URL] complies with Web Content Accessibility Guidelines 2.2 (https://www.w3.org/TR/WCAG22) Conformance Level [A, AA, AAA]. "	"On [date], the webpages listed below comply with Web Content Accessibility Guidelines 2.2 (https://www.w3.org/TR/WCAG22) Conformance Level [A, AA, AAA]: - [page title 1] on [page URL 1] - ... - [page title n] on [page URL n]."

If the **web page is not static** and changes without the organization's control (e.g., comments on a blog, dynamically inserted advertising, or a social network feed widget), it is virtually impossible to review if the added content remains accessible continuously. To address this problem, WCAG 2.2 offers two workable solutions:

- If you **monitor and fix** errors in your website's external content within two business days, you can use the standard declaration of conformance.
- If you can't monitor and fix errors, you can use a **partial declaration of conformance for content from uncontrolled sources**. You may only use this statement if that content is not really under your control. The way to do this is to add a detailed list of contents to the declaration:

"This page does not comply with WCAG 2.2 level [A, AA, AAA], but may comply with it if the following content from uncontrolled sources were removed: ..."

Another option contemplated by WCAG 2.2 is the **declaration of partial conformance due to language**. This statement may be used when the technology does not allow accessible use with the language (or languages) used on the page. For example, imagine you have a website with English and Chinese versions. However, due to technological limitations, the *alt* attributes of the images are only displayed in English, so people accessing the Chinese version cannot understand the description of the pictures. The way to indicate this partial conformance would be:

"This page does not conform, but would conform to WCAG 2.2 at level X if accessibility support existed for the following language(s):"

³⁶ If it's only applicable to multiple pages on the site, we recommend simply listing the pages, but you can also use regular expressions or Boolean logic.

Finally, if you want to include the WCAG conformance logo, it must be linked to the Conformance Claim of that page.



Illustration 11 Conformance logos according to levels



Learn more about [WCAG Conformance Claims](https://www.w3.org/WAI/WCAG21/Understanding/conformance)³⁷.

Download the [WCAG Conformance Logos](https://www.w3.org/WAI/standards-guidelines/wcag/conformance-logos/)³⁸

³⁷ <https://www.w3.org/WAI/WCAG21/Understanding/conformance>

³⁸ <https://www.w3.org/WAI/standards-guidelines/wcag/conformance-logos/>

WCAG 2.2 Conformance Claim Template

Compliance (Required)

Date of conformance	[mm/yyyy]
Guidelines	WCAG
Version	2.2
URI	https://www.w3.org/TR/WCAG22/
Level of Compliance	[A AA AAA]
Higher conformance criteria achieved	(Optional)
Other actions that allow greater accessibility	(Optional)

Scope of the declaration (Required)

Title	URL
...	..
...	..

Technologies used (Required)

Technology	Version	Rely Upon?
HTML	...	[Yes No]
ARIA	...	[Yes No]
CSS	...	[Yes No]
JPG	...	[Yes No]
PNG	...	[Yes No]
GIF	...	[Yes No]
WebP		
SVG	...	[Yes No]
MP3	...	[Yes No]
AVI	...	[Yes No]
JavaScript	...	[Yes No]
PDF	...	[Yes No]
Java	...	[Yes No]
...	...	[Yes No]

User-Agents Used in Testing (Optional)

Browser	Version	Operating system	Plug-ins	Assistive Products
Chrome
Edge
...

W3C Accessibility Statements

Accessibility statements (in general) are mainly intended for users. Typically, they will consult these statements when they face any issues. Writing in plain language and providing helpful information is crucial rather than using technical or legal terms.

Accessibility Statement for Itákora Website

This is an accessibility statement from Itákora.

Measures to support accessibility

Itákora takes the following measures to ensure accessibility of Itákora Website:

- Include accessibility as part of our mission statement.

Conformance status

The [Web Content Accessibility Guidelines \(WCAG\)](#) defines requirements for designers and developers to improve accessibility for people with disabilities. It defines three levels of conformance: Level A, Level AA, and Level AAA. Itákora Website is partially conformant with WCAG 2.2 level AA. Partially conformant means that some parts of the content do not fully conform to the accessibility standard.

Feedback

We welcome your feedback on the accessibility of Itákora Website. Please let us know if you encounter accessibility barriers on Itákora Website:

- E-mail: info@itakora.com

We try to respond to feedback within 7 business days.

Date

This statement was created on 17 April 2025 using the [W3C Accessibility Statement Generator Tool](#).

Illustration 12 Example of a minimum Accessibility Statement

WC3 Accessibility Statements include mandatory and optional information.

Table 6 Information to be included in the W3C Accessibility Statement

Required information	Optional Information
<ul style="list-style-type: none">- Identification- A commitment to accessibility- The accessibility standard applied- Contact information- Creation Date	<ul style="list-style-type: none">- Any known limitations- Measures taken to ensure accessibility- Technical prerequisites- Environments in which the content has been tested- References to applicable laws

W3C Accessibility Statement Generator

The W3C has developed a [Generator of Accessibility Statements](#)³⁹, where website owners can input their data in a form and download the statement in HTML code.

About your statement Show Info

Name of your organization
(Example: "Example Inc.")

Address of your website or mobile application
(Example: "https://example.org")

Name of your website or mobile application
(Example: "Example Store" or "Example App 1.2.3")

Accessibility standards applied Show Info

Accessibility standard followed:

WCAG 2.1 level AA
 WCAG 2.0 level AA
 Other

Conformance status Show Info

Fully conformant: the content fully conforms to the accessibility standard without any exceptions
 Partially conformant: some parts of the content do not fully conform to the accessibility standard (you can indicate these parts in later sections of this form)
 Non conformant: the content does not conform the accessibility standard
 Not assessed: the content has not been evaluated or the evaluation results are not available
 None

³⁹ <https://www.w3.org/WAI/planning/statements/>

Accessibility Statements in the European Union

The European Commission, through [Implementing Decision \(EU\) 2018/1523](#)⁴⁰, published the accessibility statement model following Directive 2016/2102 for websites and mobile applications in the public sector or that receive public funding.



All websites and mobile applications in the public sector (or that receive public funding) in Europe must include an accessibility statement that follows this model.

The statement must meet the following **requirements**:

- It should be updated **at least once a year** or every time a review is made.
- It must be provided in **an accessible format**.
- Claims of conformance shall be accurate and based on an assessment by the organization or a third party indicating the method used.
- On **websites**, the declaration of conformance will be available in a section linked from all pages and called "Accessibility" or its equivalent in the language in which the page is located.
- In **native apps**, it will be on the entity's website along with the download link or provided when downloading it from application distribution platforms.

In addition, the statement must be structured in the following **sections**:

- **Compliance status:** indicates whether the website or mobile application is fully compliant, partially compliant, or non-compliant with regulations and legislation (the latest version, EN 301 549 v 3.2.1 includes the full text of WCAG 2.1 AA).
- **Non-accessible content:** any non-compliant aspect is indicated by reference to the specific requirements of the accessibility standard. It also specifies non-compliant content because the entity has invoked the disproportionate burden exception or because it is content that does not fall within the scope of the law, such as third-party content.
- **Preparation of the declaration:** the date of the declaration and the last revision are indicated, as well as the method used.
- **Comments and contact details:** This includes contact mechanisms to inform of any non-compliance with accessibility requirements and to request information and content in an accessible version, as well as contact details of those responsible for the site's accessibility and the processing of requests.

⁴⁰<https://eur-lex.europa.eu/legal-content/ES/TXT/?qid=1539938081477&uri=CELEX:32018D1523>

- **Application procedure:** includes information on the procedure for appealing a notification or application made that is considered unsatisfactory.
- **Optional content:** other information considered appropriate, such as additional measures to achieve a higher degree of accessibility than legally required; an additional assistance telephone number for people with disabilities or users of assistive products, etc.

Accessibility Statements in the USA federal agencies

The memorandum from the Office of Management and Budget (OMB) titled "[Strengthening Digital Accessibility and the Management of Section 508 of the Rehabilitation Act" \(M-24-08\)](#)⁴¹" mandates that federal agencies provide an accessibility statement on their websites.

Section 508 is a provision of the Rehabilitation Act of 1973 that requires federal agencies to ensure that their electronic and information technologies are accessible to people with disabilities. This includes websites, software, and hardware, ensuring that individuals with disabilities have the same access to information and services as those without disabilities. Section 508 establishes standards for accessibility, encouraging the design and development of technology that can be used by everyone, including those with various disabilities unless there is an undue burden.

Section 508 standards incorporate the WCAG 2.0 (Level A and AA Success Criteria), but organizations can include success criteria from 2.1 and 2.2.

The [statement must include the following information](#)⁴²:

1. The accessibility standard and level (e.g., WCAG 2.2 AA) applied to the website and any known limitations or alternative versions, as appropriate.
2. The contact information (name and email) for the Section 508 program manager.
3. A public feedback mechanism that allows members of the public to report accessibility problems.
4. Instructions for filing a complaint alleging a violation of Section 508.
5. Information about the agency's reasonable accommodations procedures for Federal employees and job applicants.
6. Instructions on the use of the telecommunications relay service.
7. Links to any relevant, publicly available organizational policies or procedures on digital accessibility.
8. Date that the digital accessibility statement was last updated or reviewed.

⁴¹ <https://www.section508.gov/blog/strengthening-digital-accessibility/>

⁴² <https://www.section508.gov/manage/laws-and-policies/website-accessibility-statement/>

An official website of the European Union How do you know? ▾

European Union English Search

Home Principles, countries, history Institutions, law, budget Priorities and actions Live, work, study News and events Contact the EU

Home > Accessibility Statement

Accessibility Statement

This statement applies to content published on the domain: "https://european-union.europa.eu/index_en"

It does not apply to other content or websites published on any of its subdomains. These websites and their content will have their own specific accessibility statement.

This website is managed by the European Commission on behalf of all the institutions of the European Union, under the supervision of the Inter-institutional Online Communication Committee (IOCC). It is designed to be used by as many people as possible, including persons with disabilities.

You should be able to:

- zoom up to 200% without problems

Screenshot 2 European Union's website accessibility statement

An official website of the United States government [Here's how you know](#) ▾

GSA U.S. General Services Administration MENU ▾

Home > Website Information > Accessibility statement

Accessibility statement

Website Information

- Overview
- A-Z index
- Accessibility statement**
- Digital strategy
- Report a website issue
- Web records
- Website policies

We design our web pages to meet or exceed the Section 508 standards, which are the technical requirements that ensure we're complying with [federal Section 508 law](#). We also conform to the [World Wide Web Consortium \(W3C\)](#) and their industry standard [Web Content Accessibility Guidelines \(WCAG\) 2.1](#). We meet Level AA standards, which means our content is accessible to most people in most circumstances. We continually modify our websites to ensure the information, features and content are accessible to persons with disabilities.

Our Section 508 program manager

[Charles Popelka](#) can answer any questions about our implementation of 508 policy or guidelines.

Available complaint processes

If you experience difficulty accessing any resources or content on this site, please [contact us via email](#) and include:

- The nature of your accessibility problem
- The URL of the page the inaccessible content was found on
- The preferred format in which you want to receive any materials
- Your contact information (email, phone or address)

Glossary ▾

Screenshot 3 USA's website accessibility statement

We suggest visiting the accessibility statements on the [European Union](#)⁴³ and the [USA government](#)⁴⁴ official websites to learn how the official models should be implemented.

⁴³ https://european-union.europa.eu/accessibility-statement_en
https://administracionelectronica.gob.es/pae/Home/pae_Estrategias/pae_Accesibilidad/implantacion-rd-1112-2018/declaracion_accesibilidad.html

⁴⁴ <https://www.gsa.gov/website-information/accessibility-statement>

ACR and VPAT® Declarations of Conformance

An ACR (Accessibility Conformance Report) is an accessibility conformance report for a product or service that suppliers submit in the USA when contracting with government agencies bound by *Section 508* of the Rehabilitation Act of 1973. Other countries with different legislation, such as the European Union, may also use it.

Governmental agencies ensure compliance during the procurement, development, maintenance, or use of ICT. However, it is becoming more common for procurement departments to request this report in the private sector, especially in business-to-business (B2B) transactions.

These reports are typically made using the Voluntary Product Accessibility Template (VPAT®), created by the Information Technology Industry Council (ITI), which is available in four editions:

- **WCAG Edition:** To report on WCAG 2 compliance.
- **508 Edition:** To report compliance with Section 508.
- **EU Edition:** To report compliance with EN 301 549.
- **INT Edition:** The international edition used to report compliance with the above three standards.

Each edition includes a different version of WCAG :

- WCAG 2.0 is incorporated into the 508 edition
- WCAG 2.1 is incorporated into the EU edition
- WCAG 2.2 is incorporated into the WCAG and INT editions

All four editions are provided as a Microsoft Word file that can be used as is or reproduced in other formats. They all contain:

- Instructions with the essential requirements.
- Best practices for providing consistent, uniform, and more complete reports.
- General fields to describe the product, the methods used, and other details.
- Tables with the requirements for each version.



You can download the [VPAT® models at the official ITI website](#).⁴⁵

⁴⁵ <https://www.itic.org/policy/accessibility/vpat>

Assessing Web Accessibility with WCAG-EM

In this chapter, we present the methodology for validating complete websites in a step-by-step approach.

We also demonstrate how to declare conformance and explain how to utilize a tool to apply it.

The WCAG-EM methodology

Different methodologies exist for evaluating the accessibility of an entire website. The most recommended currently is the W3C's WCAG-EM methodology, which is a reliable method for determining the level of accessibility of any website.

The **goals** of the WCAG-EM methodology are:

- Serving as a **guide** to evaluators.
- Promoting a series of **good practices**.
- Avoiding common mistakes.
- Ensuring that the results of different assessments from the same or different sites are **comparable and consistent**.

WCAG-EM Guide

*[Official documentation of the WCAG-EM methodology.](http://www.w3.org/TR/WCAG-EM/)*⁴⁶

⁴⁶ <http://www.w3.org/TR/WCAG-EM/>

Review Process

The WCAG-EM methodology consists of **5 steps**, some of which may overlap or be carried out in parallel:

1. Define the **scope** of the assessment.
2. **Explore** the website.
3. Select a representative **sample** of pages.
4. **Audit** the selected sample.
5. Record the **evaluation results** and prepare a report.

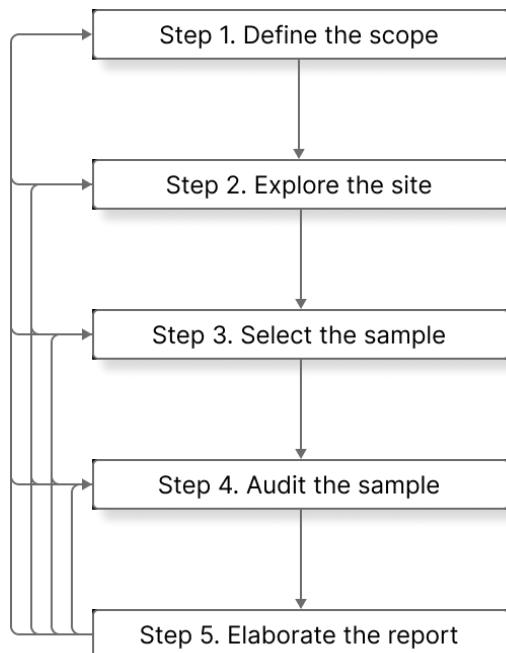


Illustration 13 WCAG-EM Methodology steps

Step 1. Define the scope of the assessment

The first step is to determine the scope of the evaluation unambiguously.

The methodology is always applied to a **complete website**: pages that serve the site's primary purpose and functionality. Therefore, part of the site's navigation, design, and complete processes cannot be excluded.

However, the evaluation can focus on areas that are separable from a website, such as the public and private sections or different versions of the website, including the mobile version and language versions.

In this phase, the following are also defined:

- The **Conformance Level** (A, AA, AAA) to be assessed.
- The **list of web browsers, assistive technologies, and other user agents** for which the features provided on the website are to be accessibility-supported. This list can vary depending on whether we analyze a public site or an intranet.
- Additional **evaluation requirements** agreed upon between the assessor and the client, such as the level of detail in the final report, unique use cases, and the involvement of people with disabilities.

Step 2. Explore Website

Once the scope has been defined, it is necessary to understand the site's use, purpose, and functionality to determine which pages to include in the sample for evaluation.

To do this, it is necessary to identify and include:

- The most relevant pages or page statuses⁴⁷:
 - home,
 - login,
 - sitemap,
 - contact,
 - help,
 - legal information,
 - and other pages heavily linked from many pages.
- Key **functionalities**, e.g., "selection and purchase of a product," "register an account," "fill out and submit a form," etc.
- Different **types of pages** and page statuses, for example:
 - created from different templates,
 - with different contents: forms, tables, multimedia, etc.
 - with different functional components: carousels, accordions, modal windows, etc.
 - that use different technologies: JavaScript, WAI-ARIA, PDF, etc.
- The **pages or statuses relevant to people with disabilities** or site accessibility, such as accessibility, help, or preferences pages.
- The **technologies used on which the content depends**, i.e., that the content would not be compliant if that technology were disconnected, or the browser did not support it. The technologies can be HTML, CSS, JavaScript, ARIA, SMIL, SVG, PDF, etc.

⁴⁷ Dynamically generated web pages can display different content, functionality, and appearance depending on the user, interaction, device, and other parameters. An example would be a website homepage that is signed in or unsigned in: there is a change to a part of the page, but they share the same URI.

Step 3. Select a representative sample of pages

Ideally, you should be able to evaluate the entire website, but in many cases, the volume of pages on the site makes it impossible to assess all of them manually.

Now that we have a deep understanding of the website, we can select a sample of pages that represent it so that the evaluation of that sample reflects the accessibility of the entire site with sufficient reliability.

The sample size will depend on the site's volume, complexity, or consistency. If we carefully select the pages and processes included in the sample, we will reduce the sample size, but it will still adequately represent the entire site.

Complete processes **must be included in the sample**. For example, if a checkout process has three steps, you can't include only the first step in the sample; you should include all the steps' pages.

Our sample consists of two types of pages.

On the one hand, a **sample of pages selected in a structured way**, in which we must include:

- Pages that are relevant to all people.
- Pages relevant specifically to people with disabilities.
- Pages with essential functionalities.
- Pages of diverse types.
- Pages that depend on different technologies.

On the other hand, a **control sample** which consists of a random selection of pages and page states within the defined scope, chosen without a predictable pattern. Its size must be 10% of the sample of selected pages; if the sample has 50 pages, we must randomly add five more as a control sample.

The control sample will be the indicator that allows us to verify the results and increase confidence in them.

Step 4. Audit the selected sample

It is time to verify whether the pages in the sample meet all the compliance criteria of the chosen level and the five compliance requirements of WCAG 2.2.

The evaluation of each page includes any content that contains, even if aggregated, embedded, or third-party content.

Each interface component must be evaluated **in different states** to verify that it meets the conformance criteria. For example, a text field in a form must be assessed in its initial state, when it receives the focus, when it has been typed in, when a validation error has occurred, and so on.

In addition, they should be checked with different browsers (Chrome, Firefox, Edge, Safari, etc.) and assistive products (e.g., NVDA, JAWS, VoiceOver, etc.; ZoomText screen magnifier; etc.) agreed upon during the scope definition in step 1.

When components **are standard to many pages**, such as the header or footer, they don't need to be evaluated repeatedly. Each can be analyzed independently, ensuring that the results are not included in the analysis of any other page.

On the other hand, **if content related to a particular compliance criterion is not present** (for example, there are no videos on the page and the criterion assesses whether the videos are subtitled), the criterion is considered "satisfied" or "not present".

Once the results are obtained, those of the selected sample are compared in a structured way with those of the control sample.

If the random sample contains contents or results not represented in the structured sample, the structured sample is not sufficiently representative. In this case, we must return to steps 2 and 3 to redefine the sample we selected in a structured way.



Some of these verifications can be done automatically with specific tools, while others must be manually checked by experts. Ideally, users with disabilities should explore the website to confirm its accessibility.

In the [Validation Tools chapter](#), you can find applications to help you perform the automatic and manual parts of the audit.

Step 5. Record the results and produce a report

To finally prepare a report, we must document the entire process, especially the evaluation results. Not all recorded data is (or can) be included in the report, for example, for confidentiality reasons.

At least the following must be documented:

- **About the assessment:** name of the assessor; the name of the stakeholder, company, or organization that requested the evaluation; and the date on which the examination was conducted.
- **Scope of the assessment** (step 1): scope, level of adequacy (A, AA, AAA) assessed, accessibility support defined, and additional requirements agreed upon.
- **Site Exploration** (step 2): Dependent technologies, and optionally the identified pages and technologies.
- **Representative sample** (step 3): list all the pages that make up the evaluated sample, differentiating the pages of the selected sample in a structured way and the pages of the control sample.
- **Sample audit** (step 4): The evaluation result for each sample page or whole sample. We must indicate whether or not the agreed level (A, AA, or AAA) is met, including at least one example for each criterion that is not met.
Depending on the level of detail agreed upon in Step 1, we can add more information, such as the detailed compliance for each criterion or all the errors found and the recommended solutions.

Even if it is internal, in step 5 it is recommended to **save all the details of the evaluation** (such as screenshots or the source code analyzed), as they can be instrumental in case of debate, improve accessibility, and later check these improvements: tools used, screenshots, data entered on the pages to replicate the results, etc.

Declaring conformance after following the WCAG-EM.

As we saw earlier, the WCAG 2.2 declaration of conformance cannot be made for entire websites based solely on the evaluation of a subset of pages, as it is always possible that conformance errors exist on other pages.

Although WCAG-EM minimizes this problem, we are not analyzing all the site pages and cannot produce a declaration of conformance for the entire website.

The resulting declaration of conformance must describe the results obtained after applying this methodology. It will include the same information defined as mandatory in the statement of conformance of WCAG 2.2.

We show two declarations of conformance to highlight the differences between the two versions:

Sample full-site declaration of conformance:

"On December 20th, 2024, all <http://example.com> pages are compliant with Web Content Accessibility Guidelines 2.2 (WCAG 2.2) at their AA level of conformance.

The set of accessibility-compatible technologies on which this statement depends is listed in <http://example.com/list.html>."

Example of a declaration of conformance after an audit based on the selection of a sample of pages:

"On December 20th, 2024, an accessibility audit of the <http://example.com> site has been conducted following the W3C WCAG-EM methodology.

The audit result satisfies the AA compliance level of the Web Content Accessibility Guidelines 2.2 (WCAG 2.2).

The set of accessibility-compatible technologies on which this statement depends is listed in <http://example.com/list.html>"

WCAG-EM Report Tool

For the audit process, the W3C has published the [WCAG-EM Report Tool](#)⁴⁸ that allows all the evaluation data to be recorded following the WCAG-EM methodology and generates a report automatically.

Although it operates online, the data can be saved locally and audited at a later time. In addition, the generated report can be downloaded in PDF and JSON format.

The screenshot shows the 'WCAG-EM Report Tool' interface. At the top, there are tabs: Overview, 1. Define Scope (which is active), 2. Explore Website, 3. Select Sample, 4. Audit Sample, 5. Report Findings, and View Report. The top right corner includes language settings (English, Nederlands, Hide translations) and the W3C logo with 'Web Accessibility Initiative - WAI'.

Step 1: Define the Evaluation Scope

Instructions: Define the overall parameters and scope of the evaluation. Ideally, this is done with the person who commissioned the evaluation (who may or may not be the website owner), to ensure common expectations about the scope of the evaluation.

Step 1: Define the Evaluation Scope

Website name: Show info

Scope of the website: Show info

WCAG Version: Show info (WCAG 2.2)

Conformance target: Show info (Level AA)

Accessibility support baseline: Show info

Additional evaluation requirements:

Your report Hide ▾
Reported on 0 of 55 WCAG 2.2 AA Success Criteria.

Perceivable	0 of 20
Operable	0 of 20
Understandable	0 of 13
Robust	0 of 2

View Report

Previous step Start Next step → Explore Website

Help improve this page
Please share your ideas, suggestions, or comments via e-mail to the publicly-archived list wai-eo-editors@w3.org or via GitHub.
[Email](#) [Fork & Edit on GitHub](#) [New GitHub Issue](#)

Screenshot 4 WCAG-EM Report Tool

⁴⁸ <https://www.w3.org/WAI/eval/report-tool>

Principle 1.

Perceivable

Our website can be visited by people with quite different needs and preferences and by robots (search engine spiders, automatic translators, etc.).

Our information and user interface components must take this into account.

Therefore, we must provide alternatives if people (and robots) cannot use their senses.

Guideline 1.1

Offer text alternatives

Provide text alternatives for any non-text content so that it can be changed into other forms that people need, such as large print, braille, speech, symbols, or simpler language.

Textual information can be represented visually, soundly, tactiley, or through a combination of these forms to suit people's needs. For example, a blind user can understand an image if offered a textual description, or a deaf user can understand a sound if presented with a text that describes it.

Non-Text Content

All non-text content presented to people has a textual alternative that serves the same purpose.

Exceptions:

- **Form controls** or mechanisms for the user to enter data: they have a name that describes their purpose.
- **Multimedia presentations** with temporal development: textual alternatives provide at least a descriptive identification of the non-text content.
- **Quizzes or exercises** that would not be valid if presented in text form: textual alternatives provide at least a descriptive identification of the non-text content.
- Content whose primary purpose is to offer a specific **sensory experience**: textual alternatives provide at least a descriptive identification of the non-text content.
- **CAPTCHA**, or analogous method of confirming that the user accessing the content is a user and not a machine: textual alternatives that identify and describe the purpose of the non-text content and alternative forms of verification are provided through different sensory perceptions.
- **Decorative content**, used to format the page or not displayed on the screen: Implemented in a way that assistive products can ignore.

Depending on the situation, a long text or short text alternative should be offered. Next, we explain when to use one or the other.



If you are unsure about which short or long description technique you should use, follow the steps in these resources:

- [Images Tutorial](#)⁴⁹
- [Decision Tree](#)⁵⁰

⁴⁹ <https://www.w3.org/WAI/tutorials/images/>

⁵⁰ <https://www.w3.org/WAI/tutorials/images/decision-tree/>

Short Text Alternatives

- Use the *alt* attribute in images.
- Use the *aria-label* or *aria-labelledby* attributes.
- Uses the *body* of the *object* element.
- Combine an image and adjacent text if they link to the same site.
- Offer a textual alternative to ASCII art drawings, emoticons, and *leet text*.
- In a group of related images, use the description of one of them to describe the group.

Long Text Alternatives

- Use the *aria-describedby* attribute in images.
- Use the *body* in the *object* element.
- Include a lengthy description in another location, with a link immediately adjacent to the non-text content.
- Include a long description near the non-text content, indicating in the short description the location of the long description.

Techniques to meet the criteria

If a short text can serve the same purpose and present the same information as non-text content:

- Use a short description alternative to include a description with the same purpose and information as non-text content.

If a short text can't serve the same purpose and present the same information as non-text content (e.g., a graph or diagram):

- Use a short description alternative to briefly describe the non-text content, plus a long description alternative.

If the non-text content is a control or form input field, identify its purpose with one of these techniques:

- Use the *aria-label* attribute to label objects like a button, region, or MathML function.
- Use the *aria-labelledby* attribute to concatenate a label from multiple text nodes, for example, in text fields included in a table.
- Use the *alt* attribute in *image map areas* and images used as buttons.
- Use the *label* attribute to associate labels with form fields; if you can't use it, use the *title* attribute.
- Provide link text that describes its purpose. For example, if the link includes an image with an *alt* attribute and text, the link text will be a combination of both.

If the non-text content is audio or video, recorded or live; is an exam or exercise that would not be valid if presented in text mode; or its primary intention is to create a sensory experience:

- Provide a descriptive tag using one of the short description techniques.

- For live audio and live-only videos, provide a short description alternative that describes their purpose.
- Provide a short description alternative, either the accepted name or a descriptive name for the non-text content.

If the non-text content is a CAPTCHA:

- Provide an alt text that describes your goal and offer another CAPTCHA that serves the same purpose using a different modality.

If assistive products should ignore images:

- Include decorative images using CSS; or
- Leave the *alt* attribute empty, and don't add the *title*.

Techniques to improve usability

- Use the CSS *margin* and *padding* attributes to format the spaces between elements instead of spacer images.

Don't do it!

- Don't use CSS to include images that contain essential information.
- Don't offer a textual alternative that doesn't include the information communicated by color differences in the image.
- Don't forget to update textual alternatives when non-text content changes.
- Don't use a textual alternative that isn't descriptive. For example, *alt="icon.jpg"* or *alt="image2"*.
- Don't mark decorative images in a way that allows supporting products to advertise them. For example, don't add *role="img"*, or a description like *alt="spacer"*.
- Don't omit the *alt* attribute in *img*, *area*, and *input* elements of type *image*, even if they are used only for decorating. Leave the attribute empty (*alt=""*).
- Don't provide long descriptions that don't convey the same purpose or present the same information as non-text content.
- Don't use characters that look like others without giving *alt* text. For example, the characters U+0063 and U+03F2 look like the letter "c," but the former is from the Western alphabet, and the latter is from the Greek alphabet, and text-to-speech tools don't process them in the same way.
- Don't use ASCII art without offering a textual alternative.

Guideline 1.2

Time-based media

Provide alternatives for time-based media.

This chapter will discuss video, audio content, and other interactive components in which time is essential to the sensory experience.

When we talk about **time-based media**, we refer to:

- audio only;
- video alone (understood as moving images);
- combined audio and video;
- audio and/or video combined with interaction mechanisms.

These time-based media can be both recorded and live.

Additionally, when we refer to **synchronized media**, we mean audio or video content synchronized with another format to present information and/or with time-based interactive components (except when it is an alternative multimedia content to a text). For example, a movie is considered "synchronized media" because it synchronizes audio and video. Also, an interactive game that presents visual and/or auditory content and whose interaction elements we use at certain times is considered "synchronized media."

Another concept that appears in this guideline is the **alternative document to the time-based media**, what we usually understand as transcription. A transcription is content that includes a correct sequence of textual descriptions of visual and/or auditory information and allows people who cannot see or hear to achieve the same results in any time-based interaction. For example, the script used to create synchronized media can accurately represent the resulting content if it describes both the visual (actions, people, on-screen text, body expressions, etc.) and the auditory (dialogue, music, laughter, applause, etc.) parts of the content.

On the next page, we show a summary table of the different means and alternatives according to their characteristics and the level of accessibility we want to reach. You may miss some cases, such as videos without live audio. Use common sense and the means available to offer the most suitable alternative.

Table 7 Overview of Time-based and Alternative Media

Media	Type	Level	Alternative	Criterion
Audio-only	Prerecorded	A	Descriptive transcript ⁵¹	1.2.1
	Live	AAA	Descriptive transcript	1.2.9
Video-only	Prerecorded	A	Descriptive transcript or Audio track with the same information	1.2.1
		AAA	Descriptive transcript	1.2.8
Synchronized media video+audio or video+audio +interaction or audio+interaction or video+interaction	Prerecorded	A	Captions + Audio description or Descriptive transcript	1.2.2 1.2.3
		AA	Captions + Audio description	1.2.5
		AAA	Captions + Audio description (extended if necessary) + Sign language interpretation + Descriptive transcript	1.2.6 1.2.7
	Live	AA	Captions	1.2.4



In addition, **all time-based media, whether live or prerecorded**, should have a title and/or description that allows people to decide if it's worth playing that content. (Criterion 1.1.1)

⁵¹ A *Descriptive Transcript* includes: synchronized text descriptions of visual and auditory elements; dialogue, sound effects, and important visual details (e.g., scene changes, gestures, or on-screen text); and interactivity details (if applicable, for interactive media). (<https://www.w3.org/WAI/media/av/transcripts/>)

Audio-only and Video-only (Prerecorded)

Audio-only (Recorded): An alternative document that presents information equivalent to its content is provided to the time-based media.

Video-only (Recorded): An alternative document or a soundtrack presents information equivalent to its content on the time-based media.

Exception:

- Audio or video is a text alternative and is clearly identified as such.

Techniques to meet the criteria

If the content is audio-only (recorded):

- Offer an alternative that presents the same information.

If the content is video only (recorded):

- Offer an alternative that presents the same information; or
- Provide audio that describes the most important content in the video.

Techniques to improve usability

- Use the HTML 5 track element to include a textual description in the *video element*.

Don't do it!

- Don't use a textual alternative that isn't descriptive (e.g., the file name).
- Don't offer long descriptions that don't serve the same purpose or present the same information as non-text content.

Captions (prerecorded audio synchronized with other media)

Captions are provided for all prerecorded audio content in synchronized media.

Exception:

- The synchronized media is a text alternative and is clearly identified as such.

There are several types of subtitles. Depending on their functionality, they can be:

- **General subtitles:** accompany the characters' dialogues, voiceovers and translated elements.
- **Captions:** Similar to general captions but they include information about sounds that happen ("background murmurs") or intonation ("shouting").

Depending on how they are incorporated into the image, they can be:

- **Open:** They are embedded in the video's image, so they cannot be altered or hidden except by placing other subtitles on top.
- **Closed:** available in an external file, an auxiliary program presents them together with the video. They can be changed in color, position, or size.

Techniques to meet the criteria

- Offer open or closed captions.
- Offer closed captioning using SMIL technology, the *HTML 5* video element track element, or another format supported by the player.

Don't do it!

- Don't skip important dialogue or sound effects in captions.
- Don't offer synchronized media without captions when that media adds additional information to the page.
- Don't forget to label the textual alternative for a synchronized media as the alternative for that media.

Audio Description or Media Alternative (Prerecorded)

Video prerecorded within a synchronized media content: An alternative document to the time-based media or an audio description of all the content is provided.

Exception:

- Synced media is a text alternative and is identified as such.

Audio description is a technique aimed at people with visual disabilities. It provides sound information that translates or explains visual content, such as a character's gestures, costumes, landscapes, etc., taking advantage of the silences between dialogues.

There is also the **extended audio description variant**, an audio description added to an audiovisual presentation by pausing the video so that you have enough time to add the additional information necessary to explain the content.

Techniques to meet the criteria

- Offer an alternative for time-based media, either with a link to the alternative next to the non-text content or by using the body of the *object element*.
- Add a second soundtrack, including the audio description, that the user accessing the content can select.
- Offer a version of the video with an audio description —or with an extended audio description— using SMIL or any player that supports audio and video.
- Offer static *alt* text to describe a video where only one user is shown speaking in an unchanging background, such as at a press conference.

Techniques to improve usability

- Add audio descriptions with the *HTML 5 video element track element*.

Captions (live)

Live audio on synchronized media: Captions are provided for all content.

Techniques to meet the criteria

- Offer open or closed captions; or
- Consider using SMIL technology or another format supported by the player to provide closed captioning.

Audio Description (Prerecorded)

Video prerecorded within synchronized media content: An audio description of all content is provided.

Techniques to meet the criteria

- Provide a second soundtrack, selectable by the user accessing the content, which includes an audio description.
- Offer a version of the video with an audio description—or an extended audio description—using SMIL or any player that supports audio and video.
- Offer static *alt* text to describe a video where only one user is shown speaking in an unchanging background, such as at a press conference.

Techniques to improve usability

- Add audio descriptions with the *HTML 5 video element track element*.

Sign Language (Prerecorded)

Audio prerecorded within synchronized multimedia content: Sign language interpretation is provided for all content.

There are diverse ways to show the sign language interpreter, either embedded within the video stream or on a different screen that can be placed in any position, even superimposed on the screen.



The [Able Player](#)⁵² can integrate sign language interpretation within the stream (first image) and on a different screen (second image). ⁵³



Illustration 14 Sign language integration in the Able Player

Techniques to meet the criteria

- Include a sign language interpreter within the video stream.
- Offer a synchronized video of the sign language interpreter that can be displayed on a different screen or overlaid on the image using SMIL.

⁵² <https://ableplayer.github.io/ableplayer/>

⁵³ This innovation was introduced in [the doctoral thesis "Non-Intrusive Accessibility in Audiovisual Communication on the Web"](#) by Emmanuel Gutiérrez y Restrepo. (<https://inclusiondigital.net/a11dnointrusiva/>)

Extended Audio Description (Prerecorded)

Video prerecorded within synchronized media: If the pauses in the foreground audio are insufficient to allow the audio description to communicate the meaning of the video, an extended audio description is provided for all content.

Remember that an extended audio description is an audio description added to an audiovisual presentation by pausing the video, which allows time to add additional information.

Techniques to meet the criteria

- Offer a video with an extended audio description using SMIL or any player that supports audio and video.

Techniques to improve usability

- Add audio descriptions with the *HTML 5 video element track element*.

Media Alternative (Prerecorded)

Prerecorded Video-only: An alternative document is provided to the time-based media.

Prerecorded synchronized media: An alternative document is provided to the time-based media.

Techniques to meet the criteria

If the content is synchronized (recorded) media:

- Offer an alternative for time-based media, either by linking to the alternative next to the non-text content or using the body of the *object* element.

If the content is video only (recorded):

- Offer an alternative that presents the same information.

Don't do it!

- Don't forget to label the textual alternative for a synchronized media as the alternative for that media.

Audio-only (Live)

Audio only (live): An *alternative document to the time-based media* that presents information equivalent to the content is provided.

Techniques to meet the criteria

- Offer a link to the verbatim transcription of the planned script or the script that was finally followed.
- Provide text-based alternatives for audio content only when broadcast live.
- Incorporate a live subtitling service for people who are deaf or hard of hearing on the website.

Guideline 1.3

Adaptable, show content in different presentations.

Create content that can be presented in different ways (e.g., with a more simpler layout) without losing information or structure.

The people who access your website have very different needs and preferences, so you have to check that all the information is available in a way that can be perceived by all people in various ways (when read aloud, when presented with a more straightforward visual design, etc.), with different user agents, and on other devices with varying screen orientations.

If information embedded in the page presentation cannot be differentiated or interpreted by the assistive products, it may not be presented in other formats as the user requires. For example, how are the areas of a page or its contents organized, or how are the content representations presented in a certain way? All information must be made identifiable and actionable using software to achieve this.

WCAG 2.2 defines **structure** as "how the pages that make up the site and the elements that make up a web page are organized and related to each other". It defines presentation as "the processing and delivery of content in a way that users can perceive".

Information and relationships

Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

Techniques to meet the criteria

If the technology provides a semantic structure for communicating information and relationships (e.g., HTML):

- Identify the zones and regions of the page (header, menu, main area and its sections, footer, etc.) using semantic HTML elements or ARIA roles.
- Use the *aria-labelledby* attribute to name zones and regions on a page and UI controls.
- Use semantic elements to mark up the structure or emphasized text. For example, using tags such as *em*, *strong*, *blockquote*, *cite*, *sup*, etc.
- Communicate with text the information transmitted by variations in the presentation of the text.
- Separate the information and structure from the presentation form to allow people to have different presentations, for example, by using their own CSS.
- Semantically identify an icon font with *role="img"*.
- Group related links using the *<nav>* element.
- Make the information and relationships conveyed through the presentation also present in the code by:
 - the *h1-h6* elements to identify headers, or the *role="heading"*;
 - the elements *OL*, *UL*, and *DL* for lists;
 - semantic markup when using color keys (e.g., if the text is colored in red to indicate that it is an error);
 - DOM functions to add content to the page programmatically.

If you need to present tabular data, use the *table element*, and add:

- the *caption element* to associate the table title with the table;
- the *scope* attribute, or the *ID* and *header attributes*, to associate header cells with data cells;

If you have a form, use:

- the *label element* for associating text labels with form controls;
- the *title attribute* to identify form controls when the *label element* cannot be used;
- the *fieldset* and *legend elements* to describe groups of form controls or grouping ARIA roles, such as *role="group"* and *role="radiogroup"*;

- The *OptGroup* element to group *option* elements within a *select* element.
- The *aria-labelledby* attribute to name UI controls, such as components of a form.

If the technology used does not offer a semantic structure to communicate information and its relationships:

- Communicate with text the information transmitted by variations in text presentation.
- Make the information and its relationships conveyed through presentation software determinable or available in text, using standard text formatting conventions for paragraphs, lists, and headings.

Techniques to improve usability

- Control the visual presentation of text with CSS.
- Locate labels to maximize the predictability of the relationships between labels and the controls they represent.
- Use the *aria-describedby* attribute to provide a more descriptive label to form controls, for example, by associating a statement to the field.
- Identify required form fields with the *aria-required* attribute.
- Organize the page using headings.

Don't do it!

- Don't use text presentation changes to convey information without proper markup or text. For example, don't simulate a heading using a paragraph with a particular style.
- Don't use whitespace characters (spaces, tabs, line breaks, or carriage returns) to format text in multiple columns or format content like a table.
- Don't use programming events to emulate bindings in a way that isn't programmatically determined.
- Don't use structure markup in a way that doesn't represent relationships to the content. For example, don't use a header tag solely to create a visual effect.
- Don't use the *th* or *caption* elements, or the *summary*, *headers*, or *scope* attributes, in the layout tables used to design the page.
- Don't use the *pre* element to mark tabular information.
- Don't insert informational content using the *:before* and *:after* selectors or the *content* property in CSS.
- Don't forget to mark the header cells in the tables of contents, and check that the association of the header cells with the content cells using the *headers* and *id* attributes has been done correctly.
- Don't use *role="presentation"* (or its equivalent *role="none"*) to mark up content with semantic information.

Meaningful sequence

When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined.

A correct reading sequence is any sequence where words and paragraphs are presented in an order that does not change the meaning of the content.

Techniques to meet the criteria

- Sort all the web page content in a meaningful sequence.
- Mark content sequences as meaningful using one of the following techniques and sort all content on the web page using a meaningful sequence.
 - Use Unicode markups to mix "right-to-left" and "left-to-right" text.
 - Use the *dir* attribute in inline items to solve problems with text that need different reading directions and are nested.
 - Position content based on structure markup.
 - Control the spacing between letters using the CSS *letter-spacing* attribute.
- Adjust the order of the DOM to the visual order.

Don't do it!

- Don't use whitespace characters (spaces, tabs, line breaks, or carriage return):
 - to format content as if it were a table;
 - to format text in multiple columns; or
 - to control the spacing within a word.
- Don't use a table to lay out content that doesn't make sense when the table is displayed linearly.
- Don't change the meaning of the content by how you place the information using CSS.

Sensory characteristics

Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.

Users of a screen reader will find it difficult to click on the "round, large, red button on the right of the screen" no matter how well it is described since they cannot differentiate it from the rest of the controls dictated by their support program.

For this reason, it is necessary to label and identify each control correctly and not only use the visual or auditory peculiarities of the elements on the screen.

Techniques to meet the criteria

- Offer textual identification of elements that would otherwise rely only on sensory input to be understood.

Don't do it!

- Don't identify content by its shape or location alone.
- Don't just use a graphic symbol to communicate information.

Orientation

Content does not restrict its view and operation to a single display orientation, such as portrait or landscape.

Exception:

- A specific orientation is essential to transmit the information or operate the content.

Exceptions can include a cashier's check, a piano app, virtual reality content, and a presentation slide. In these cases, a specific orientation is mandatory.

This criterion responds to the needs of people who have the screen in a fixed position, for example, anchored to a wheelchair. The goal is that the content can be enjoyed in any orientation, even if this requires readjusting the content to the screen's orientation.

Techniques to meet the criteria

- Include a control that allows access to content in different orientations if automatic reorientation is restricted.

Don't do it!

- Don't block the orientation of the screen.
- Don't display a message asking the user to reorient the device (except when specific targeting is essential).

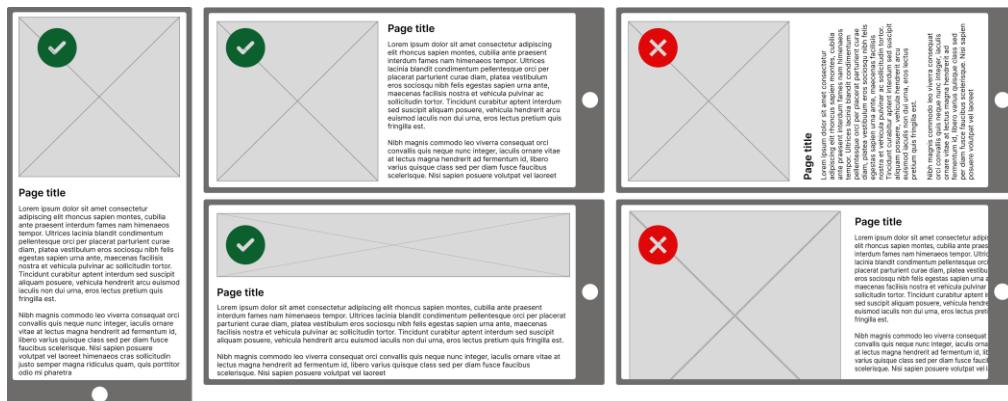


Illustration 15 Correct and incorrect ways to arrange content in landscape format given the vertical content displayed first

Identify Input Purpose

The purpose of each input field that collects information about the user can be programmatically determined when:

- the input field has an *identified* purpose, and
- the content is implemented using technologies with support for identifying the expected meaning for form input data.

This criterion aims to help people recognize and understand the purpose of the fields in a form while also allowing for automatic processing.

For example, suppose the form field that the phone asks us for is identified as such in the code (using the *autocomplete* attribute), and we have already entered our phone number in the form of another website. In that case, the browser can autocomplete the field, reducing the effort of filling in the form and the possibility of making a mistake. In addition, the browser may allow us to customize the presentation of these fields, for example, by using pictograms to identify fields of the same type more clearly on different forms or websites.

The identified and specific purpose of the field must be one of those listed by WCAG 2.2 for the autocomplete attribute, such as name, telephone number, or credit card. In the following pages, we list the values the autocomplete attribute can have and how it must be applied.

This criterion complements and goes a step beyond criterion 4.1.2 "Name, Role, Value", offering real direct value to the user experience.

Techniques to meet the criteria

- Use the HTML 5.2 *autocomplete* attribute in form fields.

Don't do it!

- Don't use incorrect values in the *autocomplete* attribute.

The *autocomplete* attribute

User agents often have features that help people fill out forms based on previously filled forms, and for this purpose, the [autocomplete attribute was formulated within HTML5.2](#)⁵⁴. The autocomplete attribute can be applied to various elements, including `<form>`, `<input>`, `<textarea>`, and `<select>`.

There are two ways to implement the autocomplete attribute:

- As an "anchor", describing the meaning of the value.
- As an "expectation", describing what is expected of people.

The autocomplete attribute behaves as an "anchor" in input elements of type `hidden`. It behaves as an "expectation" in the remaining cases.

When behaved as "expectation", the *autocomplete* attribute has three possible values:

- **Off** when we don't want the system to remember a field for the user's safety or because it will never be reused again.
- **On** when we want the user agent to offer values, but without indicating what type, and the system will decide for itself.
- **A specific token or concatenated tokens:** that tells the user agent to offer values and what kind of values. Check the list of values on the next page.

In the following examples, notice how the autocomplete attribute behaves. In this form, the autocomplete is *on* for all the fields except the email.

```
<!-- registry -->
<form action="" autocomplete="on">

    <label for="fname">First name:</label><input type="text"
        name="fname" id="fname">

    <label for="lname">Last name:</label><input type="text"
        name="lname" id="lname">

    <label for="email">Email:</label><input type="email"
        name="email" id="email" autocomplete="off">

    <input type="submit">
</form>
```

⁵⁴ <http://www.w3.org/TR/html52/sec-forms.html#sec-autofill>

For the amount, the first two fields are *hidden* and serve as anchors, dragging the amount and currency of the transaction from previous steps.

```
<!-- amount -->
<input type="hidden" autocomplete="transaction-currency"
value="EUR">

<input type="hidden" autocomplete="transaction-amount"
value="20.00">
```

For the payment method, the next three fields request information about the credit card, its expiration date, and the security code.

```
<!-- means of payment -->
<label for="number">Credit Card</label><input id="number"
type="text" autocomplete="cc-number">

<label for="expiration">Expiry date</label><input
id="expiration" type="month" autocomplete="cc-exp">

<label for="code">Security code</label>
<input id="code" type="text" autocomplete="cc-csc">
```

In this other example, we ask the user accessing the page to enter the shipping and billing addresses. Notice how the autocomplete attributes are concatenated tokens:

```
<!-- shipping address -->
<label for="eaddress">Address</label><input type="text"
name="eaddress" id="eaddress" autocomplete="shipping
street-address">

<label for="ecity">City </label><input type="text"
name="ecity" id="ecity" autocomplete="shipping address-
level3">

<label for="ecp">Postal code</label><input type="text"
name="ecp" id="ecp" autocomplete="shipping postal-code">
```

Table 8 Fields with identified and specific purposes using Autocomplete

	Autocomplete	Description	Type
Identification	name	Full name	<i>text</i>
	honorific-prefix	Honorific prefix (e.g., <i>Lady</i> , <i>Doctor</i> ...)	<i>text</i>
	given-name	Name	<i>text</i>
	additional-name	Middle name	<i>text</i>
	family-name	Surname	<i>text</i>
	honorific-suffix	Honorific suffix (e.g., <i>Ph.D</i> , <i>IV</i> , <i>Jr</i>)	<i>text</i>
Personal characteristics	sex	Sex	<i>text</i>
	language	Favorite language	<i>text</i>
	photo	Photography	<i>URL</i>
Date of birth	bday	Date of birth	<i>date</i>
	bday-day	Date of birth	<i>number</i>
	bday-month	Month of date of birth	<i>number</i>
	bday-year	Year of date of birth	<i>number</i>
Organization	organization	Company	<i>text</i>
	organization-title	Job position (e.g., <i>Interaction designer</i>)	<i>text</i>
Credentials	username	Username	<i>text</i>
	nickname	Name displayed on the screen	<i>text</i>
	new-password	New password	<i>password</i>
	current-password	Current password	<i>password</i>
	tel	Full phone number	<i>tel</i>
Telephone	tel-country-code	Country telephone code	<i>text</i>
	tel-national	Phone without the country phone code	<i>text</i>
	tel-area-code	Region phone code	<i>text</i>
	tel-local	Phone without country or region code	<i>text</i>
	tel-local-prefix	Local prefix	<i>text</i>
	tel-local-suffix	Local suffix	<i>text</i>
	tel-extension	Phone extension	<i>text</i>

	Autocomplete	Description	Type
Contact	<i>email</i>	Email	<i>email</i>
	<i>impp</i>	Instant messaging	<i>URL</i>
	<i>URL</i>	Web page	<i>URL</i>
Mailing address	<i>street-address</i>	Address, in a multi-line field	<i>textarea</i>
	<i>address-line1</i>	First, second, and third lines in one direction	<i>text</i>
	<i>address-line2</i>		
	<i>address-line3</i>		
	<i>address-level1</i>	First, second, third, and fourth administrative regions, from largest to lowest (e.g., autonomous	<i>text</i>
	<i>address-level2</i>	community > province > city > neighborhood)	
	<i>address-level3</i>		
	<i>address-level4</i>		
	<i>country</i>	Country code	<i>text</i>
Payment	<i>country-name</i>	Country name	<i>text</i>
	<i>postal-code</i>	Zip code	<i>Text</i>
	<i>cc-name</i>	Full name of the owner of the payment method	<i>text</i>
Payment	<i>cc-given-name</i>	Name of the owner of the payment method	<i>text</i>
	<i>cc-additional-name</i>	Middle name of the owner of the payment method	<i>text</i>
	<i>cc-family-name</i>	Last name of the owner of the payment method	<i>text</i>
	<i>cc-number</i>	Payment method number or code	<i>text</i>
	<i>cc-exp</i>	Expiration date of the payment method	<i>number</i>
	<i>cc-exp-month</i>	Expiration month of the payment method	<i>text</i>
	<i>cc-exp-year</i>	Year of expiration of the payment method	<i>text</i>
	<i>cc-csc</i>	Security code (CVC, CVV) of the payment method	<i>text</i>
	<i>cc-type</i>	Type of payment method	<i>text</i>
	<i>transaction-currency</i>	Transaction currency	<i>text</i>
	<i>transaction-amount</i>	Transaction amount	<i>number</i>

Identify the purpose

The purpose of UI components, icons, and regions in content implemented with markup languages can be programmatically determined.

This criterion aims to enhance personalization and adapt to individuals' preferences. For example, identifying regions of the page at the code level enables people to skip or highlight those regions with their user agent.

Many people with limited vocabularies rely on familiar terms or symbols to use the web. However, what is familiar to one user may not be familiar to another. When authors state purpose at the code level, people can leverage customization and user preferences to load a set of symbols or vocabulary that is familiar to them.

It's similar to adding *role* information (as required by criterion 4.1.2). Still, instead of providing information about which UI component is (such as an image), it includes information about what the component represents (such as a link to the homepage).

Techniques to meet the criteria

- Programmatically indicate the purpose of UI icons, regions, and components.
- Use ARIA *landmark* roles to identify regions of the page.
- Use [microformats](#)⁵⁵ to mark up UI components. Microformats are small HTML patterns that semantically represent people's contacts, calendar events, blog posts, places, etc.

Techniques to improve usability

- Enable user agents to find the version of content that best suits their needs.
- Use semantics to identify important features (e.g., `coga-simplification="simplest"`)⁵⁶.
- Use the *aria-invalid* and *aria-required* attributes.

⁵⁵ <http://microformats.org>

⁵⁶ <https://rawgit.com/w3c/coga/master/techniques/index.html>

Guideline 1.4

Distinguishable, separate background from foreground

Make it easier for users to see and hear content, including separating foreground from background.

This guideline tries to make the predefined presentation as easily perceived as possible. It is especially about contrast, not only of colors but also applies to sound content.

Use of color

Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.

Techniques to meet the criteria

If information is communicated by the color of certain words, backgrounds, or other content:

- Verify that the information represented by the colors is available in text.
- Include a text indication for colored form control labels, for example, required fields or fields with errors.
- Check for additional visual cues when color differences in text convey information.
- When links or controls differ only by color, such as linking words within a paragraph, check that their color has a minimum contrast ratio of 3:1 to adjacent text, and provide additional visual cues when hovering over them, such as underlining, in bold or with a larger font size.

If information is communicated by color within an image:

- Differentiate data not only through colors but also with patterns or icons of different shapes.
- Check that the information communicated by color differences is also available in the text.

Techniques to improve usability

- Use CSS to change the presentation of UI components when they receive focus.

Don't do it!

- Don't offer a textual alternative that doesn't include the information communicated by color differences in the image.
- Don't create links that people with color blindness can't distinguish.
- Don't identify required fields or fields with errors using only color differences.

Audio control

If any audio on a web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level.



Since any content that does not meet this criterion may interfere with people's ability to use the entire page, **all content on the website must meet this criterion.**

Techniques to meet the criteria

- If a sound plays automatically, turn it off automatically within three seconds.
- Offer a control near the top of the page that allows people to turn off sounds that play automatically.
- It plays sounds only at people's request.

Don't do it!

- Don't play any sound for more than three seconds without a mechanism for people to turn it off.
- Don't forget to include a way to pause or stop an HTML 5 media item that plays automatically.

Contrast (Minimum)

The visual presentation of text and images of text has a contrast ratio of at least 4.5:1.

Large-scale text and images of large-scale text have a contrast ratio of at least 3:1.

Exception

The following text or images of text have no contrast requirement:

- part of an inactive user interface component
- pure decoration
- not visible to anyone
- part of a picture that contains significant other visual content.

Techniques to meet the criteria

The minimum contrast ratio between the colors of the text (including the text within the text images) and the background should be:

Normal text	
Under 18 points	Equal to or greater than 18 points
4.5:1	3:1

Bold text	
Under 14 points	Equal to or greater than 14 points
4.5:1	3:1

The ratio of dot sizes to CSS pixels is $1\text{pt} = 1.333\text{px}$, so 14 dots and 18 dots are roughly equivalent to 18.5px and 24px.

In both cases, as an alternative to complying with the color contrast:

- Don't specify the background color or text color, nor change the predefined ones.
- Offer a mechanism with sufficient contrast to enable people to switch to a presentation with adequate contrast.

Techniques to improve usability

- Select a technology that enables you to modify the foreground and background colors of text blocks. These controls are standard in user agents, so design pages that work with the most popular browsers so that the developer doesn't override these controls that allow you to change colors.

Don't do it!

- Don't specify foreground colors without specifying background colors, or vice versa.
- Don't use background images that offer insufficient contrast to foreground text (or text images).

Resize Text

All text can be resized without assistive technology up to 200% without loss of content or functionality.

Exception:

- This criterion does not apply to captions or images of text.

Techniques to meet the criteria

- Choose a technology that supports zoom, which is commonly found in user agents such as browsers or PDF readers.
- Ensure that text containers resize when text is resized, and that measures are relative to each other, using one or more of the following techniques:
 - Use em units to specify the size of text containers.
 - Use relative measures to define font size, i.e., percentages, font size names, or em units.
 - To resize text containers, use liquid layouts, or calculate the size and position so that it scales with the size of the text.
- Offer a mechanism on the website that allows users to increase the size of all text on the page by up to 200%.
- Verify that there is no loss of content or functionality when text is resized, but text containers are not resized.

Techniques to improve usability

- Scale form elements that contain text.
- Specify relative measurements for column widths so that lines can average 80 characters or less when resizing the browser window.
- Control the visual presentation of text using CSS.

Don't do it!

- Prevent text, images, or controls from being cut off, overlapped, or hidden when text is displayed at 200%.
- Don't forget also to resize text-based form controls when text is displayed at 200%.
- Don't misuse VW (Viewport Width) units in defining text size because it can cause text not to grow when zoomed in or resized.

Images of Text

Text is used to convey information rather than images of text if the desired visual presentation can be achieved with the technologies used.

Exceptions:

- The text image is visually **customizable** according to users' requirements.
- The particular way of presenting the text is **essential** for the information being conveyed (e.g. a logo).

Techniques to meet the criteria

- Control the visual presentation of text using CSS.
- Use CSS to replace text with text images and provide a mechanism for people to revert the change.
- Separate the information and structure from the presentation form to allow people to have different presentations, for example, by using their own CSS styles.

Techniques to improve usability

- Indicate font sizes in percentages, *em* units, or font size names.
- Control the spacing of letters within words using the CSS *letter-spacing* attribute.
- Position content based on structure markup.

Contrast (Enhanced)

The visual presentation of text and images of text has a contrast ratio of at least 7:1.

Large-scale text and images of large-scale text have a contrast ratio of at least 4.5:1.

Exception

The following text or images of text have no contrast requirement if they are:

- part of an inactive component of the user interface,
- pure decoration,
- not visible to anyone,
- part of a picture that contains significant other visual content, or
- part of a logo or brand name.

Techniques to meet the criteria

The minimum contrast ratio between text (including text in text images) and background should be:

Normal text		Bold text	
Under 18 points	Equal to or greater than 18 points	Under 14 points	Equal to or greater than 14 points
7:1	4.5:1	7:1	4.5:1

In both cases, as an alternative to complying with the color contrast:

- You cannot specify the background color or text color, and you cannot change the predefined ones.
- You can offer a mechanism with enough contrast to allow people to switch to a presentation with enough contrast.



Follow the same techniques to improve usability and avoid the same errors as in conformance criterion 1.4.3.

Low or No Background Audio

For prerecorded audio-only content that

- contains foreground speech,
- is not a sound CAPTCHA or a sound logo, and
- is not a vocalization whose primary intention is to serve as musical expression (like singing or rapping),

at least one of the following is true:

- there's no background sound,
- background sounds can be turned off, or
- background sounds are at least 20 decibels lower than foreground voices (except for occasional sounds that last only 1 or 2 seconds).

Techniques to meet the criteria

- Mix audio files so that sounds not coming from a voice are at least 20 decibels below the sound content of the voice.



A 20 decibel difference roughly equates to background sounds having a perceived volume of 4 times less than foreground voices.

In the chapter [Validation tools](#) you can find a tool and a tutorial to check the contrast of sounds.

Visual presentation

In the visual presentation of blocks of text, a mechanism is provided so that:

- users can choose the background and foreground colors,
- the width is less than or equal to 80 characters or signs (40 if Chinese, Japanese or Korean),
- the text is not justified (aligned to the left and right margins at the same time),
- the spacing between lines is at least one and a half spaces within the paragraphs
- the spacing between paragraphs is at least 1.5 times greater than the line spacing
- text can be resized up to 200% without the need for assistive technology or a horizontal *scroll* to read a line of text in a full-screen window.

Techniques to meet the criteria

To let people change the background and foreground colors, choose one of the following options:

- Specify the background and text colors of secondary content (*banners*, navigation menus, etc.) in the CSS if you have not specified the background and text colors of the main content.
- Specify borders and *layout* in CSS to delimit page areas if you haven't specified background and text colors.
- Choose a technology that allows you to change text blocks' foreground and background color. These controls are common in user agents, so design pages that work with the most popular browsers so that the developer doesn't override these controls that allow you to change colors.
- Don't specify the background or text color, nor change the presets.
- Offer a color selection mechanism on the page that allows people to choose background and foreground colors.

To make the width less than or equal to 80 characters or signs (40 if Chinese, Japanese, or Korean), choose one of the following options:

- Let the user agent handle text repositioning when the window narrows.
- Specify relative measurements for column widths so that lines can average 80 characters or less when resizing the browser window.

To make the text unjustified (aligned to the left and right margins at the same time), choose one of the following options:

- Specify the alignment of text to the left or right in CSS.
- Provide a mechanism to remove the full justification from text.
- Align the text on a single side.

To ensure that the line spacing is at least one and a half spaces within the paragraphs; and the paragraph spacing is at least 1.5 times larger than the line spacing, choose one of the following options:

- Offer a button on the page that allows people to increase line and paragraph spacing.
- Specify line spacing in CSS.

To allow text to be resized up to 200% without the help of assistive products and without needing a horizontal *scroll* to read a line of text in a full-screen window, choose one of the following options:

- Let the user agent handle text repositioning when the window narrows.
- Use a liquid design and relative measurements using one or more of these techniques:
 - Use relative measures to define font size, i.e., percentages, font size names, or *em units*.
 - Use percentages to specify the size of text containers.
 - Calculate the size and position of text containers to scale with the text size.
- Offer options within the content to change the layout without requiring horizontal scrolling to read the content.

Don't do it!

- Don't specify foreground colors without specifying background colors, or vice versa.
- Don't justify the text (simultaneously aligned to the left and right margins).

Images of Text (no exception)

Images of text are used only as decorative elements or when a particular form of text presentation is essential to the conveyed information.

Logos (texts that are part of a logo or brand name) are considered essential content.

Techniques to meet the criteria

- Control the visual presentation of text using CSS.
- Use CSS to replace text with text images and provide a mechanism for people to revert the change.
- Separate the information and structure from the presentation form to allow people to have different presentations, for example, by using their own CSS.

Techniques to improve usability

- Indicate font sizes in percentages, *em* units, or font size names.
- Control the spacing of letters within words using the CSS *letter-spacing* attribute.
- Position content based on structure markup.

Reflow

Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions for:

- Vertical scrolling content at a width equivalent to 320 CSS pixels.
- Horizontal scrolling content at a height equivalent to CSS pixels.

Exception:

- Those parts of the content that require a two-dimensional layout for usage or meaning.

This criterion aims to help people with low vision increase the size of the content up to 400% correctly without requiring them to move their head in two directions.

320 CSS pixels equals a *viewport width* of 1280 pixels at 400% zoom, while 256 CSS pixels equals 1024 pixels at 400%.

Content that requires two-dimensional scrolling and falls within the exceptions are large images such as maps or diagrams, videos, games, presentations, data tables (not individual cells), or toolbars that must be in view while manipulating the content.

Techniques to meet the criteria

- Use *media queries* and *grid* CSS to rebalance columns.
- Use CSS Flexbox to readjust the content.
- Allow readjustment of text strings and long URLs.
- Use the *width*, *max-width*, and Flexbox attributes to adjust labels and form fields.
- Programmatically calculate sizes and positions of elements so that they scale with the size of the text.
- Provide options within the content to switch to a layout that doesn't require *scrolling* to read a line of text.

Techniques to improve usability

- Use *media queries* to remove headers and fixed footers.
- Use the *max-width* and *height* attributes to adjust images to the screen size.
- With CSS, resize simple data tables and make data cells fit in the screen size.
- Offer a mechanism to switch to mobile view at any time.

Don't do it!

- Prevent content from disappearing or becoming unavailable when content is readjusted.

Non-Text Contrast

The visual presentation of the following elements has a contrast ratio of at least 3:1 with adjacent colors:

- **User Interface components:** Visual information needed to identify components and states, except for inactive components or when the appearance is determined by the user agent and not modified by the author.
- **Graphical objects:** Parts of graphics necessary to understand the content, except when a particular presentation is essential to the information being conveyed.

Buttons, form fields, non-decorative icons, graphics (and their different parts), or other active components of the interface need to be clearly perceived and differentiated by all people in all their states. For example, an active form field with a light gray color that does not contrast at least 3:1 with the background color will be difficult to perceive.



Illustration 16 Form with insufficient contrast (first) and with sufficient contrast (second)

Disabled fields, logos, flags, color gradients representing a measurement (such as color maps), naturalistic drawings, or photographs **are not required to pass this contrast ratio**.

The following four pie charts represent the same information. In charts 1 and 2 the South and East sectors have similar tones and are difficult to tell apart. However, in charts 3 and 4, the sectors are separated by lines, which allows the boundary of each sector to be clearly differentiated.



Illustration 17 Graphs with insufficient contrast (1, 2) and sufficient contrast (3, 4). Vision without color blindness

Notice how the same graphs with diverse types of color blindness simulated with the Stark plug-in for Figma would look.

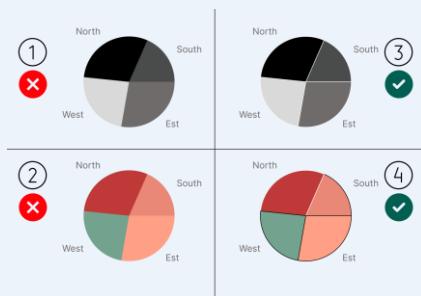


Illustration 18 Blue-yellow color blindness - Tritanopia



Illustration 19 Complete color blindness - Achromatopsia

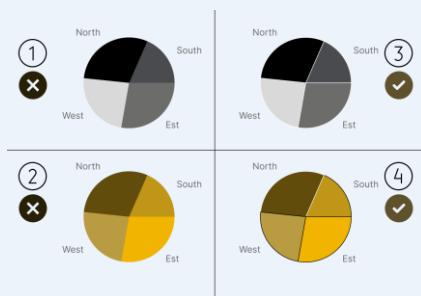


Illustration 20 Red-green color blindness – Protanopia



Illustration 21 Red-green color blindness - Deutanopia

Separation lines would not be necessary in contiguous graph areas with colors sufficiently contrasting. Notice how the same graph is perceived according to the type of color blindness.



Illustration 22 Bar graph with insufficient contrast with each other (left) and with sufficient contrast with each other (right). Vision without color blindness

Notice how this chart would look like with diverse types of color blindness.



Illustration 23 Blue-yellow color blindness - Tritanopia

Illustration 24 Complete color blindness - Achromatopsia

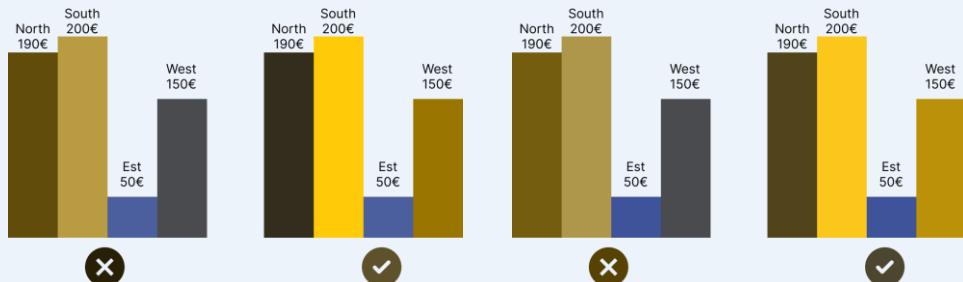


Illustration 25 Red-green color blindness - Protanopia

Illustration 26 Red-green color blindness - Deuteranopia

In addition, according to the criterion "1.4.1 Use of color", the information must not be transmitted solely by color. Therefore, the label associated with each area of the graph must also be included and not only by a legend.

Techniques to meet the criteria

When color is used to identify UI components or their states:

- Use a visible focus indicator designed by the UX/UI designer.
- Provide a control with sufficient contrast to allow users to switch to a presentation with adequate contrast.

When color is required to understand graphic content:

- Make sure that the icons have a contrast ratio of at least 3:1.
- Provide sufficient contrast at the boundaries between contiguous colors.

Don't do it!

- Don't design the outlines and edges of elements in a way that eliminates or makes invisible the visual indicator of focus.

Text spacing

If the markup language used supports the style properties listed below, the text can be formatted as follows without losing content or functionality and without the need to touch other styles:

- **Line spacing:** At least 1.5 times the font size.
- **Spacing following paragraphs:** At least 2 times the font size.
- **Letter spacing:** At least 0.12 times the font size.
- **Word spacing:** At least 0.16 times the font size.

Exception:

- Human languages and scripts that don't use any of these properties can achieve compliance by using only the existing properties for that language-writing combination.

This approach aims to ensure that people who need to modify the presentation of the text, such as people with dyslexia or low vision, can adapt it to their needs so that the content remains readable and operable.

This criterion does not require us to use specific spacing (as in criterion 1.4.8) or to include a tool to customize it. It indicates that if the user configures the spacing in the specified way, no content or functionality should be lost. Additionally, we must not impede or interfere with the customizability of these styles.

Some of these metrics are not applicable to certain languages or writing systems; for example, Japanese does not use spaces between paragraphs.

This criterion only applies if the text is implemented using a markup language (e.g., HTML) but not to PDF documents. It also doesn't apply to text images or captions embedded in the video (rather than included in a separate document).

Techniques to meet the criteria

- Allow the spacing of your text to be overridden or adjusted.

Techniques to improve usability

- Control letter spacing with the *letter-spacing* property in CSS.
- Control the spacing between lines with the *line-height* property in CSS.
- Define the size of text containers using *em* units.

Don't do it!

- Don't allow content to be cropped or overlapped when text spacing is adjusted.



In the [Validation Tools](#) chapter, you'll find a bookmarklet to help you review this criterion.

Content on hover or focus

When pointer hover or keyboard focus is applied and then removed, causing additional content to appear and then disappear, the following conditions apply:

- **Dismissible:** There is a mechanism for discarding additional content without moving the pointer or keyboard focus (for example, the ESC key); unless it communicates a data entry error or does not cover or replace other content.
- **Hoverable:** If the cursor pointer can trigger the additional content, then the pointer can be hovered over it without it disappearing.
- **Persistent:** New content remains visible until the pointer or focus is removed, the user discards it, or its information is no longer valid.

Exception:

- The user agent controls the visual presentation of the additional content and is not modified by the design (for example, the tooltip generated by the title attribute).

Drop-down submenus or custom tooltips are examples of additional content displayed when a component receives the pointer or focus.

This criterion's goal is to help people who have triggered the interaction by mistake, those who have not realized that new content has appeared, or for whom the new content interferes with their ability to perform a task. By complying with this criterion, we ensure that all people can perceive the new content and that they can discard it without problems.

Techniques to meet the criteria

- Make the content displayed when hovering or receiving focus disposable, scrollable, and persistent.
- Use the ARIA role=tooltip.
- Use the CSS hover and focus pseudo-classes.

Don't do it!

- Don't prevent the pointer from landing on the additional content.
- Don't prevent additional content from being discarded without moving the pointer or keyboard focus.
- Don't prevent additional content from remaining visible until it's discarded or invalid.

Principle 2.

Operable

We must be aware of the different devices, peripherals, and assistive products used to access our websites: computers, tablets, mobile phones, consoles, televisions, smart watches, household appliances, cars, street kiosks, augmented and virtual reality, assistive products...

Since we don't control the device our audience chooses, we must design the UI components and navigation elements so that everyone can use them.

Guideline 2.1

Accessible with a keyboard, not everybody uses a mouse

Make all functionality available from a keyboard.

When a functionality can be managed using only the keyboard, it can also be operated using other input systems, such as voice or a mouse, and with a wide variety of assistive products. No other form of input has such flexibility or is universally compatible and operable by people with different disabilities.

Conversely, suppose you only design your website to use with a mouse. In that case, many people will not be able to manage your page, such as a blind user who uses a screen reader.

In addition, WCAG encourages developers to provide additional data entry methods other than the keyboard and mouse, such as optimized speech input.

Keyboard

All content functionalities can be operated via a keyboard interface without reaching a certain speed for each keystroke.

Exceptions:

- When the underlying function requires that input depends on the path of the user's movement and not just the endpoints.

Most actions performed with a pointer can also be done with the keyboard. However, some can only be done with pointers, such as drawing freehand. Additionally, drawing straight lines or geometric shapes, selecting, scaling, or dragging objects can be done only with a keyboard and would not fall within the exception.

This criterion applies not only to physical keyboards but also to keyboards shown on screen, virtual keyboards projected onto surfaces, and any other keyboards that may be invented in the future.

Techniques to meet the criteria

- Ensure that all functions can be operated with the keyboard only.
- Ensure keyboard control by using form controls and HTML links.
- Offer keyboard event handlers using one of the following options:
 - Use keyboard functions and other device-specific functions, such as using *mousedown* in conjunction with *keydown*.
 - Use the *onClick* event in links and buttons, as the *onClick* event of these items is device-independent.
 - Make mouse and keyboard events redundant, such as *Mouseover* with *OnFocus*.

Techniques to improve usability

- Use the role, state, and value attributes of (X)HTML if you want static UI elements to behave like interactive components and add keyboard-accessible actions for static HTML elements.

Don't do it!

- Don't use just pointer-specific event handlers like *onmousedown*.
- Don't programmatically take the focus off an item right when it receives it.
- Don't emulate links with events, as they are not recognizable by software.

No Keyboard Trap

If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface.

And, if it requires more than an unmodified arrow or tab key or other standard exit methods, the user is advised of the method for moving focus away.



Since any content that does not meet this criterion may interfere with people's ability to use the entire page, **all content on the website must meet this criterion.**

Techniques to meet the criteria

- Check that keyboard users don't get caught up in the content.

Don't do it!

- Don't combine multiple content formats so that people become trapped within one format type and cannot return to the rest of the content with just the keyboard.

Keyboard (no exceptions)

All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes.

Techniques to meet the criteria

To meet this criterion, follow the techniques cited in criterion 2.1.1, and keep in mind that no exceptions are allowed.

If it is necessary or mandatory to use input that depends on the path of the user's movements and not only on the start and end points, such as a freehand drawing, then this criterion will not be met.

Character Key Shortcuts

If a keyboard shortcut is implemented in content using only letters (including upper- and lower-case letters), punctuation, numbers, or symbol characters, then at least one of the following is true:

- **turn off:** A mechanism is available to turn the shortcut off;
- **remap:** A mechanism is available to remap the shortcut to include one or more non-printable keyboard keys (e.g., Ctrl, Alt);
- **active only on focus:** The keyboard shortcut for a user interface component is only active when that component has focus.

This criterion prevents people who access content with a speech recognition program from accidentally activating keyboard shortcuts, which is common when they are associated with a single character. Additionally, people who use the keyboard and have mobility impairments may also accidentally press a key and trigger functionalities unexpectedly.

This criterion does not apply to implementing complex keyboard shortcuts (for example, *Alt+F*), the *accesskey* attribute (which is handled differently), or components such as lists or drop-down menus since their shortcuts are only active when the components are in focus.

Techniques to meet the criteria

- Include a mechanism that allows people to disable or reassign character keyboard shortcuts.

Don't do it!

- Don't implement single-character keyboard shortcuts without a way to turn them off or reassign them to a keyboard shortcut.

Guideline 2.2

Give enough time, don't rush

Provide people with enough time to read and use the content.

Imagine you're reading a web page, and in the middle of the process, the page automatically reloads and changes, so you can't finish what you were reading. Annoying, to say the least, isn't it? We all need different amounts of time to complete tasks, and this time is usually longer for older people and people with functional diversity.

This guideline is about removing time limits or providing enough time to access content or complete tasks.

Timing Adjustable

If there is a time limit set by the content, at least one of the following is true:

- **turn off:** The user is allowed to turn off the time limit before encountering it;
- **adjust:** The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting;
- **extend:** The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times).

Exceptions:

- The time limit is a required part of a **real-time event** (for example, an auction), and no alternative to the time limit is possible.
- The time limit is **essential** and extending it would invalidate the activity.
- The time limit is longer than **20 hours**.

This compliance criterion helps people complete a task without unexpected changes in content or context that result from a time limit.

This criterion should be considered with [conformance criterion 3.2.1](#), which limits unexpected changes in content or context resulting from an individual's action.

Techniques to meet the criteria

If there is a session time limit:

- Provide a checkbox on the first page of a form spread over multiple pages, allowing people to request a longer session or no time limit.
- Offer a mechanism that allows people to override the time limit

If the time limit is controlled by a script on the page:

- Offer a mechanism that allows people to override the time limit.
- Provide a way to increase the time limit by 10 times from the predefined limit.
- Alert people when the time limit is about to expire and allow people to increase the predefined limit.

If there is a time limit for reading the content:

- Allow people to pause content and replay it from where it left off.
- Offer a way for people to turn off the time limit.
- When content is automatically moved programmatically, provide a mechanism to pause it.

- Offer a mechanism that allows users to display moving, scrolling, or auto-updating text in a static window or area.

Don't do it!

- Don't refresh or redirect the page using the *http-equiv="refresh"* meta tag with a time limit.
- Don't use server-side techniques to redirect pages after a time limit automatically.

Pause, stop, hide

For information that moves, blinks⁵⁷, or scrolls: if it starts automatically, lasts more than 5 seconds, and is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it.

Exceptions:

- When movement, blinking, or scrolling is essential to an activity.

For information that updates automatically: if it starts automatically and is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it; or control the refresh rate

Exceptions:

- When auto-update is an essential part of an activity.



Since any content that does not meet this criterion may interfere with people's ability to use the entire page, **all content on the website must meet this criterion.**

Suppose the information is updated automatically (by scheduling or because it is received by streaming), and we include the pause option. In that case, it is not mandatory to preserve or present the information between the start of a pause and the restart of the presentation, which could be technically impossible or send an erroneous or misleading message.

For example, a live streaming match in which the user pauses at a specific time. They will reconnect with the game when they restart it, as it is live. This does not prevent the possibility of resuming from where it was paused if the technology used allows it and it is clearly warned.

Additionally, imagine an animation that indicates the preload of a content: if the user could stop it or change its frequency, a misleading message could be sent. In that case, the updated information would be considered essential, and the exception to the criterion would apply.

Techniques to meet the criteria

- Allow people to pause content and replay it from where it left off.
- When content is automatically moved programmatically, a mechanism is provided to pause it.
- Create content that flashes for less than 5 seconds.

⁵⁷ You can find the definition of *blinking* in guideline 2.3.

- Use technologies that allow people to turn off the flicker of content from their user agent.
- Design animated *gifs* to stop after n cycles played in 5 seconds.
- Control the blink programmatically and stop it in 5 seconds or less.
- Offer a mechanism on the page that stops the content from moving, flickering, or refreshing. The control can be at the top of the page or next to the moving content.
- Offer a mechanism to reload the page without flickering content. Deleting the flashing content is only valid if the information from the flashing content is also on the reloaded page without flashing content.

Don't do it!

- Don't include scrolling content without a mechanism to pause and restart it unless it's essential to the activity.
- Don't use the *blink* element.
- Don't create blinking content (with the *text-decoration: blink* property in CSS, programmatically, or within an *object* or *applet*) without also providing a mechanism for people to stop it after 5 seconds or less.

No timing

Time is not essential to the event or activity the content presents, except for non-interactive synchronized media and real-time events.

Techniques to meet the criteria

Allow people to complete an activity without any time limit.

Interruptions

People can delay or suppress interruptions.

Exception:

- Interruptions imply an emergency.

The overall goal is not to interrupt people's flow unless we need to warn them of situations that may harm their health, safety, or money. For example, when the user is going to carry out an operation that involves the deletion of data, as also dealt with in criteria 3.3.4 and 3.3.6.

Techniques to meet the criteria

- Offer a mechanism to postpone any content updates.
- Offer a mechanism to request a content update instead of updating it automatically.
- Schedule non-essential alerts to be optional.

Don't do it!

- Don't refresh or redirect the page using the *http-equiv="refresh"* meta tag with a time limit.

Re-authenticating

When an authenticated session expires, users can continue their activity without losing data after re-identifying themselves.

Techniques to meet the criteria

Offer the option to continue the activity without data loss using one of the following techniques:

- save the data so that the user can take it back after re-authenticating; or
- scramble the data entered by the user as hidden or encrypted data on the page to re-authenticate.

Don't do it!

Don't limit session time without providing a mechanism to save the user's entries and restore that information when they re-authenticate.

Timeouts

Users are warned of the duration of any user inactivity that could cause data loss unless the data is preserved for more than 20 hours when the user does not take any actions.

Overly complex forms, such as buying a plane ticket or booking a hotel room, can be overwhelming for people with cognitive limitations. These people need to pause to make decisions or translate texts. If the data is lost at that moment of pause, the user should start the process again, with the consequent added mental load.

It is important to provide correct notice of time limits and, if possible, add mechanisms to adjust, extend, or cancel them, as explained in conformance criterion 2.2.1.

Privacy regulations may require individuals' express consent before their identification and data is stored, for example, with the European GDPR regulation. The W3C recommends consulting with a legal specialist to apply this criterion in each country.

Techniques to meet the criteria

- The session expires after at least 20 hours of inactivity.
- Save the user's data for more than 20 hours.
- Advise of the duration of inactivity at the beginning of the process.

Guideline 2.3

Flashes and motion animations under control

Don't design content in a way that could cause seizures or physical reactions.

People with photosensitive epilepsy react to sudden changes in luminosity. The highest speed allowed to display flashing content is three flashes per second. However, some people respond at lower speeds, so removing this type of content is recommended.

To delimit the concept of *flash*, it is necessary to recover the idea of *blinking*, which appeared in criterion 2.2.2:

- **Blinking:** switch back and forth between two visual states in a way that is meant to draw attention. Blinking is distracting for some people, and, as we saw in criterion 2.2.2, it is only allowed if it lasts less than 5 seconds or can be stopped.
- **Flash:** Changes in relative brightness (sudden changes from light to dark) can cause seizures, dizziness, and headaches in some people if they are over a specific size or frequency range. Flashes, unless they meet particular requirements, are not allowed for a second because they could cause damage before the user can avoid them, or there may even be people who do not know that they are sensitive to flashes.

Three Flashes or Below Threshold

Web pages do not contain anything that flashes more than three times in any one-second period or the flash is below the general flash and red flash thresholds.



Since any content that does not meet this criterion may interfere with people's ability to use the entire page, **all content on the website must meet this criterion.**

A **general flash** is defined as a pair of increments followed by decreases (or vice versa) of 10% or more in maximum relative luminosity, where the relative brightness in the darkest image is less than 0.80.

A **red flash** implies a couple of increments followed by decreases (or vice versa) involving a saturated red.

Techniques to meet the criteria

- Check that no component of the content flashes more than 3 times per second.
- Make the flashing content take up a small space, meaning the flashing area should be less than 25% of the 10 degrees of the visual field (representing the central area of vision in the eye). For example, the safe area for a resolution of 1024x768 at a viewing distance of between 22 and 26 inches is any shape with an area less than 21824 pixels, or what is the same, an approximate square of 148 pixels on a side.
- Use a tool to ensure that the content does not violate the general or red flash threshold.



The [Validation tools chapter](#) includes a reference to the PEAT tool, which can help you determine whether your content respect these thresholds.

Three flashes

Web pages contain nothing that flashes more than three times a second.

Techniques to meet the criteria

Check that no component of the content flashes more than three times per second.

Animation from interactions

Motion animation triggered by interaction can be disabled.

Exception

- the animation is essential to the functionality or the information being conveyed.

Animated **movements** include steps between different positions or sizes to create the illusion of movement or smooth transition. However, animated movements do not involve changes in color, sharpness, or transparency (for this, criteria 2.3.1 and 2.3.2 relate to flashes).

An example of animations that can cause problems is "*parallax effects*," where layers slide at different speeds as the user moves up or down the page. Another example is fantasy transitions between pages or a quick zoom-in on elements that take focus.

This criterion aims to help people with dizziness, vertigo, or other vestibular disorders. A "safe" decision would be to remove animations, but this is too radical and isn't always the best option. When used judiciously, animation can be a good way to communicate the relationship between different areas, attract people's attention, or add some fun to the interaction.

Techniques to meet the criteria

- Use the `{prefers-reduce-motion}` media query to avoid animations based on user-agent accessibility settings.
- Allow people to define a preference that prevents animation.

Guideline 2.4

Easily navigable information architecture

Provide mechanisms to help people navigate, find content, and determine where they are.

Finding content and knowing where it is can be complex tasks for people with disabilities, especially those accessing using screen readers or people with cognitive disabilities.

Bypass blocks

There is a mechanism that allows people to bypass blocks of content that are repeated on multiple web pages.

Techniques to meet the criteria

Create links to skip repeated content blocks with one of these techniques:

- Add a link at the top of the page to jump straight to the main content.
- Add a link at the start of each repeated block to go to the end of that block.
- Add links at the top of the page to go to each content area.

Group repeated content blocks in such a way that they can be skipped using one of these techniques:

- Identify regions of the page using ARIA *landmarks roles*.
- Provide header elements at the beginning of each content section.
- Use *iframe* and *frame* elements identified with the *title* attribute.
- Use expandable and collapsible menus.

Techniques to improve usability

- Position content based on structure markup.
- Group related links together using the *<nav>* element.

Page Titled

Web pages have titles that describe their subject matter or purpose.

Techniques to meet the criteria

- Provide a descriptive title for each web page using the *title* element.

Techniques to improve usability

- Identify the relationship of the web page to a set of pages. For example, in the page's title, include the site's name.

Don't do it!

- Avoid page titles that do not identify the page or its contents, for example, with texts such as "Untitled document", "page.html" or empty text.

Focus order

If a web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.

"Navigate sequentially" means to move forward with focus from one element to the next in the defined order, using a keyboard interface, such as the Tab key.

Techniques to meet the criteria

- Place interactive elements in an order that follows sequences and relationships within the content.
- Match the DOM's order to the visual order so that the focus travels through the elements in an order that follows the sequences and relationships within the content.
- If you want a page to change dynamically, choose one of these options:
 - insert dynamic content into the DOM immediately after the trigger element
 - create custom dialogs that can be activated independently of the input device and placed in the DOM after the element that triggered them, or
 - reorder sections of the page with the DOM.

Don't do it!

- Don't use the `tabindex` attribute to create a tab order that doesn't preserve the meaning and operability of the page.
- Avoid using dialogs or drop-down menus not adjacent to the button that activates them when navigating sequentially.

Link Purpose (In Context)

The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context.

Exception:

- When the purpose of the link is ambiguous to people in general.

Techniques to meet the criteria

- Provide link text that describes its purpose (if you include an image, your *alt* text is part of the link text).
- If using an image map, offer textual alternatives for area elements.
- Allow people to choose between short or long links with a control at the top of the page or programmatically.
- Identify the purpose of the link using the link text combined with the text of the encompassing sentence.
- Provide additional information about the purpose of the link:
 - with the *title* attribute of the link.
 - in the link text itself, some text is hidden with CSS. It is crucial that you hide it with a technique that hides it visually but not for the screen reader.⁵⁸
- Identify the purpose of a link using the link text combined with its software-determined context, using:
 - the *aria-labelledby* attribute (example 1 in the following table); or
 - the *aria-label* attribute; or
 - the paragraph containing the link (example 2); or
 - the list item that contains the link (example 3); or
 - in a nested list, the parent element that contains the list in which the link is located (example 4); or
 - the table cell that contains the link plus its associated headers.

In the Table 9 there are four examples of a link text combined with its context in a way that can be *programmatically determined*:

⁵⁸ Learn different techniques to hide content at "[Hiding Elements in CSS: the Accessible Way](#)" by Abdulquodus Abubakre .

https://dev.to/ibn_abubakre/hiding-elements-in-css-the-accessible-way-5h1b

Table 9 Link text combined with software-determinable context

Example	Correct	Incorrect
1	Annual Report Download PDF <pre><h3 id="t1"> Annual Report</h3> <p> Download in PDF</p></pre>	Annual Report Download PDF <pre><h3>Annual Report</h3> <p>Download PDF </p></pre>
2	Annual Report in PDF <pre><p>Annual Report in PDF</p></pre>	Annual Report in PDF <pre><p>Annual Report</p> <p>in PDF</p></pre>
3	1. Annual Report (PDF) 2. ... <pre> Annual Report (PDF) ... </pre>	1. Annual Report 2. (PDF) <pre> Annual Report (PDF) </pre>
4	- Annual Report - (HTML) - (PDF) - ... <pre> Annual Report (HTML) (PDF) ... </pre>	Annual Report - (HTML) - (PDF) <pre><p>Annual Report </p> (HTML) (PDF) </pre>

Techniques to improve usability

- Combine an image and text into a single link instead of having two links, one for the image and one for the text.
- Identify the purpose of the link using your own text and previous heading.

Don't do it!

- Don't offer links when the context needed to understand them isn't related to the code-level link (with *aria-label* or *aria-labelledby*; or because they're in the same sentence, paragraph, list item, or table cell).
- Don't use an image without an accessible name when the image is the only content in a link.

Multiple Ways

More than one way is available to locate a web page within a set of web pages

Exception:

- The page is an intermediate step or the result of a process.

This criterion allows people to find content that best suits their needs.

For example, some people may prefer to use a search engine rather than a navigation menu. However, the search engine must find content within the set of pages on which it is located. For example, an e-commerce portal may have a search engine on product pages, which can search only within this set of pages, and a search engine within the help center, which can search only within the help center.

Techniques to meet the criteria

Offer two or more of the following mechanisms:

- links to navigate to related web pages,
- a table of contents,
- a site map,
- a search engine,
- a list of links to all the pages of the site on the homepage (it is effective on small websites),
- a list of links to all pages of the site on all pages of the site (effective on small websites).

Techniques to improve usability

- Identify the relationship of the web page with the set of pages by means of *link* elements plus the *rel* attribute, which can be used by the navigation tools available in some browsers.

Headings and Labels

Headings and labels describe the topic or purpose.

This compliance criterion does not require headings or labels, nor does it require that the content acting as a heading or tag be correctly marked or identified, as these aspects are covered in other criteria, such as conformance criteria 1.3.1, 2.4.11, 3.3.2, or 4.1.2.

This criterion only indicates that, in the case of providing headings and labels, they must be descriptive. Remember that "descriptive" doesn't mean they need to be long.

Techniques to meet the criteria

- Offer headings and tags that are descriptive.

Focus Visible

Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.

Techniques to meet the criteria

- Use interface components that the user agent highlights when they receive focus.
- If you want to change the layout of the elements when they receive focus, do it with CSS or scripting.
- Use the browser's native focus indicator, which will allow you to use the high-visibility keyboard focus settings that platforms or certain assistive products allow.
- Design a highly visible system focus indicator.
- Create a two-color focus indicator to ensure sufficient contrast across all components.

Don't do it!

- Don't programmatically take the focus off an item right when it receives it.
- Don't hide the visual indicator of the spotlight due to the style of the edge or outer line of the element. For example, don't hide it with the `outline: none` or `outline:0` style.

Location

Information about the user's location within a set of web pages is available.

Techniques to meet the criteria

- Offer a breadcrumb path.
- Provide a site map.
- Indicate the current location in the navigation menu.
- Identify the web page's relationship with a set of pages with *link* elements plus the *rel* attribute, which can be used by the navigation tools available in some browsers.

Link Purpose (Link Only)

A mechanism is available to allow the purpose of each link to be identified from link text alone.

Exception:

- When the purpose of the link should be ambiguous to people in general.

Techniques to meet the criteria

- Use the *aria-label* attribute.
- Provide link text that describes its purpose (if you include an image, your *alt* text is part of the link text).
- If you're using an image map, offer textual alternatives for area elements.
- Allow people to choose between short or long links with a control at the top of the page or programmatically.
- Provide supplementary information about the purpose of the link in the link text itself, hiding part of the text with CSS. It is crucial that you hide it visually but not for the screen reader, for example with *text-indent:-1000px* but not with *display:none*.⁵⁹

Techniques to improve usability

- Combine an image and text into a single link instead of having two links, one for the image and one for the text. (Example 1 and 2 of the table below)
- Identify the purpose of the link using its *title*. (Example 3)

Table 10 Identifying the purpose of the links

Example	Result	Correct
1	<u>Annual report</u> <small>pdf</small> 	<pre> Annual Report </pre>
2	 Home	<pre> Home </pre>
3	<u>OK</u>	<pre>OK</pre>

⁵⁹ Learn different techniques to hide content at "[Hiding Elements in CSS: the Accessible Way](#)" by Abdulquodus Abubakre.

https://dev.to/ibn_abubakre/hiding-elements-in-css-the-accessible-way-5h1b

Don't do it!

- Don't use links like "Click here" or "Read more" without a mechanism to change the link text to more specific text.
- Don't use an image without an accessible name when the image is the only content in a link.

Section Headings

Section headings are used to organize the content.

This criterion refers to structuring sections of text, not UI components.

Techniques to meet the criteria

- Organize the page using headings.
- Provide headings (*h1-h6*) at the beginning of each section of the content.

Header tips

1. Headings should **break** up large chunks of text.
2. Headings should be placed just **before their associated content**.
3. Headings should **summarize** the content they are associated with.
4. Headings should **be short** to allow for quick page scanning.
5. Headings should be **consistent** throughout the website.
6. Headings should **structure** the contents of the page and not just be used to display visual effects.
7. All visual headings must have a **structured header system** (*h1-h6*).
8. Pages should have a **single h1 heading**, which is the main title of the page. At least on the inside pages, this heading should be located at the beginning of the main content of the page.
9. Header levels should not be **skipped**. The subheadings of *h1* should be *h2*. The subheadings of *h2* should be *h3*, and so on. For example, you should not jump from an *h1* to an *h3*.
10. The **legend and label elements should be used in the forms**, not the *h1-h6* headings.

Focus Not Obscured (Minimum)

When a UI component receives keyboard focus, the component is not entirely hidden due to author-created content:

- If the user can reposition the content (move the toolbars, move the non-modal dialog boxes...), only their initial position is considered.
- Content opened by people can hide the component that receives the focus. In these cases, if the user can reveal the focused component without advancing the focus with the keyboard, the focused component is not considered hidden. For example, by pressing the ESC key to dismiss open content; or by using the arrow keys to scroll the content.

The most common examples of content that overlaps an element with the focus are fixed headers and footers or non-modal layers, such as the cookie acceptance layer. However, it must be considered that if the superposition is complete but semi-opaque, it would fulfill the criterion because it would not completely hide the focus.

Drop-down menus, date selection calendars, drop-down fields, or tooltips are components that cover the content, but they are not persistent. Therefore, they do not violate this criterion. However, **if the implementation of the component allows them to persist**, they may fail to meet the criteria. For example, if you open a drop-down menu and, when you leave it with the keyboard focus, it does not close, and the keyboard focus ends up in content hidden by that unclosed drop-down menu, the criterion would be violated.

Techniques to meet the criteria

- Use the scroll-padding CSS property to allow people to access UI components (e.g., links, buttons, and form fields) that are initially completely hidden by a fixed-position component (e.g., a fixed cookie consent banner at the bottom of the page).

Don't do it!

- Don't use fixed headers or footers or stickers (known as "sticky components") in such a way that they can completely hide focusable elements (such as buttons, links, form fields, etc.).

Focus Not Obscured (Enhanced)

When a UI component receives keyboard focus, no part of the component is hidden by author-created content.

This criterion is similar to the previous criterion but stricter.

Conformance criterion 2.4.11 allows content created by the author (either the UX/UI designer, the developer, or the content creator) to hide some part of the component that receives the focus, while **2.4.12 prohibits any part of the component that receives the focus from being hidden** by content created by the author.

Techniques to meet the criteria

Follow the same techniques and advice from criterion 2.4.11 to meet this criterion.

Focus Appearance

When the keyboard focus indicator is visible, an area of the focus indicator meets all the following:

- it is at least as large as the area of a 2-pixel CSS pixel thick perimeter of the component or subcomponent without the focus; and
- it has a contrast ratio of at least 3:1 between the same pixels in the state with the focus and without the focus.

Exceptions:

- The focus indicator is determined by the user agent and cannot be adjusted by the author.
- The author has not changed either the indicator of the spotlight or the color of the indicator's background.

This criterion complements:

- criterion 2.4.7 (Focus Visible) requires the focus to be visible; criterion 2.4.13 now also requires a minimum level of visibility based on size and contrast; and
- criterion 1.4.11 (Non-text contrast) requires an adequate contrast of the focus indicator with the background; criterion 2.4.13 requires sufficient contrast between the two states of the component with and without the focus.

The criterion refers specifically to the edge, not to effects such as shadows or highlights.

In the explanation of the criterion, you will find multiple practical cases. For example, if we want the focus indicator to show a border of two black pixels (with enough contrast with the background) around a button, we can choose four techniques, and not all of them would meet the criteria:

- In the first three techniques, buttons with focus ("outset", "outline", "border") would meet the criterion since they are equal to or larger than the interface element;
- However, the fourth button with focus ("Inset") would not comply, and would need the indicator to be at least 3px.

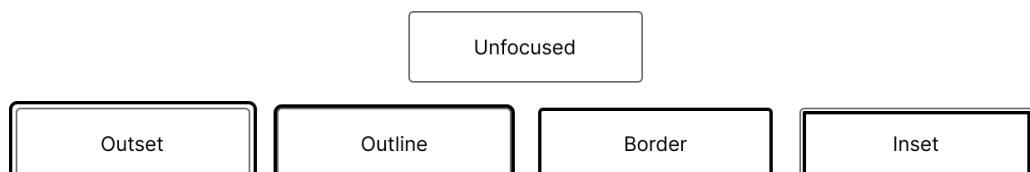


Illustration 27 Examples of buttons without and with focus

Techniques to meet the criteria

- Design a highly visible system focus indicator.
- Create a two-color focus indicator to ensure sufficient contrast across all components.
- Create a solid focus indicator within the component.

Don't do it!

- Don't use a CSS border on inline text that can be readjusted.

Guideline 2.5

Input modalities, multiple ways to enter information

Make it easier for users to operate functionality through various inputs beyond the keyboard.

All functionalities should be accessible via pointer input devices like a mouse, touchscreen, stylus, laser pointer, or other interactions that may arise in the future.

Users with motor impairments may struggle with timed or complex gestures, such as drag-and-drop or swiping, or small targets to accurately position the pointer hover them. Therefore, providing simpler, untimed alternatives and big targets are essential to ensure accessibility.

Pointer gestures

All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture.

Exception:

- The use of a multipoint or path-based gesture is essential.

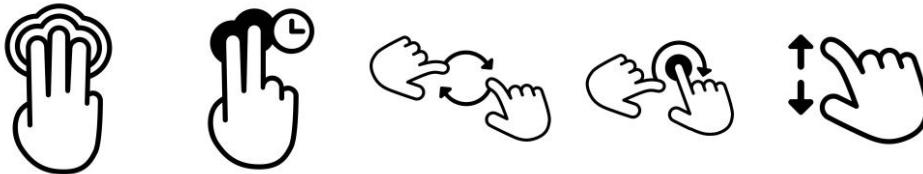


Illustration 28 Different pointer gestures with 2 or 3 fingers

These images show some examples of these gestural paths or interactions with several points at the same time: two touches with three fingers at the same time, a long press with two fingers, turning with two fingers, tapping with one finger and rotating with another... or the classic one of enlarging the screen with an index finger and thumb in opposite directions.

People with limited mobility, who cannot perform these movements, or people with learning difficulties need a simpler alternative mechanism to interact with the system, such as a pointer (even though the system must also be operable via keyboard).

Tap (click), double tap (double click), and long presses are examples of single-point activation.

Note that the criterion does not apply to actions that are required to operate the user agent or assistive product, such as swiping to scroll, spreading two fingers to zoom, or rotating fingers to activate the VoiceOver rotor. However, it would apply to swiping actions implemented on the page, such as the gesture of swiping a carousel.

Techniques to meet the criteria

- Provide controls to achieve the same result as multipoint or path-based gestures. For example, alternative previous and next buttons to the swipe gesture.
- Enable the sliders to be operated with a single touch.

Don't do it!

- Don't provide gesture-based functionality without an alternative that can be operated with a single point.

Pointer cancellation

For functionality that can be operated using a single pointer, at least one of the following is true:

- **no down-event:** The down-event of the pointer is not used to execute any part of the function;
- **abort or undo:** Completion of the function is on the up-event, and a mechanism is available to abort the function before completion or to undo the function after completion;
- **up reversal:** The up-event reverses any outcome of the preceding down-event;
- **essential:** Completing the function on the down-event is essential.

When we press a key, mouse button, or control on the touchscreen, there are at least two movements, one of pressing "down" (`onmousedown`, `onkeydown`, `onkeypress`, `ontouchstart`) and one of releasing "up" (`onmouseup`, `onkeyup`, `ontouchend`).

People with difficulty handling the pointer, low vision, or cognitive limitations need the assurance that, when handling content, a touch "down" by mistake can be corrected. In the default one-click link and button interaction, event cancellation is built in by default. If the user discovers that they have selected the wrong item, they can cancel the action by moving the pointer or finger away from the target before releasing it.

An example of an **exception** would be a keyboard displayed on the screen. We are used to the fact that when the key is pressed, the chosen character is shown on the screen; that is, the functionality is triggered in the "down" event, and, therefore, it would be an essential feature of the functionality. The same would happen in a piano application; it would be essential that the sound sounded in the event below.

It should also be noted that the criterion does not apply to the actions required to operate the user agent or the supporting product.

Techniques to meet the criteria

- Ensure that drag-and-drop actions can be canceled.
- Use native controls to ensure the functionality is triggered in the "up" event.
- Screen tap events are triggered only when the touch is removed from the controller.

Don't do it!

Don't trigger a control on a "down" event if the "up" event doesn't reverse the result.

Label in Name

For user interface components with labels that include text or images of text, the name contains the text that is presented visually.

A best practice is to have the text of the label at the start of the name.

Many HTML elements have accessible names and are calculated in a variety of ways: from their content, an attribute, or an associated element. In the following examples, the accessible name will be, in all cases, "Back to Start":

- Back to Start
-
-

-
 Back to
- <input type="button" value="Back to Start" />
- <input type="checkbox" id="xx">Check this box for <label
 for="xx">Back to start</label> at checkout.

With the *aria-label* and *aria-labelledby* properties of the ARIA standard, we can modify these accessible names. You can find more information in the [ARIA chapter, the ally of accessible HTML](#).

In the following example, the accessible name of the "Read more" link would become "A new storm enters the city":

```
<h2 id="headline_1">A new storm approaches the city</h2>
<p>[News lead]</p>
<p><a href="..." aria-labelledby="headline_1">Read more</a></p>
```

Accessible names allow users of assistive products to differentiate components, navigate through them, and finally manage them. It may seem that the above example is a good choice, because the screen reader will read me a clearer link text, thus meeting criterion 2.4.4.

However, **people who access by voice will not be able to interact** with that link. These people will say "Read more" to activate the link (if there are several links with the same name, they will be highlighted with a number, so that the user indicates which one they want to activate). But the link no longer responds to the name "Read more", but to the name "A new storm approaches the city", therefore, it will not be activated.

The correct thing to do is:

```
<h2 id="headline_1">A new storm approaches the city</h2>
<p>[News lead]</p>
<p><a href="..." id="en1" aria-labelledby="en1 headline_1">Read
more...</a></p>
```

With this code:

- the screen reader will read a descriptive link: "Read more... A new storm approaches the city", also understandable out of context;
- people who use a screen reader, but who see the screen, will have a better experience if what the reader advertises is the same as the label they see, or at least contains it; and
- the link will already be accessible by voice through its visible "Read more" label.



You can read more about [how browsers calculate element's accessible name](#)⁶⁰ and [how to map the names of each element with aria-label, aria-labelledby, and aria-describedby](#) ⁶¹.

Techniques to meet the criteria

- Include the visible label text as part of the accessible name.
- Make the accessible name match the visible tag.

Techniques to improve usability

- Position labels in a way that helps predict relationships.
- If a tile doesn't have accompanying text, consider using floating text as accessible name.

Don't do it!

Avoid accessible names that:

- do not contain the visible text,
- contain the visible text, but the words are in a different order,
- contain the visible text, but one or more words intermingle with the words on the visible label.

⁶⁰ <https://www.w3.org/TR/accname-1.1/>

⁶¹ <https://www.w3.org/TR/core-aam-1.1/>

Motion Actuation

Functionality that can be operated by device motion or user motion can also be operated by user interface components, and responding to the motion can be disabled to prevent accidental actuation.

Exceptions:

- Motion is used to operate functionality using an accessibility-supported interface.
- Movement is essential to functionality and failure to do so would invalidate the activity.

This criterion ensures that functions triggered by device movement (e.g., tilting or shaking the phone) or user movement (e.g., moving the hand in front of the webcam) can also be activated by user interface components unless the movement is essential to that function.

This criterion only applies to motion sensors inherent in devices (such as accelerometers and gyroscopes), not to GPS geolocation or to movements indirectly associated with the use of keyboards, pointers, and other technologies.

In addition, people with tremors or motor disabilities must be able to deactivate this type of movement action to avoid accidentally performing it.

Techniques to meet the criteria

- For all motion-activated inputs, it provides conventional controls that perform the same function, and preference settings that allow people to turn off motion actuation.
- Ensure compatibility with system features that allow people to turn off motion actuation.

Don't do it!

- Don't implement a motion performance without the ability to turn it off.
- Don't disable or otherwise hinder people's ability to turn off system-level motion actuation.

Target Size (Enhanced)

The size of the target for pointer inputs is at least 44 by 44 CSS pixels.

Exceptions:

- **Equivalent:** The target is available through an equivalent link or control on the same page that is at least 44 by 44 CSS pixels.
- **Inline:** The target is in a sentence or block of text
- **User Agent Control:** The size of the target is determined by the user agent and is not modified by the author
- **Essential:** A particular presentation of the target is essential to the information being conveyed.

The aim of this criterion is to help people who have tremors, low vision, poor dexterity pointing at small targets with their finger or pointer, or who are simply traveling on the bus and are exposed to movements of the transport itself.

Links included within a phrase or block of text are an exception, but they are not an exception if the words or block of text is a link as a whole.

Although 44 by 44 pixels is the minimum measurement, **it is recommended to be higher** when:

- the control should be used frequently;
- the result of their interaction cannot be easily undone;
- is near the edge of the screen;
- is difficult to reach;
- is part of a sequential task.

Techniques to meet the criteria

- Ensure that the interactive areas are at least 44 by 44 CSS pixels.
- Ensure that online links have a large enough area of interaction.

Don't do it!

- Don't design interactive areas smaller than 44 by 44 CSS pixels.

Concurrent Input Mechanisms

Web content does not restrict the use of input modalities available on a platform.

Exceptions:

- When the restriction is essential.
- When the restriction is necessary to ensure the security of the content.
- When the restriction is necessary to respect the user's settings.

We should not assume that people always use a single particular input mechanism, so we should allow the use of any input mechanism at their disposal: keyboard, mouse, voice, touchscreen, etc.

We must also allow people to switch from one modality to another at their will.

Techniques to meet the criteria

- In JavaScript, use only high-level event handlers, independent of the input mode (e.g., *focus*, *blur*, *click*).
- In JavaScript, use redundant event handlers for keyboard, pointer, and touchscreen (for example, enable the same functionality in *keydown*, *mousedown*, *touchstart*, or *pointerdown*).

Don't do it!

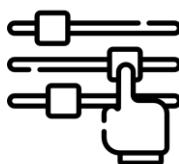
- Don't limit interactions on touchscreen devices, assuming that no other input mechanisms, such as a mouse or keyboard, will be added or used.

Dragging Movements

Any functionality that operates by dragging must be able to be performed with a simple pointer without dragging.

Exceptions:

- The dragging motion is essential.
- The user agent determines functionality and is not modified by the developer.



Some people cannot perform dragging movements accurately. Others use an input device, such as a head pointer, voice control, or eye tracking, making dragging complicated, error-prone, or impossible.

"Swipe" is not the same as "dragging". We speak of "dragging" when, in the down event, an element (or a representation of its position) follows the pointer to the upload event. Whether you need simple gestures to "swipe," for example, a carousel on a touchscreen, is already covered by the "2.5.1 Pointer Gestures" criterion.

It should also be clarified that a component with *drag & drop* already has to be accessible by keyboard thanks to the criterion "2.1.1 Keyboard", but this does not imply that this solution works by simple taps in touch access; that is covered by this new criterion.

Techniques to meet the criteria

- Ensure that an alternative is available for drag moves that operate on content that does not require dragging.

Don't do it!

- Don't forget to provide a unique pointer method for people to operate a function that requires a dragging motion.

Target Size (Minimum)

The size of the target for pointer inputs is at least 24 by 24 CSS pixels.

Exceptions:

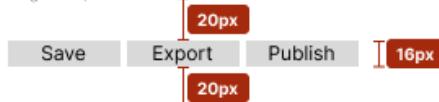
- **Spacing:** Interaction zones that are smaller than 24 by 24 CSS pixels in size are positioned so that if a circle of 24 CSS pixels in diameter is drawn centered on the bounding box of each, the circles do not intersect with another object or with the circle of another interaction zone smaller than 24 by 24 CSS pixels.
- **Equivalent:** The feature can be achieved through a different control on the same page that meets this criterion.
- **Inline:** The interaction area is within a sentence, or its size is restricted by the line height of text that is not part of the interaction area. The height of the line should be interpreted as perpendicular to the flow of text. For example, in a language written vertically, the height of the line would be horizontal.
- **User agent control:** The size of the interaction area is determined by the user agent and is not modified by the author. For example, it could be calendar days in an `input type="date"`.
- **Essential:** A particular presentation of the area of interaction is essential or legally required for the information being transmitted. For example, on maps, pins indicating places may be presented close together, but it is essential to display them in the correct location on the map.

This criterion is the least severe variant of the AAA-level "2.5.5 Target Size (Enhanced)" criterion, where the minimum size is defined as 44 by 44 CSS pixels, a dimension that should be pursued whenever feasible.

The purpose of both criteria is to ensure that interactive elements are easy to activate without the risk of unintentionally tapping a nearby one. When the interactive region of an object is small, people with tremors in their hands or difficulties in fine motor dexterity face difficulties in activating them accurately. By providing an appropriate size, or sufficient space between interaction areas, you decrease the likelihood of accidentally triggering the wrong control.

Look at the example of the Illustration 29. It displays three 16-pixel-high buttons with a bottom and top margin of 20 pixels. The interaction area (a circle of 24 pixels) can act on the buttons without interfering with other components.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

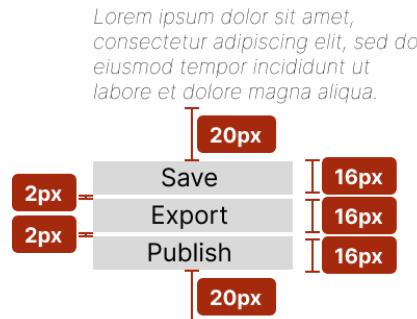
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Illustration 29 Buttons with a sufficient interaction area

However, as seen in the Illustration 30, when you view the same page on a smaller screen, the buttons are stacked on top of each other:



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Illustration 30 Buttons with an undersized interaction area

In this case, the interaction area of the elements overlaps, and there is not enough space to click on them without making a mistake. Therefore, it would not be enough to meet the criterion.

It is important to note that links marked as listed in a navigation structure, such as the one shown in the Illustration 31, do not count as online links and are, therefore, no exception. UX/UI designers can anticipate the relative position of these links and provide enough space for the interaction zone.

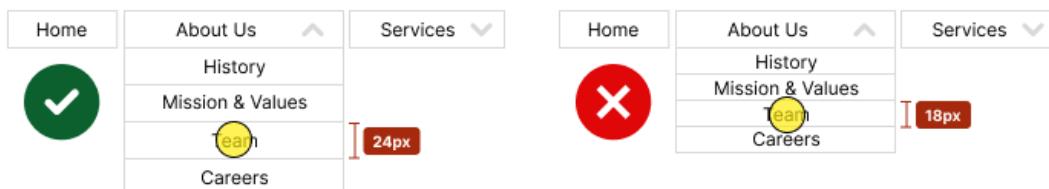


Illustration 31 Drop-down menu with links

Techniques to meet the criteria

- Use the `min-height` and `min-width` attributes to ensure sufficient space between hotspots.

Principle 3.

Understandable

If the people who access our website don't understand what we're telling them or we make them feel lost, we have a problem.

We must design our website simply, and this includes both the information and the user interface.

Guideline 3.1

Readable, clear for everybody

Make text content easy to read and understandable.

Content is king on the internet. This guideline is intended especially for page editors so that they prepare texts in such a way that anyone can understand them.

As a complement to this guideline, we have dedicated a chapter to "[Caring for the words](#)".

Language of Page

The default human language of each web page can be programmatically determined.

Techniques to meet the criteria

Identify the default language(s) using the language attributes.

For example, in HTML 5 you would specify English like this:

```
<html lang="en">
```

In XHTML 1 we must use the *lang* attribute and the *xml:lang* attribute together, with the same value each time we set the language.

```
<html lang="en" xml:lang="en"  
xmlns="http://www.w3.org/1999/xhtml">
```

Techniques to improve usability

Specify the default language for the entire page using the *server-side* HTTP header.

For example, in PHP it would look like this:

```
<?php header('Content-language: en'); ?>
```

Language of the parts of the page

The human language of each passage or phrase in the content can be programmatically determined.

Exceptions: proper nouns, technical terms, words in an indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

For images that include text in another language, the *alt* attribute must contain those words, and the language attribute must be used in the image for the screen reader to read it correctly.

Techniques to meet the criteria

- Identify language changes using the language attribute (*lang*, *xml:lang...*) in HTML elements.



You can read more about [internationalization on the W3C website](#)⁶².

⁶² <https://www.w3.org/International/questions/qa-html-language-declarations>

Unusual words

A mechanism is provided to identify specific definitions of words or phrases used in unusual or restricted ways, including idioms and jargon.

Techniques to meet the criteria

If the unusual phrase or word has a unique meaning within the website:

- Define it the first time it appears using one of these techniques:
 - link to definitions; you can use definition lists (*DL*);
 - add the inline definition; in HTML you can use the *dfn* element.
- Define it each time it appears using one of these techniques:
 - link to definitions; you can use definition lists (*DL*);
 - provide a glossary;
 - provide a search functionality in an online dictionary.

If the unusual phrase or word is used with different meanings within the website:

- Link to definitions; you can use definition lists (*DL*).
- Add the inline definition; in HTML you can use the *dfn* element.

Abbreviations

A mechanism is provided to identify the expanded form or meaning of abbreviations.

Techniques to meet the criteria

If the abbreviation has a unique meaning within the website:

- Provide the expansion or explanation the first time it appears:
 - offering the expanded form immediately before or after, e.g., WAI (*Web Accessibility Initiative*); or
 - linking to its definition; or
 - using the *abbr* element.
- Provide the expansion or explanation each time it appears:
 - linking to its definition; or
 - using the *abbr* element; or
 - offering a glossary; or
 - providing a search functionality in an online dictionary.

If the abbreviation is used with different meanings within the same web page:

- Provide the expansion or explanation each time it appears:
 - linking to its definition, or
 - using the *abbr* element.

Reading level

When text requires reading ability more advanced than the lower secondary education level (after removal of proper names and titles), then supplemental content, or a version that does not require reading ability more advanced than the lower secondary education level, is available.

The level of education established in this criterion is between 7 and 9 years of schooling (as a reference, it typically corresponds to a level of education between 12 and 15 years old).

Techniques to meet the criteria

- Offer a summary that can be understood by people with a reading level of secondary education (between 7 and 9 years of schooling).
- Offer drawings, photographs, and symbols that help explain ideas, events, and processes.
- Offer a spoken version of the text.
- Make the text easy to read.
- Offer a sign language version for the information, ideas, and processes that must be understood to use the content.



In the chapter [Validation tools](#) you will find tools that will help you assess the difficulty of reading a text.

You can find recommendations for making easy-to-read documents in the chapter [Caring for the words](#) in this book.

Pronunciation

A mechanism is available for identifying specific pronunciation of words where the meaning of the words, in context, is ambiguous without knowing the pronunciation.

Techniques to meet the criteria

- Offer the pronunciation immediately after the word.
- Link to pronunciations.
- Offer a glossary that includes the pronunciation of words whose meaning depends on the pronunciation.
- Offer pronunciation using [the HTML ruby element](#)⁶³, which adds small help texts next to a base text and indicates pronunciation or makes small comments, as in the following example. Don't confuse the *ruby* element of HTML with the programming language of the same name.

```
<ruby>  
<rb>WCAG</rb>  
<rp>(</rp><rt>Wuh-KAG</rt></rp>)</rp>  
</ruby>
```

Wuh-KAG
WCAG

- Offer pronunciation using standard diacritical marks, but that can be disabled. In some languages, the written translation of oral language is challenging and can be presented in a variety of ways, such as '*pinyin*' for Chinese or the [marked diacritics 'okina' and 'kahako' for Hawaiian](#)⁶⁴. Diacritical marks may be completely necessary for the understanding of the texts but, sometimes, the chosen font does not support these characters. Therefore, it is recommended to use them, but always giving people the possibility of not showing them if their computer does not support the font used.

⁶³ <http://www.w3.org/TR/ruby>

⁶⁴ <https://www.hawaii.edu/site/info/diacritics.php>

Guideline 3.2

Make your page predictable, don't reinvent browsing standards

Make web pages appear and be handled predictably.

If a page is continually refreshing, changing content, opening new windows... without the user controlling it, it can confuse you, whether or not you have a disability.

Be coherent and consistent when designing and naming interface elements, allowing easy access to information.

On Focus

When any user interface component receives focus, it does not initiate a change of context.

A **change of context** is a major change in the content of a web page that, when done without people's consent, can disorient those who can't see the entire page simultaneously. The following are changes of context:

- **switch applications:** for example, open the mail manager or a PDF document reader;
- **switch View:** the area where the content is viewed;
- **switch focus:** move focus from one element to another;
- **change the content** in such a way that the meaning of the page changes.

A **change of content** is not always a change of context. Changes in content as a result of displaying new or hidden content, such as a drop-down accordion; a dynamic menu; or a tab control—do not necessarily change the context unless they also produce some other change that is considered a context change, such as:

- opening a new window;
- automatically moving the focus to another component (very common when filling in bank account or date fields);
- going to another page (including making the user believe that they have gone to another page); or
- rearranging the content of a page in a meaningful way.

Techniques to meet the criteria

- Use an *activate* action (press a button, click a link) to perform a context switch instead of when the element is focused. Notice that we are talking about changes in context, not changes in content.

Techniques to improve usability

- Open new windows and tabs from a link only when necessary.
- Alert people that a new window is about to open.

Don't do it!

- Don't programmatically remove the focus from an item immediately after it receives it.

On Input

Changing the settings of any UI component does not automatically cause a context switch unless the user has been warned of that behavior before using the component.

Techniques to meet the criteria

- To initiate a context switch, offer a button that the user can press. For example, a send button, or a selection element accompanied by a button.
- If a change to a form control causes a context switch, describe what will happen first.
- Use the *onchange* event on the *select* element without causing a context switch.

Techniques to improve usability

- Alert people that a new window is about to open.

Don't do it!

- Don't send a form automatically without warning, presenting new content, when the last field of the form is filled in.
- Don't open a new window without warning when the status of a radio button, checkbox, or picklist change.

Consistent navigation

Navigation mechanisms repeated on multiple web pages within a set of web pages always appear in the same relative order each time they are repeated unless the change is caused by the user using the site themselves.

This criterion aims to predict the location of navigation mechanisms (menus, search, breadcrumbs, pagination, etc.), which is particularly useful, for example, for people with low vision.

"Same order" does not mean elements cannot be added or removed sequentially from one page to another. For example, on some pages, we can include a secondary navigation block; we have to check that the relative order of the rest of the elements is maintained.

On the other hand, this criterion refers to mechanisms that are repeated "within a **set of pages**", that is, a collection of pages that share a common purpose and are created by the same author, group, or organization.

For example, the checkout pages of an e-commerce store are functionally and visually different from the other site pages. They are, therefore, considered a distinct set of pages. This set of pages can have different navigation mechanisms, such as including a thread of steps or removing the main menu to avoid abandoning the purchase process. The navigation mechanisms must be presented in the same relative order within this set of pages. Still, this order may differ from another set of pages on the portal.

Techniques to meet the criteria

Present the repeated components on multiple web pages in the same relative order each time they appear.

Don't do it!

Don't change the order of navigation links on different pages.

Consistent identification

Components with the same functionality within a set of web pages are consistently identified.

This criterion ensures consistency in identifying functional components that appear repeatedly throughout the website. For example, if you use an arrow icon with the *alt* text "Download" plus the file name, always use the same formula. Or if an icon means one thing on one page, it shouldn't mean something different on another.

The criterion refers to components "within a **set of pages**", that is, within a collection of pages that share a common purpose and are created by the same user, group, or organization.

For example, within a web portal, there may be a blog, a set of visually different pages published by a different team of people. The consistent identification requirement applies to this set of pages, but it does not have to be consistent with that of the rest of the portal.

Techniques to meet the criteria

Use tags, names, and textual alternatives consistently (which doesn't mean they have identical texts) for content that has the same functionality. To include tags, names, and textual alternatives, follow the techniques listed in the conformance criteria "1.1.1 Non-text Content" and "4.1.2 Name, Role, Value."

Don't do it!

Don't use two different tags for the same functionality on different pages in the same set of site pages.

Change on request

Changes of context are initiated only at the user request, or a mechanism is provided to turn off such changes.

Techniques to meet the criteria

- If the webpage **allows automatic updates**, offer a mechanism to ask people to do so instead of automatically.
- If the web page **can auto-redirect**, do it on the server-side instead of the client-side; or do it on the client-side with a *meta-refresh tag* with null time or "0".
- Use the *onchange* event on the *select* element without causing a context switch.
- If a new window **needs to be opened**,
 - implement it through the *target* attribute and alert the user in the link text; or
 - use a *script* respecting the principle of progressive improvement and warn that the content will open in a new window.

An example of how to use a *script* to successfully open new windows is as follows:

1. Include this link to the "popup.js" file in the header of your website:

```
<script type="text/javascript" src="popup.js"></script>
```

2. Enter the link with an id on your website:

```
<a href="link.html" id="newwindow">Link text</a>
```

3. Now, in the "popup.js" file, include this code:

```
//We capture the event
window.onload = addHandlers;

function addHandlers() {
//We capture the link
var objAnchor = document.getElementById('newWindow');
if (objAnchor) {

    // we add the opening notice in new window
    objAnchor.firstChild.data = objAnchor.firstChild.data + '
(opens in new window)';
```

```

// We launch it in both the pointer and keyboard event
objAnchor.onclick = function(event){return NewWindow(this,
event);}
objAnchor.onkeypress = function(event){return Newwindow
(this, event);}
}

// Opening function
function NewWindow(objAnchor, objEvent){
var iKeyCode, bSuccess=false;

// We only want it to work if the keys pressed are Space
(32) or Return (13)
if (objEvent && objEvent.type == 'keypress'){
if (objEvent.keyCode)
iKeyCode = objEvent.keyCode;
else if (objEvent.which)
iKeyCode = objEvent.which;
if (iKeyCode != 13 && iKeyCode != 32)
return true;
}

// We open the window and tell it the URL
bSuccess = window.open(objAnchor.href);

// If it does not open in a new window, we tell the
browser to open the link in the same window
if (!bSuccess)
return true;

// Stop applying the code
return false; }

```

Techniques to improve usability

- Open new windows and tabs from a link only when necessary.

Don't do it!

- Don't open windows that people have not requested.
- Don't open windows as soon as the content loads.
- Don't open windows when users enter text in an input field.
- Don't change core content using automatic updates that people can't disable with an in-content mechanism.
- Don't cause a context switch when users take focus away from a form element.
- Don't refresh or redirect the page using tags such as *http-equiv="refresh"* meta with a time limit.

Consistent Help

If a web page contains any of the following help mechanisms that are repeated on multiple web pages within a set of web pages:

- human contact details (phone, email, opening hours),
- human contact mechanisms (chat, contact form, social networks),
- self-help option (FAQ page),
- fully automated contact mechanism (a *chatbot*),

they occur in the same relative order to other page content (unless the user initiates a change).

Help mechanisms can be provided directly on the page or through a direct link to a different page containing the information.

The purpose of the criterion is not to require that help options exist but to ensure that if they exist, they can be found in a consistent location on all pages.

This placement refers to the **order of the content when serializing the page's DOM**; however, it is recommended that the visual position of this help mechanism be consistent throughout the website as well. If the help element is visually in a different location but in the same relative order, it will not violate this criterion, but it will be less useful and usable, thus worsening the user experience.

The location may change in the responsive version. This criterion refers to the relative order of contact and help items between pages displayed in the same page variation (for example, the same zoom level and orientation).

Moreover, some websites consist of multiple different sets of pages for various purposes. This conformance criterion allows **different sets of web pages to use different helper locations**. However, it is best if the helper mechanisms are located as consistently as possible, even between different sets of related web pages.

Techniques to meet the criteria

- Provide a "Contact Us" link consistently across all pages.

Don't do it!

- Don't include help in an inconsistent location.

Guideline 3.3

**Help users to input data.
To err is human.**

Help people avoid and correct mistakes.

We all make mistakes, and it won't be any different on the internet. Therefore, we must prevent people from making mistakes and, if they do, help them overcome them and continue with their activity.

Most errors occur in forms, so pay special attention when designing them.

Error Identification

If an error in data entry is automatically detected:

- the erroneous element is identified, and
- the error is described to the user through text.

Techniques to meet the criteria

If the user hasn't filled in the required fields on a form:

- Provide a textual description that identifies the required fields that have not been completed.
- Use the *aria-invalid* attribute to indicate fields that have failed.
- Validate the form on the client side and display an *alert()*.

If the information provided by the user must follow a specific format or must be in a certain range of values:

- Use ARIA (*live regions*, *role="alertdialog"*, *role="alert"*) to notify assistive products, such as a screen reader, of data entry errors.
- Use the *aria-invalid* attribute to indicate fields that have failed.
- When the user includes a value not on the list of allowed values, notify them with a text message explaining it. If possible, include the list of values or suggest the allowed value most similar to the value entered.
- When the user includes data that does not comply with the required format or values, notify them with a text message explaining it. You must provide:
 - correct examples, or
 - describe the correct entry, or
 - display correct values similar to those entered by the user, with instructions on entering one of these correct values if the user chooses to do so.
- Validate the form on the client side and add an error message in the DOM or with an *alert()*.

Techniques to improve usability

- Provide a mechanism for people to navigate from the error message to the field and vice versa.
- Alert the user whether the form has been successfully submitted or not.

Labels or instructions

Labels or instructions are provided when the content requires human input.

Techniques to meet the criteria

- Offer descriptive tags and one of the following options:
 - uses *aria-describedby* to describe interface controls;
 - uses *aria-labelledby* to define the label by concatenating texts from several nodes;
 - uses ARIA roles (*role="group"*, *role="radiogroup"*) to identify related form controls;
 - indicate the expected format of the data and give an example;
 - provides textual instructions at the beginning of the form, or a group of fields, describing the required entry;
 - position labels in such a way as to help predict relationships;
 - add a textual description to identify the required fields not filled in;
 - indicates the required fields using the *label* or *legend* label;
- Use the *label* element to associate labels with controls
- Describe field groups using *fieldset* and *legend* elements
- Use a button adjacent to a field to explain the purpose of that field. This technique is considered a last resort, as it should only be used when the other techniques cannot be applied on the page.

Techniques to improve usability

- If a change to a form control causes a context switch, describe what will happen first.

Don't do it!

- Don't visually format a group of fields intended to enter a phone number (prefix, number, and extension) without using a text label for each field.

Error Suggestion

If an error in the data entry is automatically detected and suggestions for correction are available, then the suggestions are presented to the user, unless this would jeopardize the security or purpose of the content.

Techniques to meet the criteria

If a field's information is required to be in a specific data format:

- Use ARIA's *role="alertdialog"* to notify assistive products, such as a screen reader, of data entry errors.
- When the user includes data that does not comply with the required format or values, notify them with a text message explaining it. You must provide:
 - correct examples, or
 - describe the correct entry, or
 - display correct values similar to those entered by the user, with instructions on how to input one of these correct values if the user chooses to do so
- Suggest a way to fill in the field correctly.

If the information provided by the user is required to be in a specific range of values:

- Use ARIA's *role="alertdialog"* to notify assistive products, such as a screen reader, of data entry errors.
- When the user enters a value not on the list of allowed values, notify them with a text message explaining the error. If possible, include the list of values or suggest the allowed value most similar to the value entered.
- Suggest a method for filling in the field correctly.

Techniques to improve usability

- Provide a mechanism for people to navigate from the error message to the field and vice versa.
- Alert users whether the form has been submitted successfully or not.
- Validate the form on the client side and add an error message in the DOM or with an *alert()*.

Error Prevention (Legal, Financial, Data)

For websites that

- represent legal commitments or financial transactions for the user;
- modify or delete data stored in systems that the user can control; or
- send the user's answers to a test,

at least one of the following is true:

- **reversible:** The user can undo or cancel the submission.
- **checked:** The system detects errors in data entry and allows them to be corrected.
- **confirmed:** A mechanism is provided for the individual to review, confirm, and correct the information before submitting the data.

Techniques to meet the criteria

If an action by the user causes a legal or commercial transaction, such as filing a tax return or making a purchase, choose an option:

- **Before sending**, the user can review and correct their answers; or they must select a checkbox indicating that they have reviewed the data and want to carry out the transaction, which will be done with a send button.
- **After submission**, give the user time to modify or cancel the action.

If an action by the user is going to cause a deletion of information, choose an option:

- **Before sending**, the user must confirm the action to continue; or they must select a checkbox to indicate that they have reviewed the data and are sure they want to delete the information, which will be done with a send button.
- **After submission**, the user can retrieve the deleted information.

If the website includes a test or exam:

- **Before submission**, the user can review and correct their answers; or is asked to confirm to proceed with the selected action.

Techniques to improve usability

- Alert the user whether the form has been successfully submitted or not.
- Validate the form on the client side and display errors to the user with an *alert()*.

Help

Context-sensitive help is provided.

Techniques to meet the criteria

If a form requires text input:

- Provide a help link on each web page.
- Offer help through an assistant, that is, a multimedia avatar.
- Check the spelling of text entered in the field and offer text suggestions. For example, by means of a message "Did you mean...?" with a link to the suggested word.
- Provide textual instructions at the beginning of the form or a group of fields that describe the required input.

If a form requires text input in a particular data format:

- Indicate the expected format of the data and give an example.
- Provide textual instructions at the beginning of the form or a group of fields that describe the required input.

Techniques to improve usability

- Use the *title* attribute to present contextual help.

Error Prevention (All)

For web pages that require the user to submit information, at least one of the following is true:

- **reversible:** The user can undo or cancel the submission.
- **checked:** The system detects errors in data entry and allows the user to correct them.
- **confirmed:** A mechanism is provided for the individual to review, confirm, and correct the information before submitting the data.

Techniques to meet the criteria

Simply follow the techniques [in criterion 3.3.4](#) for all forms on which the user submits information.

Redundant entry

Information previously entered by the user, or provided to the individual, and which must be entered again in the same process, may be auto-filled or available for the individual to select.

Exceptions:

- Re-entering the information is essential.
- The information is necessary to ensure the security of the content (e.g., including the new password twice).
- The information previously entered is no longer valid.

This criterion aims to ensure that people can successfully complete multi-step processes. It reduces cognitive and physical effort when information is requested more than once during a process and the need to remember information provided in a previous step.

For example, if you have fields to enter shipping and billing details in a checkout process, offer the possibility to populate the billing details with the shipping data included.

It should be noted that this criterion always refers to the same process; it does not add the obligation to store information between sessions.

Techniques to meet the criteria

- Provide data from a previous step in other steps in the process.
- Don't ask for the same information twice.

Accessible authentication (Minimum)

A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step of the authentication process.

Exceptions:

- An **alternative authentication method** is provided that does not rely on a cognitive function test.
- A **mechanism is available to assist users in completing the cognitive function test**, such as password manager support or the ability to copy and paste a password.
- The cognitive function test consists of **recognizing objects** (they can be images, videos, or sounds).
- The cognitive function test aims to **identify non-text content** that the user provided to the website (it can be images, videos, or sounds).

This criterion's goal is to ensure the availability of a login and access method that is accessible, easy to use, and secure, with a particular focus on improving accessibility for individuals with cognitive disabilities. The authentication process includes any captchas and the option to recover passwords.

A "cognitive function test" is a task that requires the user to remember, manipulate, or transcribe information, such as:

- Memorization includes remembering a username, password, character set, pictures, or patterns.
- Character transcription, such as typing text or code without being able to copy it.
- Use of correct spelling.
- Performing calculations.
- Puzzle solving.

Including your name, email, or phone number **is not considered a cognitive function test**, as they are personal to each user and consistent across all websites.

A **common mistake** is to force you to enter a password or code in a format other than the original, for example, to include only the characters of certain positions, as shown in the following image:

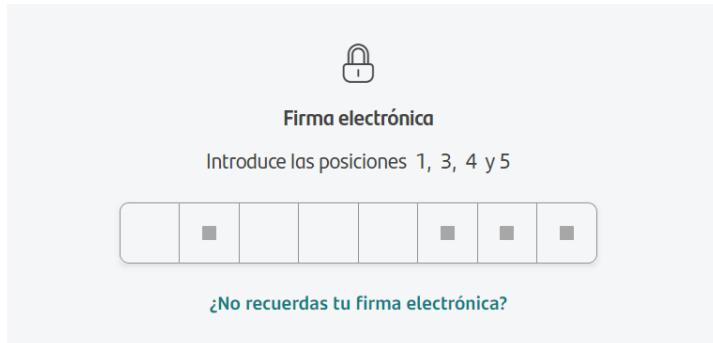


Illustration 32 Request for specific items in an electronic signature

Techniques to meet the criteria

- Facilitate authentication without a password through a link sent via email.
- In HTML, mark the email and password fields correctly.
- Allow the use of [WebAuthn](#)⁶⁵ or [oAuth](#)⁶⁶ as alternatives to username and password:
 - WebAuthn is a web authentication standard allowing people to log into websites and apps using biometric data, such as fingerprint or facial recognition.
 - oAuth is an open protocol that enables secure authorization using a simple, standard method from web, mobile, and desktop applications.
- Use 2-factor authentication with two different techniques.

Don't do it!

- Don't prevent entering the password or code in the original format in which it was created.

⁶⁵ <https://webauthn.io/>

⁶⁶ <https://oauth.net/>

Accessible Authentication (Enhanced)

A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step of an authentication process.

Exceptions:

- An **alternative authentication method** is provided that does not rely on a cognitive function test.
- There is a **mechanism** available to help the user complete the cognitive function test (such as password manager support or the ability to copy and paste a password)

This criterion is a stricter version of the previous criterion. Cognitive function tests involving object recognition or identification of non-text personal content are not accepted at level AAA.

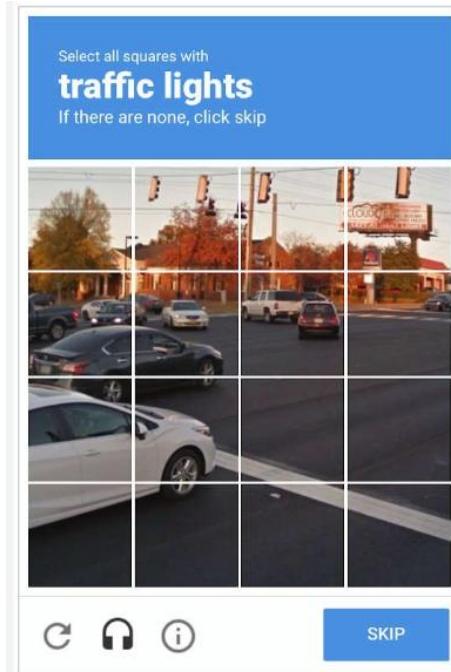


Illustration 33 Object-recognition-based captchas are supported at level AA (criterion 3.3.8) but not level AAA (criterion 3.3.9)

Techniques to meet the criteria

Follow the same techniques, and avoid the same errors, as in criterion 3.3.8.

Principle 4.

Robust

This is the principle most dependent on technology.

It refers to the website's ability to be interpreted by user agents, supporting products, and devices of all kinds, as well as current and future versions.

Guideline 4.1

Respect and care for the code

*Maximize compatibility with current and future user applications,
including assistive products.*

Our code should be as clean as possible and respect the standards. This way, browsers and other user agents can render the page correctly.

This guideline and its checkpoints are directly intended for developers.

Parsing

This criterion was originally adopted to address the problems that assistive products had when parsing HTML directly. However, assistive products no longer need to parse HTML directly, as they communicate with the browser via the accessibility API. Consequently, these issues no longer exist or are addressed by other criteria. Therefore, this criterion is no longer helpful and has been officially removed from WCAG 2.2.

However, we recommend following their advice as it makes it easier for browsers to parse the code.

In the content implemented using markup languages:

- the elements have the complete start and end tags;
- the elements are nested according to their specifications;
- the elements do not contain duplicate attributes; and
- any IDs are unique.

Exception:

- When the specifications allow these characteristics.

Techniques to meet the criteria

- Validate that you fully meet the specification. You can use the automatic HTML and CSS code validators referenced in the [Validation Tools chapter](#).
- Choose the standard you want and follow its specification, ensuring that the website can be processed because it is well-formed or at least:
 - Start and close labels are used as per specification.
 - Each *ID* attribute has a different value on the web page.
 - Elements do not have repeated attributes.

Don't do it!

- Don't make mistakes in opening or closing labels or marking attributes.
- Don't use repeated identifiers.

Name, Role, Value

For all UI components (such as form elements, links, *scripted* components, and more):

- the name and function may be programmatically determined;
- states, properties, and values that can be adjusted by people are set by software using methods that are supported by user agents, including assistive technologies; and
- notification of changes to these items is available to user agents, including assistive technologies.

This criterion primarily aims at developers who program their UI components or modify standard controls. Standard HTML controls automatically satisfy this criterion when used according to the specification.

Techniques to meet the criteria**If you use standard UI components in a markup language (for example, HTML):**

- Use the *aria-label* attribute to provide an invisible label when you can't use a visible label.
- Use the *aria-labelledby* attribute to provide a name for UI controls.
- Use the characteristics of the markup to expose the name and function; allow properties that can be configured by people to be set directly; and warn of changes; by applying:
 - standard HTML form controls and links;
 - *label* elements to associate text labels with form controls; or the *title* attribute to identify controls if you can't use the *label* element;
 - the *title* attribute of the *iframe* and *frame* elements;
 - HTML as specified.

If you use code to reuse a standard UI component in a markup language:

- Expose names and roles, allow user-configurable properties to be set directly, and alert you to changes. You can use *aria-labelledby* to provide a name for UI controls.

If you use a standard UI component with a programming technology:

- Use accessibility API features to expose names and notifications of changes.

If you create UI components using a programming language:

- Use technologies that support notification of accessibility of changes, like:

- ARIA roles, states, and properties to expose information about UI components.
- The *aria-labelledby* attribute to provide a name for the UI controls.

Don't do it!

- Don't code *div* or *span* elements to behave like UI controls without giving them a role.
- Don't implement custom controls that don't use the Accessibility API or do so incompletely. WAI-ARIA can expose a custom control's function, name, value, states, and properties through the Accessibility API.
- Don't forget to update the textual alternatives when the non-text content changes.
- Don't forget that UI controls need a name that can be determined programmatically.
- Don't place focus on a UI component in such a way that it cannot be determined by software or in such a way that the focus status change notification is unavailable.
- When entering information in several separate fields, such as a phone number or bank account number, don't forget to provide names for each field.
- Don't forget to give an image an accessible name when it's the only content in a link. You can do this with the *alt*, *aria-label*, or *aria-labelledby* attribute.

Status messages

In content implemented using markup languages, status messages can be programmatically determined through their role or properties so that they can be presented to the user of assistive technologies without receiving focus.

Status messages inform people about the outcome of an action, the wait status of an application, the progress of a process, or the existence of errors.

To be considered a status message, **it cannot be delivered through context switching or capture focus**. This criterion does not affect important information that is presented in modal dialogues, which must be recognized by the user immediately and must capture focus.

For example, they are status messages:

- When you click on an "Add to shopping cart" button, a text appears next to the cart that reads "Cart: 1 item".
- When entering an email in a form field in error, an error message appears warning that "The email format is incorrect".
- When a process takes a while, a 'loading' icon appears on the screen.
- When you submit a form, a floating message is displayed announcing "Form Submitted."
- The list of search results is not considered a status message; however, the short message "18 results found" did.

The aim of this criterion is to ensure that all people receive this information and are aware of significant changes in content without unduly interrupting their work. In addition, people who use assistive products can delay or delete those status messages or, on the contrary, can highlight them when they require it.

Techniques to meet the criteria

If the status message reports on the success or outcome of an action or the status of an application:

- Indicate that the data was sent successfully in combination with the use of *role="status"*.

If the status message conveys a suggestion or warning about the existence of an error:

- Use ARIA's *role="alert"* or *live regions* to identify errors in combination with one of these techniques:
 - provide text descriptions to identify unfilled required fields;

- provide text descriptions when fields do not conform to permitted or required values or formats;
- offer suggestions for correction;
- provide a spell check and suggestions for data entry.

If the status message conveys information about the progress of a process:

- Use the *log* role to identify updated sequential information.
- Use the *progressbar* role.
- Use the *status* role to display status messages and provide help through a wizard on the web page.

Techniques to improve usability

- Use *aria-live* in chats or in content displayed with *hover* and *focus*.
- Use the *marquee* or *timer* roles.
- When appropriate, move the focus to the new content included by using *alertdialog* to identify errors.
- It offers customization options that support non-essential alerts to be optional.

Don't do it!

- Don't provide status messages that their roles and properties cannot programmatically determine.
- Don't use *role="alert"* or *aria-live="assertive"* on content that isn't important or time-sensitive.

Accessible PDF documents

Files linked from a web page for viewing or downloading, such as spreadsheets or PDFs, are part of the site's content and must meet the same accessibility requirements as web pages.

Due to its standardization and popularity, this chapter outlines the steps for making a PDF accessible and provides guidance on evaluating its accessibility.

Accessibility in PDF documents

The PDF format (Portable Document Format) allows you to combine text, videos, sound, links, forms, and others. This content, like the content on a web page, must comply with the principles of WCAG 2.2.

As explained in the introduction to this book, PDF technology supports accessibility. This means that if the PDF document has been produced in an accessible way, assistive products such as a screen reader or braille displays can understand and provide the information contained in it. Thus, if the PDF document is natively accessible, you wouldn't have to provide an alternative HTML version.

Additionally, specific technologies cannot be excluded from the declarations of conformance. For example, we cannot say that the website is double-A accessible to WCAG 2.2 except for PDF documents. Therefore, if our PDF documents are not accessible, our website will not be accessible either.

The WCAG 2.2 compliance criteria are written as verifiable statements independent of a particular technology so that they can be applied to web pages as well as to any electronic document. Although the compliance criteria that PDF documents must meet are the same as those that websites must meet, we can distinguish between:

- Criteria **that do not apply** to PDF documents, such as the need to be able to increase the text size, since PDF technology supports zooming.
- Criteria **that apply and are met with general techniques**, regardless of the technology used, such as color contrast requirements, not conveying information solely by color, not giving instructions that rely on sensory characteristics, visual presentation, dividing content into sections with titles or making them descriptive, etc.
- Criteria **that apply and are met with techniques specific** to PDF technology. Most of these specific techniques **will not produce visual changes** in the document, as they are designed to help especially people who access it with a screen reader or a braille display. These techniques also have additional benefits, such as improving the indexing and **positioning** of the document in search engines, as well as enhancing its **usability** for all users.

PDF-specific techniques

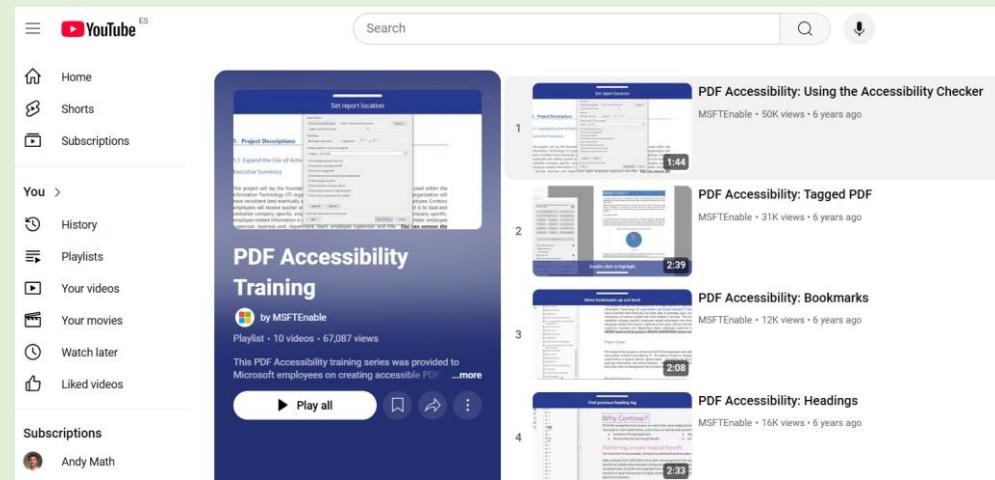
WCAG 2.2 includes 23 techniques specific to PDF technology.

1. **Add alt text to images** so that people without vision can perceive them.
2. **Create bookmarks** so everyone can scroll and find information quickly.
3. **Ensure the correct reading** and tabbing order so that the content makes sense if accessed linearly, such as with a screen reader.
4. **Hide decorative images** so as not to disturb or hinder understanding.
5. **Indicate the required form controls** to prevent people from making mistakes.
6. **Label tabular data** using table elements (Table, TR, TH, and TD) so screen readers can read it in an organized manner.
7. **Perform OCR** on a scanned document to get the actual text; otherwise, you get an image that the assistive products can't interpret.
8. **Define abbreviations** by expanding them in the text or using alt text on their labels so that a screen reader can read their extended form.
9. **Include headings** marked as such to understand the organization of the contents and so that the user of a screen reader can "skim" the document.
10. **Label interactive form controls** so users can understand what each control is for.
11. **Incorporate links** labeled as such with descriptive text so that the user can decide whether or not to click and understand where the link is going.
12. **Indicate the name, function, and value** of form fields so it will be easy to know they should be filled out, especially when accessed with a screen reader.
13. **Offer alt text in links** when your text isn't sufficiently explanatory.
14. **Introduce headers and footers** on each page to help users find their way around.
15. **Add a "Submit" button to forms** so people can decide when to submit.
16. **Set the default language** so the screen reader can pronounce it.
17. **Visually number pages**, for example, in the document's footer, in a manner consistent with their numbering in the PDF viewer controls.
18. **Give the document a title**; it is very useful for understanding its content
19. **Specify the language** of a paragraph or sentence when it differs from the language of the document.
20. **Repair mislabeled tables** with Adobe Acrobat Pro's table editor.
21. **Use the L and LI tags** in lists so users can scroll through content quickly.
22. **Alert people when they make a mistake** when filling out a field because it must follow a certain format or enter values within a range.
23. **Offer interactive form controls** ensuring they work properly when accessed with the keyboard only.

Implement PDF techniques

Implementing these techniques involves working on both the source document and the PDF itself. An accessible **source document** helps to make the resulting PDF much more accessible when saved or exported. Unfortunately, this is not always enough, especially in complex documents, and it is often necessary to **modify the PDF** with a specific editing tool to correct certain issues. These modifications may be minimal or even unnecessary if good practices have been followed in the source program and the document is simple.

Here's how to implement the techniques in the source program and a PDF editor, how to export to PDF without losing the accessibility improvements made to the source document, and how to review the PDF document for accessibility.



The screenshot shows a YouTube interface with a sidebar on the left and a main content area. The sidebar includes links for Home, Shorts, Subscriptions, You (History, Playlists, Your videos, Your movies, Watch later, Liked videos), and Subscriptions. The main content area displays a playlist titled "PDF Accessibility Training" by "MSFTEnable". The playlist has 10 videos and 67,087 views. The first video in the list is "PDF Accessibility: Using the Accessibility Checker". Below the video list, there are four smaller thumbnail images labeled 1 through 4, each representing a different aspect of PDF accessibility: 1. Project Description, 2. PDF Accessibility: Tagged PDF, 3. PDF Accessibility: Bookmarks, and 4. PDF Accessibility: Headings.

We recommend the playlist "[PDF Accessibility Training](#)" on YouTube by Microsoft.⁶⁷

To modify the source document, we have chosen Microsoft Word (although sometimes Open Office, Google Docs, or InDesign are also mentioned), and to modify the PDF itself we have chosen Adobe Acrobat Professional. The steps are common to other programs (word processors, spreadsheets, design programs...), but they are carried out through different menu options.

⁶⁷ <https://www.youtube.com/playlist?list=PLtSVUgxlo6KrxMh-wNhSG7MuZ2gfPf7co>

Adobe Acrobat Professional⁶⁸ is currently the most recommended tool for reviewing and modifying the accessibility of a PDF. However, it is proprietary and paid software that can be installed locally in a computer, different from Adobe Acrobat Reader. Although PDF accessibility can be worked on from the Adobe Acrobat Pro 6 version, it is recommended to use the DC version due to the significant improvements in its specific accessibility tools.

There is another tool, CommonLook PDF GlobalAccess⁶⁹, which is also local, paid, and quite powerful. It allows you to review and modify certain aspects of the PDF.

As a free alternative, you can try PAVE,⁷⁰ an online application that allows you to open and evaluate a PDF and correct some of its accessibility issues.

The screenshot shows the PAVE web interface. At the top, there's a navigation bar with the PAVE logo, a graduation cap icon labeled 'ZHAW', an envelope icon labeled 'CONTACT', a question mark icon labeled 'HELP/FAQ', and a globe icon labeled 'LANGUAGE'. Below the navigation is a dropdown menu showing 'acuseRecibo.pdf'. The main area has tabs for 'TASKS', 'PROPERTIES', 'ISSUE DETAILS', and 'READING ORDER', with 'ISSUE DETAILS' being the active tab. A green box titled 'Automatic Correction' contains the message 'PAVE was already able to automatically correct 122 issues.' A pink box titled 'Language is not specified' contains the message 'The document's properties contain information such as the document's title and language. Some of these are mandatory. To fix these issues, correct the fields in the document properties.' At the bottom of this section is a button labeled 'EDIT PROPERTIES'. To the right of the main content area, a small preview window titled 'Page 1' shows a snippet of the PDF document.

Screenshot 5 Results of a PAVE assessment

⁶⁸ <https://www.adobe.com/acrobat.html>

⁶⁹ <https://commonlook.com/accessibility-software/pdf/>

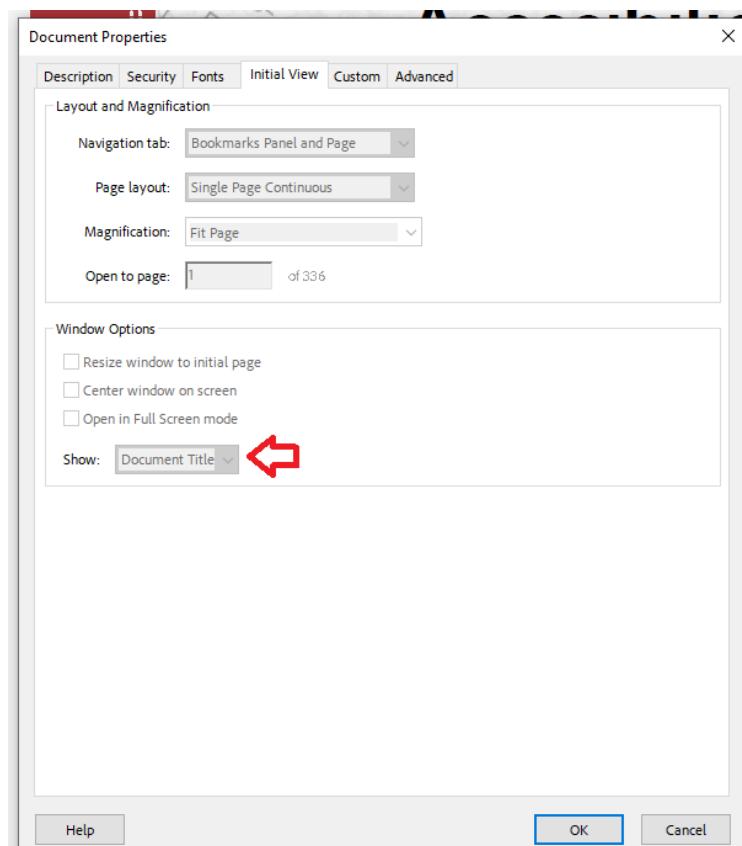
⁷⁰ <http://pave-pdf.org>

Title of the document

The document title is the first thing the screen reader announces to the user, who can consult it anytime with a keyboard shortcut. In most source programs, you can indicate the document's title in their properties so that it will be exported as the title of the PDF.

In **Word**, the title of the document is defined in its properties in the "File > Information > Properties > Title" menu. You should keep in mind that in Word, the title of the document is never shown in the title bar, but always the name of the file, and it will be this that the screen reader announces, but the title you include in the document properties will be exported to the PDF.

In **Acrobat**, you can include or modify the title in the "File > Properties > Description > Title" menu. In addition, you must indicate that it is the title, not the file's name, displayed in the PDF reader's title bar. This is defined in the menu "File > Properties > Initial View > Show > File Name".



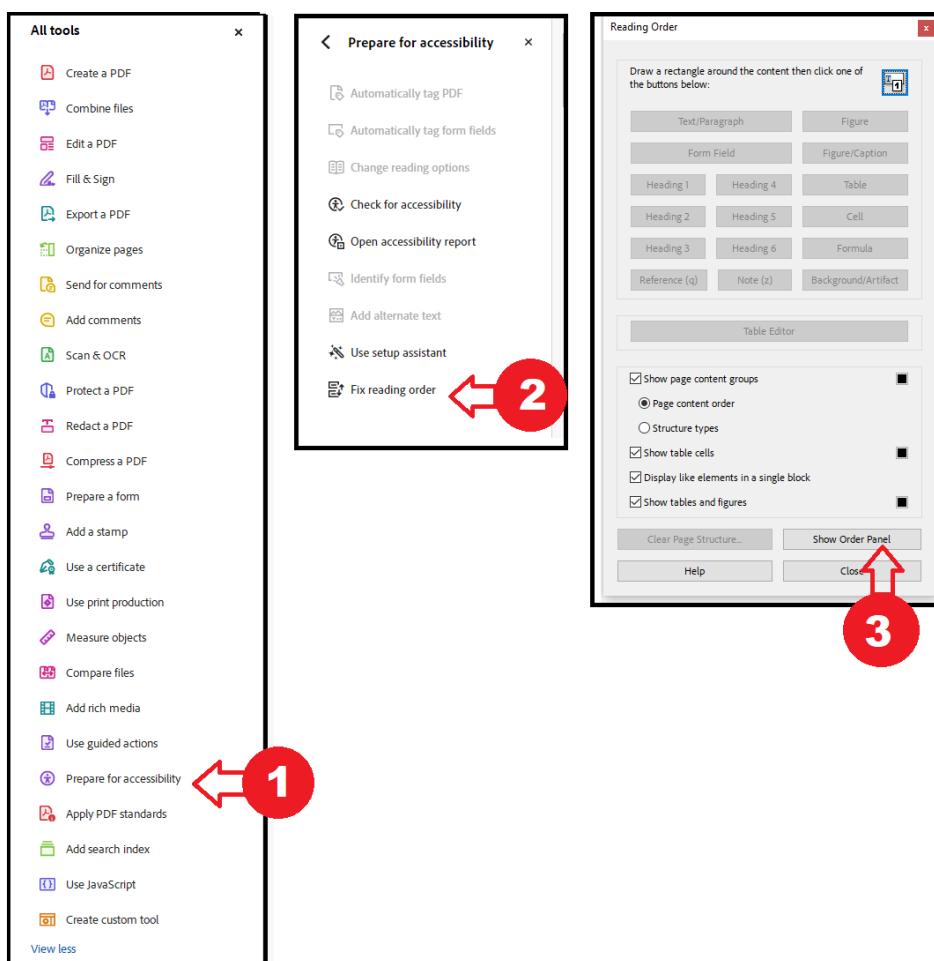
Screenshot 6 "Initial View" of a document's properties in Adobe Acrobat Professional DC

Reading order

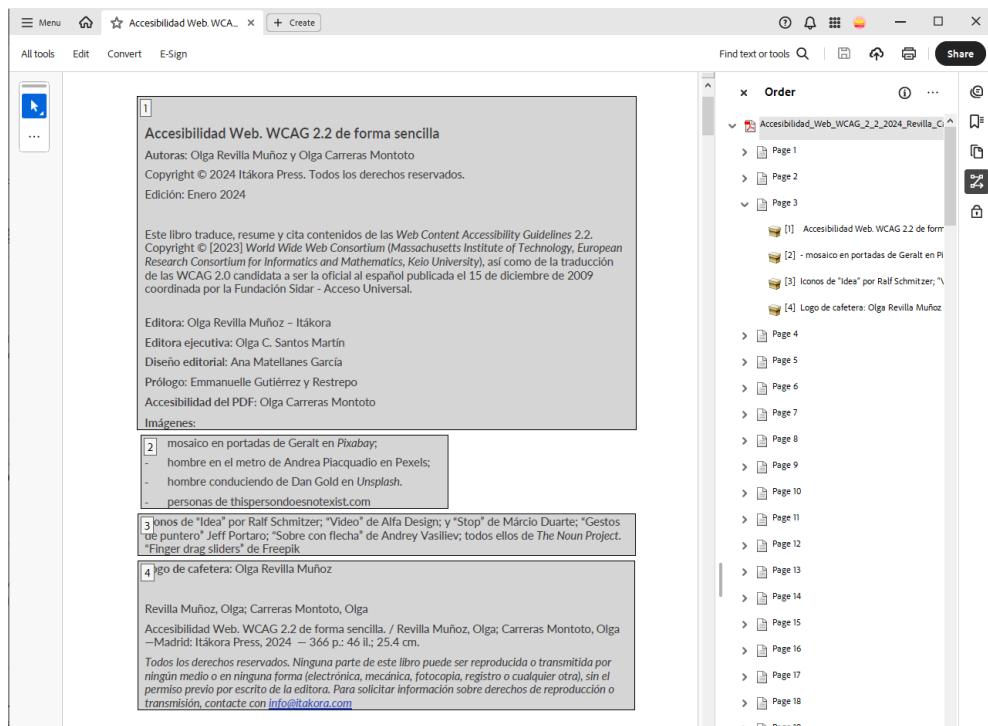
The document's contents' visual order often does not match the assistive product's reading order. The meaning may change or become incomprehensible when the content is read in another order.

In the Acrobat reading options, the screen reader user can decide whether the content is read according to the order defined in the "Order" panel or the "Tags" panel, so we must ensure that the order is correct in both panels.

When we activate the "Order" panel, or the "Prepare for Accessibility > Fix Reading Order" tool, each content element on the page is displayed with a number, which indicates its reading order.



Screenshot 7 Process to show the "Order" panel at Adobe Acrobat Professional



Screenshot 8 Adobe Acrobat Professional "Order" panel

The order can be changed by reordering the contents in the tree in the "Order" panel: select an item and drag it to another position in the tree so that the numbering of the contents changes to reflect its new reading order.

The incorrect reading order is usually a consequence of how the content has been laid out in the source program. A series of **good practices in Word** so that the reading order in the PDF is correct and does not have to be modified, are:

- Don't use floating frames.
- If you are mocking up in multiple columns, do not simulate them with tabs, floating boxes, or borderless tables, but use the "Columns" tool.
- Insert images as you compose the text, aligning them with the content.

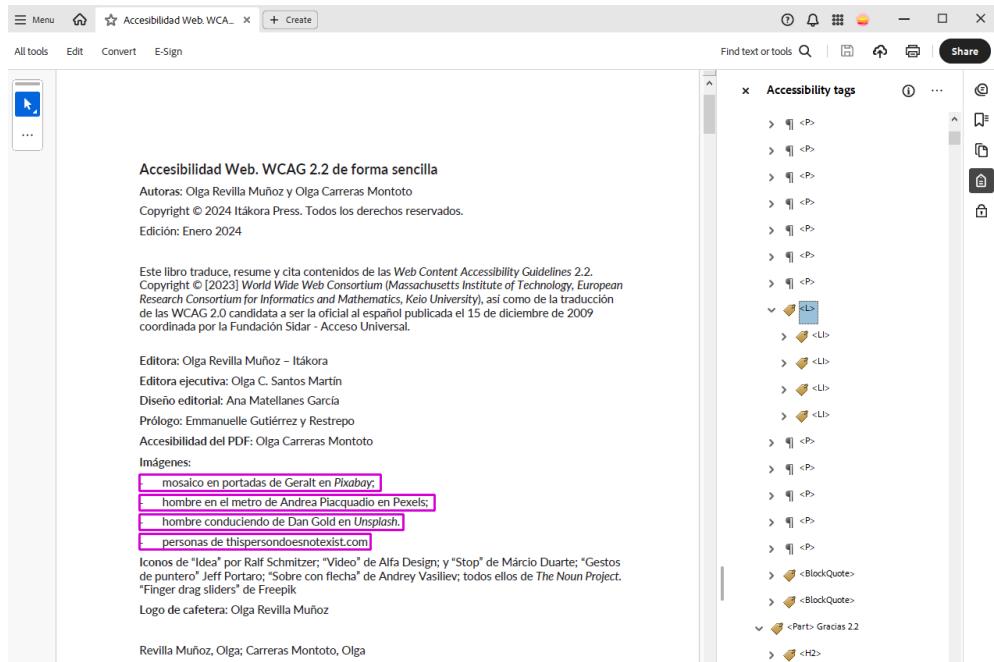
In **InDesign**, the reading order is determined by the order in which the boxes were created. This order can be modified from the "Layers" panel, sorting the elements from bottom to top.

If you change the reading order and an item disappears and you don't know why or how to fix it, check out the different layers of content, the most usual problem is that one layer is covering the content of other.

Semantic tagging

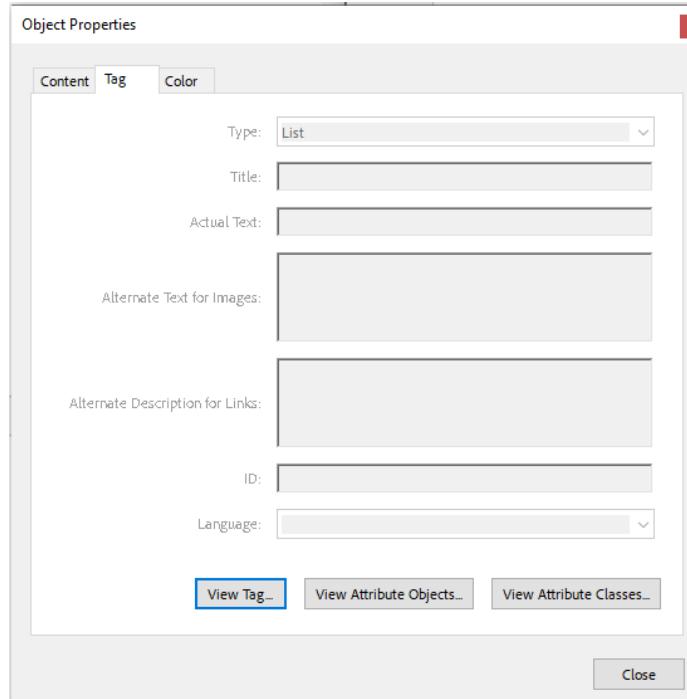
Each piece of content should have an internally associated tag that indicates what type of content it is: a heading, paragraph, list, table, image, etc. This allows, among other things, the screen reader to correctly announce each content according to the tag associated with it, such as announcing that it is a list with four items or that it is a level 1 heading. In addition, it makes it easier for users of a screen reader or braille display to "skim" the document and jump from one content to another using the shortcuts of the assistive product.

In Acrobat Professional, the tag tree is located in the "Tags" panel. If the document has not been generated by tagging, the tag tree can be created automatically from the dashboard. However, to avoid problems, it is recommended that the tagged PDF always be generated from the source program.



Screenshot 9 Adobe Acrobat Professional "Tags" panel

You can create and delete tags from the tree in the "Tags" panel, or access their properties and specify, among other things, what type they are.



Screenshot 10 Tag properties in Adobe Acrobat Professional

As we have already mentioned, screen reader users can choose to have the reading order follow the labeling order; therefore, **the sequence of tags in this tree must also be arranged appropriately to reflect the order in which the content should be read.**



In order to review and correct the labeling, you need to be familiar with the [standard PDF tags](#).⁷¹

If the document is laid out with the appropriate tools in the source program (headings, lists, tables, table of contents, etc.), the document will be labeled correctly, and it will hardly be necessary to retouch it in Acrobat.

Good practices to be followed in the source program are:

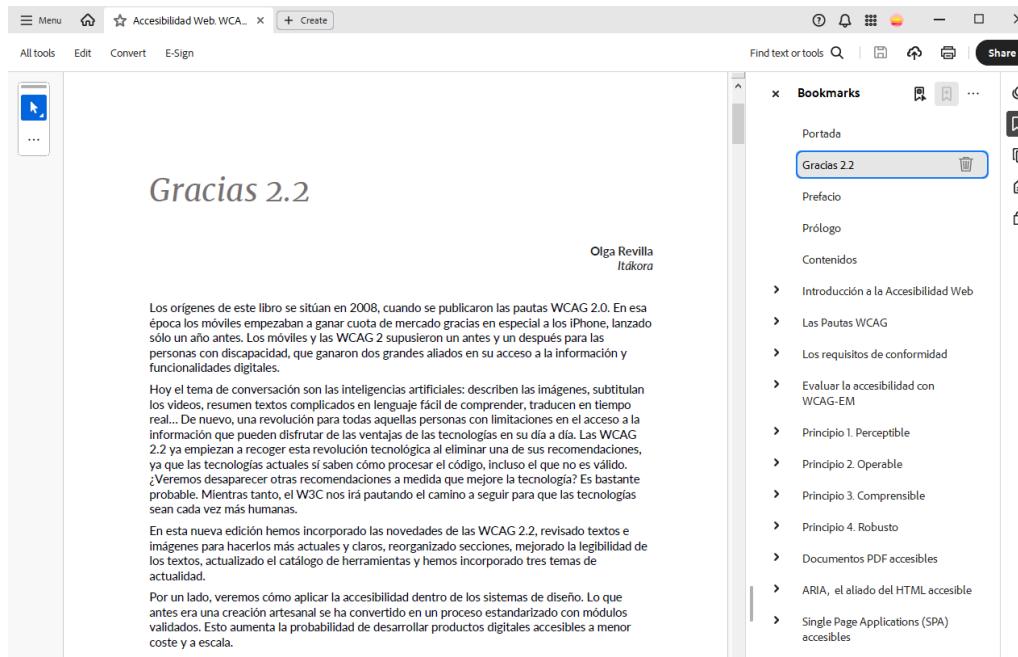
- Define titles using title styles, never simulate them by simply changing their size or color. In InDesign, you must create paragraph styles and specify the PDF export tag in their properties by going to "Paragraph Style Options > Export Tagging > PDF > Tag".
- Assign titles consistently and not based on how they look. Don't skip levels.
- Define lists using the Bulleted List or Numbered List tool.

⁷¹ <https://helpx.adobe.com/acrobat/using/editing-document-structure-content-tags.html>

- Don't include carriage returns to separate paragraphs or skip pages, as they are exported as empty paragraphs that confuse screen reader users. The separation between the contents must be defined by their margins, spacing, and line spacing. Similarly, page breaks must be included with the Page Break tool.
- If you include notes, an index of contents, bibliographic references, or other similar content, it should always be done with the corresponding tool.

Bookmarks

The "Bookmarks" panel in Acrobat Professional includes an index of the document generated from its headings. This panel is especially useful because it allows you to understand the document's structure and jump to any section within it.



Screenshot 11 Adobe Acrobat Professional Bookmarks panel

In the PDF export options, you can indicate that the PDF is generated with the bookmark index, or it can be automatically generated from the "Bookmarks > New Structure Bookmarks" panel.

Alternatively, in the document properties, you can specify that the bookmark index is visible by default when the document is opened.

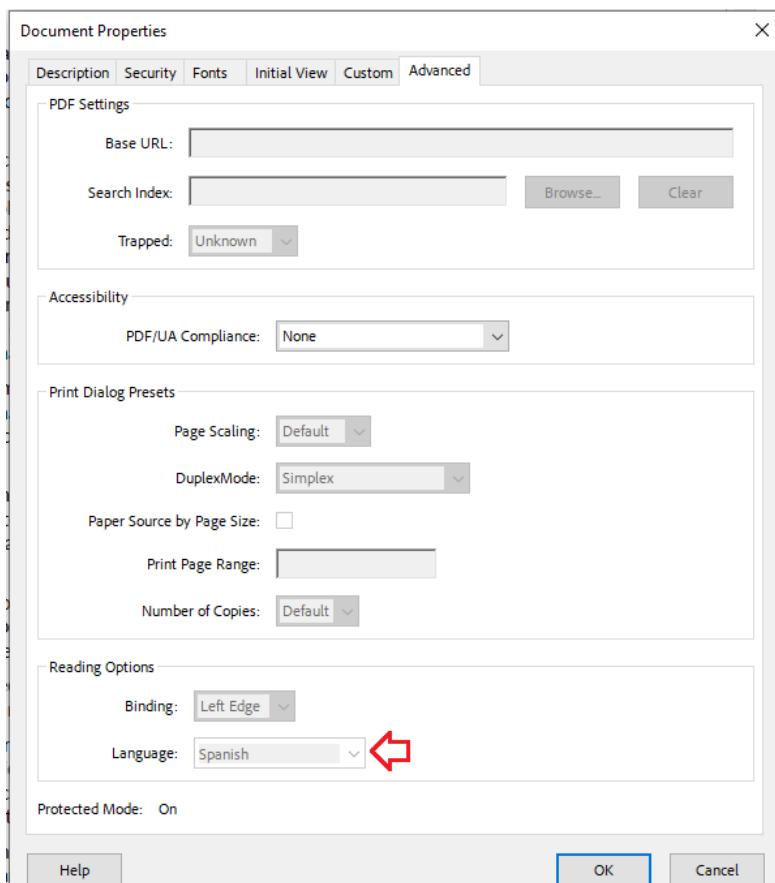
Document's language

Indicate the document's language and any language changes in the content so that the screen reader can read the document with the proper phonetics.

You can indicate the language of the document in the source program:

- in Microsoft Word from the "Review > Language" menu;
- in OpenOffice Writer from the "Tools > Language" menu;
- in Google Docs from the "File > Language" menu.

In Acrobat Professional, the language of the document is set in the "File > Properties > Advanced > Language." If the document's language does not appear in the drop-down menu, as is the case with Catalan or Basque, you can enter its ISO code, in this case, "ca" or "eu," respectively.



Screenshot 12 Language selection in the "Document Properties" menu in Adobe Acrobat Professional

If a piece of content is in a language other than the primary language, it can be indicated in its tag properties from the tree in the "Tags" panel. The language of a piece of content can also be defined from the "Contents" panel.



If a piece of content has a different language defined in the "Contents" panel, it will prevail over the language defined in its tag properties (in the "Tags" panel) and over the language defined in the document properties.

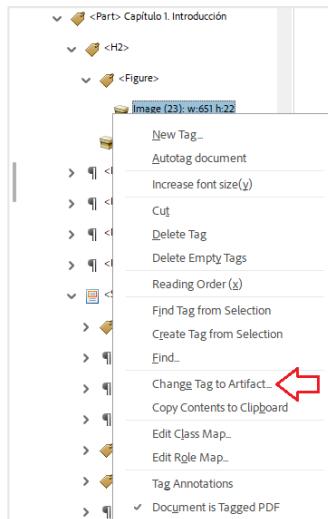
- In Word, you can change the language of a piece of content using the language tool available from the program's status bar.
- In InDesign, you can assign a language to boxes or styles.

Decorative images

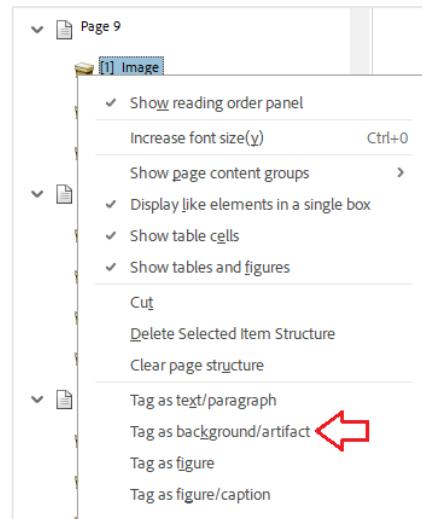
Decorative images do not provide any information or serve any function and, therefore, do not need to be announced by the assistive technology. These images are labeled as "**artifacts**".

In Acrobat Professional, we can turn an image or any other content into an "artifact", that is, remove it from the reading order so that it is not announced, in two ways:

- by selecting its tag and clicking on the "Change Tag to Artifact" option, or
- using the "Accessibility >Fix Reading Order" tool, selecting the decorative content within the document, and clicking "Background".



Screenshot 13 Convert a tag to "artifact" in the Tab panel



Screenshot 14 Convert a tag to "artifact" or background in the Order panel

The **header and footer of the document** must always be included with the specific tool for this purpose. This will allow them to be exported as page artifacts, preventing the screen reader from announcing them on all pages. In InDesign, include these elements on the master page.

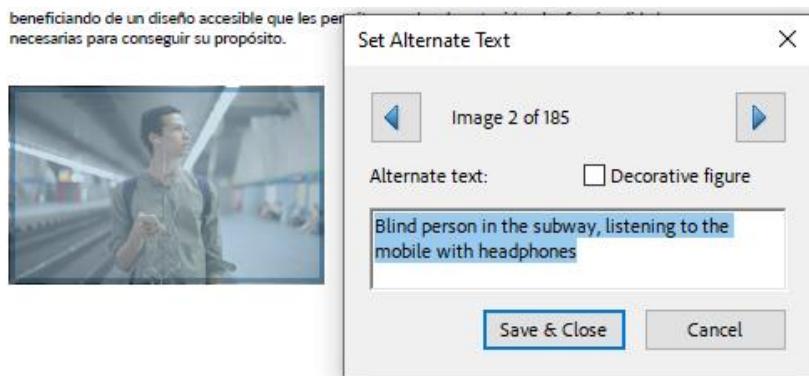


Never remove content from the tag tree.
We can only delete empty tags from it.

Informative images

Informational images are those that have a function or communicate relevant information. In these cases, we must associate the image with alternative **text that conveys the same information or function** to the users of assistive technology.

In Acrobat Professional, *alt* text for images is included in the tag properties in the "Alternate Text" field. Acrobat also has an "Accessibility > Set Alternate Text" tool that allows you to scroll through all the images in your document to review and modify their alt text or even indicate whether they are decorative.



Screenshot 10 Adobe Acrobat Professional "Accessibility > Set Alternate Text" Tool

Most source programs allow you to include alternative texts alongside the images, which will be exported with the images to the PDF.

In Microsoft Office, *alt* text for images is included in the "Alt Text > Description" field in the "Format Picture" window. However, in OpenOffice it is added in the "Alternative" field of the "Picture > Options" menu of the image.

In InDesign, it's defined in the menu "Object > Object Export Options > Alt Text > Custom."

Consistent pagination

The document's pagination, that is, the page number that appears in the header or footer, must match the actual page number. The goal is to make navigation through the Adobe pagination bar accurate, understandable, and consistent with the document's visual pagination and any table of contents.



Screenshot 11 Acrobat Pagination Bar

If the two pagination numbers do not match, you can change the page numbers in the header or footer from the source program, or modify the pagination in Acrobat using the "Page Thumbnails" panel. To do this, select the page or page range you want to change and go to the "Number Pages" or "Page Labels" option, depending on the version of Acrobat.

Destination of links

We recommend that you include the hyperlinks in the source document so that they can be exported to the PDF. The links texts must be significant out of context, and if they are not, we must clarify their destination with an alternative text.

In Acrobat Professional, the "Alternative Text" field in the tag properties allows you to include *alt* text in a link.

Note that this *alt* text **will replace the link text**, that is, the assistive technology announces the link *alt* text instead of its text. For this reason, you must ensure that the *alt* text has all the necessary information and that it makes sense when read alongside the text that precedes and follows it.

In the latest versions of Acrobat, link properties also have an "Alternate Description for Links" field, which is the equivalent to the *title* attribute of links in HTML. This description is visually displayed as a *tooltip*, but screen readers, such as NVDA, do not yet read it in any access context.

Tables

If their structure is complex, data tables can be difficult to understand and access with assistive technology. Therefore, we must always first consider whether the table can be simplified, divided into several simpler ones, or presented in another format.

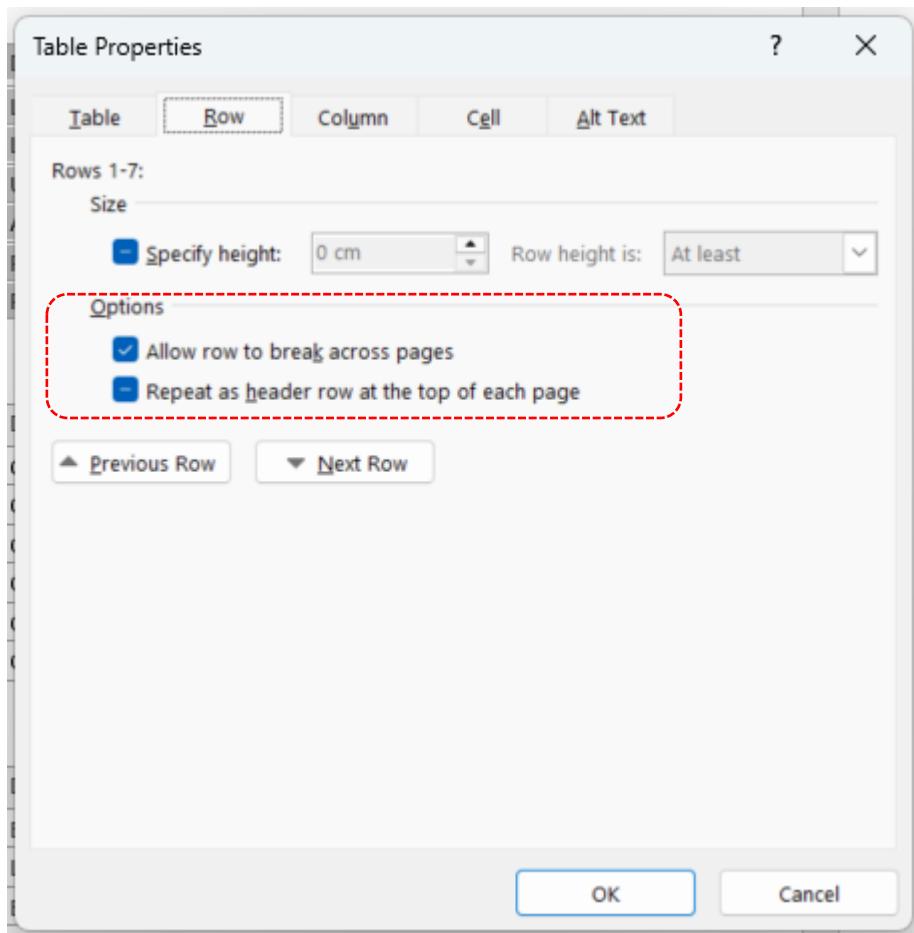
It is highly recommended **not to merge cells**, as this will make the table easier to understand and navigate with a screen reader. In the following table, there are cells with identical data; however, they are not merged but left as separate cells so that everyone can understand them more easily.

Table 11 Table with cells that have the same data but have not been joined together to improve their understanding

Task	Number of hours	Price (Euros)
Task 1	8	400
Task 2	2	100
Task 3	2	100

If the table is split across multiple pages, it's important to repeat the header cells on each page and avoid splitting the contents of a cell between pages. In Microsoft Word, this is defined as follows:

- Select the header row and check the "Table Properties > Options > Repeat as header row at the top on each page" option.
- Turn off the "Table Properties > Row > Allow row to break across pages" option.



Screenshot 15 "Table Properties" in Word

Properly label a table

The table must be correctly labeled; that is, it must indicate which cells are header cells (labeled TH) and which cells are data cells (labeled TD) so the screen reader will announce the header cells before reading each data cell.

We can review and modify the table's structure from the tag tree or from the "Table Editor," which is accessed from the table's context menu.

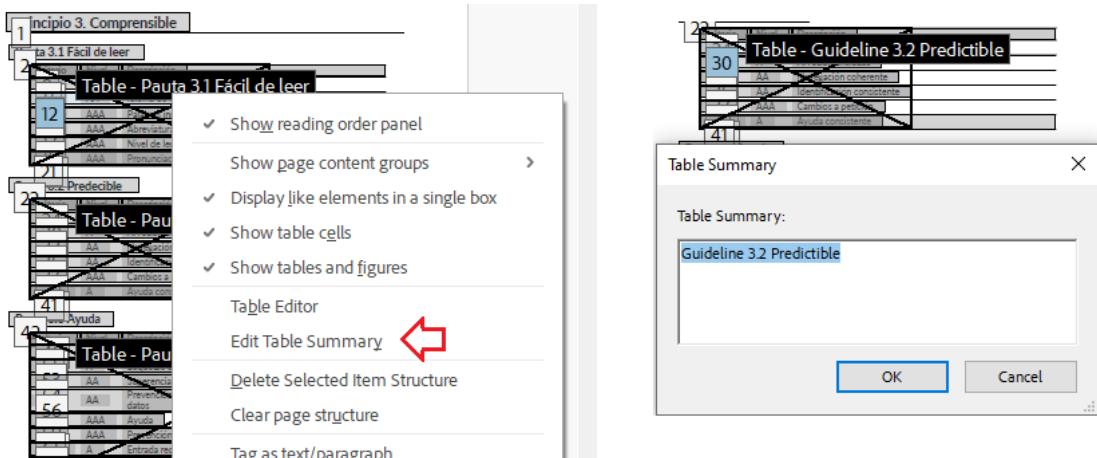
The **table title** must be labeled *Caption*. The PDF standard, which we must respect, indicates that the *Caption* tag must be the first tag nested within the *Table* tag. This will allow the screen reader to read the table's title when it picks up focus. It is highly recommended that tables have a title that is visually placed before the table. The title should be concise and identify the table.

Good **practices in the source program** for tables to be generated with correct labeling are:

- Insert the tables with the "Table" tool.
- Indicate the header row.
- In Microsoft Word, these options are defined as "Table Design > Header Row" and "Table Design > First Column." However, you can't identify multiple levels of header cells; this should be done from Acrobat or by simplifying the table.
- In InDesign, when you create a table, you can also indicate that it has a row with header cells.
- Insert the table title with the corresponding tool, for example, in Word using the "Insert Caption" option in the table's context menu.

The "Description" field in the "Table Properties > Alt Text" in Microsoft Word allows you to associate a description with tables; however, the description is not exported to the PDF.

In Acrobat Professional, the table description is added by selecting the table using the "Accessibility > Fix Reading Order" tool, then right-clicking the table to open the context menu and selecting the "Edit Table Summary" option.



Screenshot 16 "Edit Table Summary" in Adobe Acrobat Professional

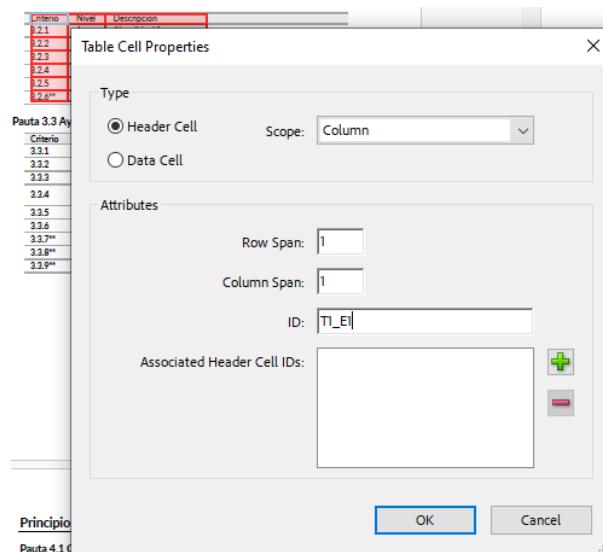
Indicate the relationship between header cells and data cells

If the table is complex, with more than one level of headers, each data cell must be associated with all the cells that head it. This can only be done in Acrobat Professional and is quite laborious, so it is always advisable to try to simplify the table first.

Defining the relationship between data and header cells is done using the "Table Editor", accessed from the table's context menu.

Using the "Table Editor > Table Cell Properties" option, we must:

- Assign each header cell a unique ID in the document.
- Specify in each data cell the IDs of all the cells that head it.



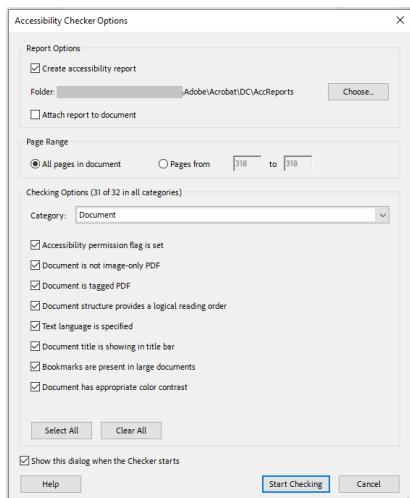
Screenshot 17. Adobe Acrobat Professional "Table Editor"

Validate PDF accessibility

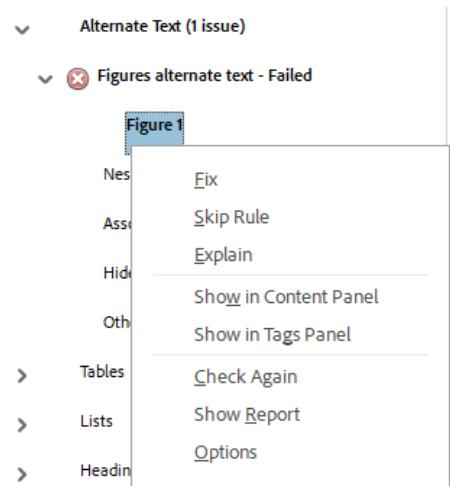
Nothing can replace manually assessing document accessibility and listening to it with the screen reader. However, as an aid, automatic accessibility validators can be particularly useful.

From the 2010 version of **Microsoft Office**, you can review your document using the accessibility validator, located under "File > Information > Check for Issues > Check Accessibility". This validator detects errors such as images without *alt* text, unnecessary carriage returns, or tables where the header row is not specified. It is highly recommended that you check whether this validator detects problems in your document and resolves them before exporting it to PDF.

Adobe Acrobat Professional features an accessibility validator in "Accessibility > Check for accessibility", which is likely the most reliable on the market. While you can't evaluate all the accessibility requirements that the PDF must meet, the errors you find always have a reason for being and need to be corrected.



Screenshot 18 Adobe Acrobat Professional Accessibility Checker Options



Screenshot 19 Adobe Acrobat Professional Accessibility Checker Results

Adobe Acrobat Professional also includes the "Make Accessible" tool within the "Use setup assistant". It does not make the PDF accessible, but as an assistant, it guides us through the actions we must perform.

Although we always recommend validating with the Adobe Acrobat Professional tool, there are many automatic accessibility validators, such as [PAC \(PDF Accessibility Checker\)](https://pac.pdf-accessibility.org/en/check)⁷², which is free.

⁷² <https://pac.pdf-accessibility.org/en/check>

PAC 2024 - PDF Accessibility Checker 2024

Sobre PAC 2024

Abrir documento

Título: **Accesibilidad Web. WCAG 2.2 de forma sencilla**

Nombre del archivo: **Revilla_30_12_2023_2_Accesibilidad_Web_2_2.pdf**

Idioma: **en-US**

Páginas: **386**

Etiquetas: **11339**

Tamaño: **6 MB**

PDF/UA WCAG Calidad

Este archivo PDF no es compatible con PDF/UA.

Control	Aprobado	Prevenido	Ha fallado
⚠ Sintaxis PDF (ISO 32000-1)	12120	2	0
✗ Fuentes	26	0	5
✓ Contenido	497151	0	0
✗ Archivos incorporados	0	0	0
✓ Idioma natural	239389	0	0
✗ Elementos de estructura	1535	51	659
✗ Árbol de estructura	10804	528	7
✓ Mapeo de roles	11387	0	0
✗ Descripciones alternativas	22501	0	582
✗ Metadatos	2	0	1
✓ Configuración de documentos	104	0	0

Resultados en detalle Informe PDF

Screenshot 20 Results of a PAC assessment



You can consult a list of PDF validators in "[Validators and tools for accessibility and usability consultancies. Validate PDF](#)"⁷³ by Olga Carreras.

⁷³ <https://www.usableyaccesible.com/recursomisvalidadores.php#accesibilidadpdf>

Export to PDF

Once the accessibility of the source document has been validated, it is time to save or export it as a PDF. Depending on the program and version, the export options that should be selected are:

- Export as **tagged PDF** so that it is generated with the tag tree that, as we have seen, is the basis of PDF accessibility.
- Export **Bookmarks** to automatically create the bookmark index from the headers in the document.
- Add **Links** to preserve the links in the document.

In the export options, it is common to select the **PDF version** that you want it to be compatible with. A version later than Acrobat 6.0 / PDF 1.5 must be chosen, as content tagging options were introduced starting with that version. The most common choice is to select compatibility with Acrobat 8 or higher / PDF 1.7. Don't choose the latest version as selecting PDF 2.0 will make the PDF inaccessible in Adobe Acrobat XI or earlier; it will only open in the DC version.

Remember that a **document scanned as an image can never be an accessible document**: it cannot be searched or selected; its structure cannot be interpreted by assistive technologies, nor its content; when zoomed in, the text will appear *pixelated*, etc. That's why it's imperative to convert it to text using an optical character recognition (OCR) tool. Adobe Acrobat Professional itself includes a tool for this, called "Text Recognition" or "Digitize and OCR," depending on the Adobe version.

PDF/A and PDF/UA

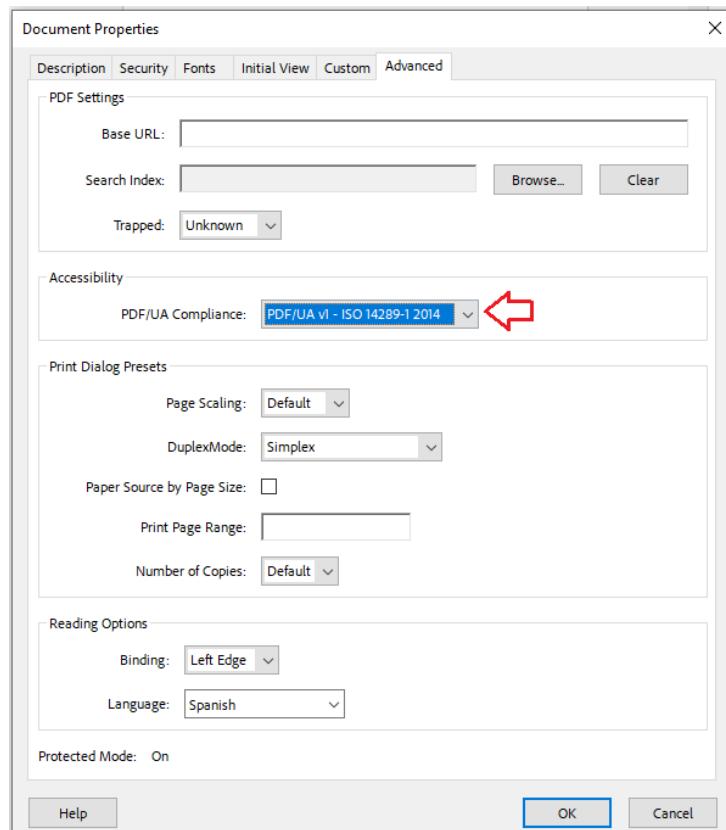
Although the PDF standard was developed by Adobe, since 2008 it has been a formal, open standard published by ISO as ISO 32000⁷⁴. There are different PDF sub-standards: PDF/A, PDF/X, PDF/E, or PDF/UA, which define a PDF that must meet certain requirements.

It's a common misconception that a **PDF/A** is an accessible PDF. The option to save or export a PDF as a PDF/A is a common option in many programs. A PDF/A is not an accessible PDF, it is simply a PDF that is intended to remain the same in the long term, that is, that it can be reproduced accurately in the future. For this reason, it has to meet certain requirements, such as being labeled or self-contained, that is, that the sources, images, etc. are embedded in the document.

On the other hand, the **PDF/UA standard** (PDF/Universal Accessibility) defines the characteristics a PDF must meet to be accessible.

⁷⁴ ISO 32000 <https://www.iso.org/standard/75839.html>

The latest versions of Acrobat Professional allow the author of the document to indicate in its properties whether it is a **PDF compliant with the PDF/UA standard (ISO 14289-1:2014 Document management applications – Electronic document file format enhancement for accessibility – Part 1: Use of ISO 32000-1 (PDF/UA-1))**⁷⁵.



Screenshot 14 "Document Properties" in Adobe Acrobat Professional

Neither Adobe's accessibility validator nor its standards validation tool, which includes validation according to the PDF/UA standard, can determine whether the PDF is accessible according to the PDF/UA standard; they only perform some syntax checks. Therefore, only a manual assessment can determine whether the PDF is accessible and compliant with the PDF/UA standard.

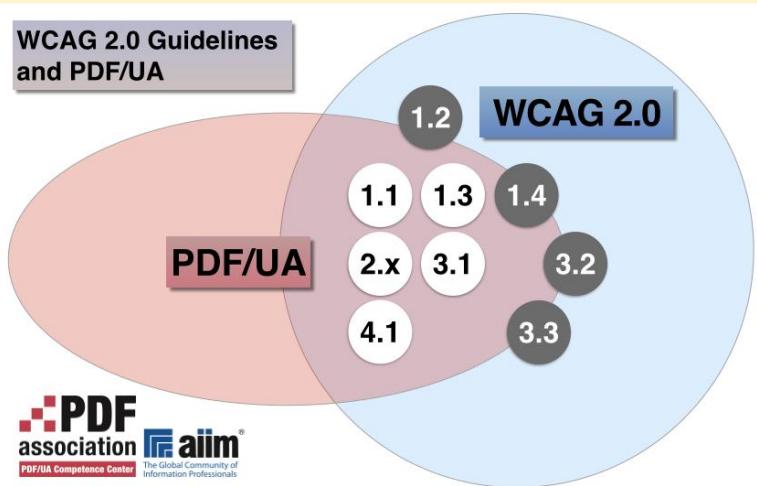
The PDF/UA standard includes many requirements common to WCAG 2.2 and some specific ones. Additional requirements for the PDF/UA standard are:

1. All fonts must be embedded in the PDF.
2. All characters must correspond to Unicode characters, which ensures that the screen reader can interpret and read all text correctly.
3. Document security settings (defined in "File > Properties > Security") should not prevent assistive products from accessing the content.

⁷⁵ <https://www.iso.org/standard/64599.html>



You can find more information about the "[PDF/UA standard and its relationship with WCAG 2](#)"⁷⁶ by the PDF Association.



Tutorials for accessible documents

- [Official documentation of accessible PDF techniques](#)⁷⁷.
- [EPUB](#)⁷⁸
- [Word](#)⁷⁹
- [PowerPoint](#)⁸⁰
- [PDF](#)⁸¹

⁷⁶ <https://pdfa.org/infographics-pdfua-and-wcag-2-0/>

⁷⁷ <https://www.w3.org/WAI/WCAG22/Techniques/#pdf>

⁷⁸ <https://www.accessibletextbooksforall.org/stories/creating-accessible-epub>

⁷⁹

https://www.csulb.edu/sites/default/files/2023/documents/AIM%20Center_Microsoft%20Word%20Accessibility%20Guide%202023.pdf

⁸⁰

https://www.csulb.edu/sites/default/files/2023/documents/AIM%20Center_Microsoft%20PowerPoint%20Accessibility%20Guide%202023.pdf

⁸¹ <https://helpx.adobe.com/acrobat/using/create-verify-pdf-accessibility.html>

ARIA, the ally of accessible HTML

Websites often have non-native controls, usually developed with JavaScript, that screen readers cannot interpret correctly.

This results in users not understanding them or being unable to use those controls to interact with the website.

To solve this problem, the W3C developed ARIA (Accessible Rich Internet Applications) standard, which allows semantic information to be added to any interface element.

Through the accessibility API, the browser transmits this semantic information to the assistive product, which then conveys it to the user.

This way, people who use screen readers or other assistive products can interact with the website normally.

What is ARIA?

In an ideal world, HTML elements are used and function as intended: a button is a button, a link is a link, and a list is just that, a list of elements. However, HTML doesn't have a tag to define a drop-down tree, tabs, or alert messages. Therefore, we have no choice but to create these new components based on standard HTML elements plus JavaScript programming (or AJAX or any JavaScript framework).

Sometimes, for design reasons, we use an HTML element for a function other than the one it was defined for, even though there is an HTML element that we could use instead. For example, we can make a *div* behave like a *form* element (e.g., a checkbox, a drop-down, or a button) by modifying its appearance using CSS styles and adding JavaScript events to it.

People who can see the screen will understand, from the visual representation of the item, that the *div* is a button, or they will see that a list is a drop-down tree that is open or closed. But the user of an assistive product, such as a screen reader or a braille display, does not have it so easy. To understand why, we must first briefly explain **how an assistive product works**.

The browser exposes information about the interface to the operating system's accessibility API. In turn, the screen reader extracts this information from the accessibility API and announces it to the user. If what we have is a list of items (*ul*), even if it behaves as a drop-down tree, the browser will expose it to the API as a list. In turn, the assistive product will take this information and announce to the user that it is a list of items.

The screen reader will announce that it is a list of three items in the example below.

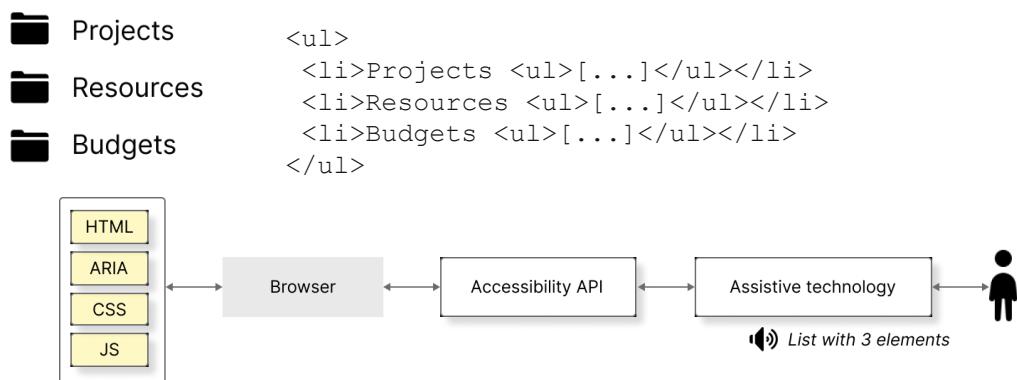


Illustration 34 Communication scheme between the browser and the supporting product via the Accessibility API. Example of an item list.

How do we make the assistive product announce to the user that the list is a drop-down tree? To do this, WAI-ARIA, or ARIA (Accessible Rich Internet Applications), was created, a taxonomy of roles, states, and properties that helps define the elements of the interface.

In the case of the drop-down tree, thanks to ARIA, we can define the new function of the list and add different properties and states, for example, the name with which we want the tree to be announced, to indicate whether it is open or closed, etc.

- 📁 Projects
 - 📁 Resources
 - 📁 Budgets
- ```
<ul role="tree" aria-label="File list">
<li role="treeitem" aria-expanded="false"
aria-posinset="1" aria-setsize="3"
aria-level="1" tabindex="0">
Projects [...]
```

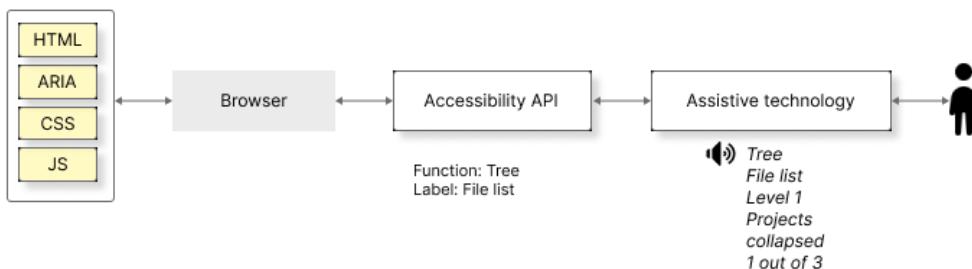


Illustration 35. Communication scheme between the browser and the supporting product via the Accessibility API. Example of a tree defined with ARIA.

In this way, the user of the assistive product will have all the information to be able to understand and manage the tree. In this case, a screen reader such as NVDA will announce "File List Tree Level 1 Projects collapsed, 1 of 3".

Therefore, ARIA can be defined as a **set of attributes that are added to HTML tags** so that user agents (browsers and assistive products) understand that these tags behave differently from the default behavior.

ARIA was born in 2008. The first recommendation, WAI-ARIA 1.0, was published in 2014. The latest recommendation is WAI-ARIA 1.2, released in 2023, and virtually all [browsers and supporting products support ARIA](#).<sup>82</sup>

<sup>82</sup> <https://caniuse.com/#feat=wai-aria>

It is becoming a better-known standard slowly, but it is often poorly implemented. This is because, as we have seen, knowing the problems of people who access websites with assistive products is not sufficient to really understand its importance and apply it correctly.

WCAG 2.2 proposes using ARIA as an optimal way to meet some of its compliance criteria since, when used well, it allows overly complex designs and functionalities to be fully accessible.

With ARIA, you can create complex accessible controls, such as progress bars, sliders, drop-downs, floating tooltips, alerts, popover with dialog options, sorting of lists of elements, collapsible and drop-down content trees, drag-and-drop elements, carousels, tabs, accordions, toolbars and menus, dynamic grids, etc.

With ARIA, you can define content areas that are updated without user intervention and configure how the screen reader should announce them. For example, we can mark a social network timeline, a clock that indicates the time we have left to fill out a form or a notification so that the updates do not go unnoticed by the user who cannot see them.

But you can also use ARIA for simpler things, such as marking the function of a region of the page, naming the interface components or attaching an additional description to them, indicating the mandatory nature of a form field, or indicating that it is invalid because it has given a validation error.

ARIA is not a language but a complementary to HTML5 and SVG since, although we do not discuss it in the book, it also allows SVG content to be made accessible.

## Example 1. Change semantics and operation: a layer that behaves like a button

Imagine that instead of putting a "Submit" button on a form, we include an image that performs the same function with the following code:

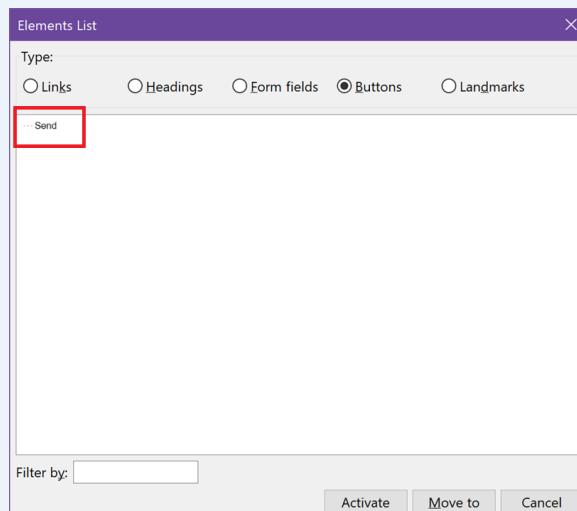


```
<div></div>
```

Instead of using the native `<button>` tag we're going to use the `<div>` tag and we need to make it behave like a `<button>` would. To inform the browser of the new functionality of the `<div>` and to transfer the information to the screen reader, through the accessibility API, we must add the `role="button"` attribute to the tag.

```
<div role="button"></div>
```

From this point on, for the screen reader it is a button: it announces it as such ("Submit graphic button"); it appears in the list of buttons on the screen reader, and we can reach it by pressing the screen reader's keyboard shortcut for buttons (usually the letter "b").



Screenshot 21 NVDA Button Listing (insert+F7)

However, simply stating that it's a button does not make it behave like one. We'll need to add its behavior, including the corresponding events, with non-intrusive JavaScript.

Even if we associate the ***onclick*** and ***onkeypress*** events, the `<div>` will not take the focus when we navigate with the keyboard tab or a button, as a `<button>` would. Therefore, it will not be accessible by keyboard, only by mouse.

To get it to take the keyboard focus, we will have **to add the `tabindex="0"` attribute**, which tells the browser to include the `<div>` in the list of elements with which the user can interact. In this way, we can use the TAB key to reach the `<div>` element (in the order that corresponds to it in the DOM), and the SPACE or ENTER key to press it.

The improved final code would look like this (the *onclick* and *onkeypress* events are assumed to be added using non-intrusive JavaScript):

```
<div role="button" tabindex="0" id="buttonSend">
</div>
```

The case explained is an example of something that can be done, but **should it be done?** If we can use the native tag, it is best to do so. ARIA is an add-on that improves page accessibility, not a substitute for native controls.

The four cases in which **it is recommended to use ARIA instead of native HTML elements** are:

1. The feature is not available in HTML.
2. The feature is available in HTML but not implemented in user agents.
3. The feature is available and implemented in HTML, but the user agent does not provide support for the accessibility of that element.
4. Visual design 'forces' a certain style, but we couldn't design a native element with that visual aspect.

# Roles

Roles in ARIA allow you to indicate the function of an interface element. The role is assigned with the role attribute. **ARIA 1.2 defines 94 roles**, twelve more than the ARIA 1.1 version, categorized as follows:

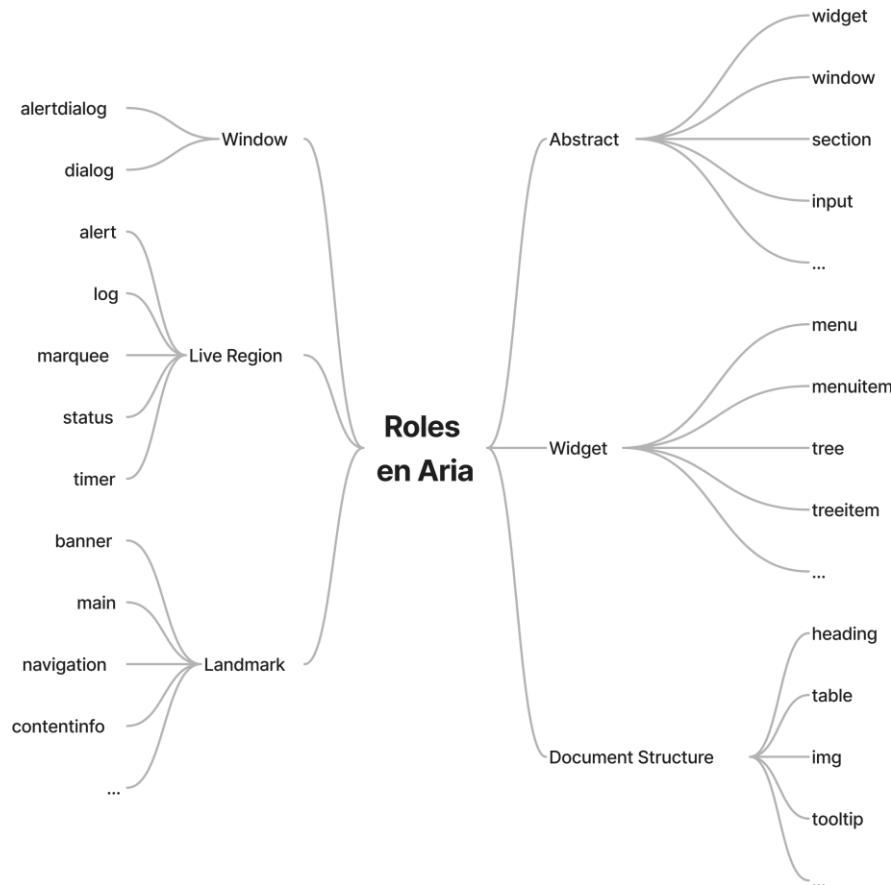


Illustration 36 Role Outline in ARIA

- **Abstract:** *widget, window, section, input, etc.* They are used only to define general role types, are not applied in components, and therefore should never be found on an HTML page.
- **Widget:** interactive elements with more complex functionalities than native HTML elements: *menu, menuitem, tree, treeitem, tab, tablist, etc.*
- **Document Structure:** *heading, table, img, tooltip, etc.* They are not usually interactive.
- **Landmark:** *banner, main, navigation, contentinfo, etc.* They allow you to define the large regions of the page, such as HTML5 tags (*header, main, nav, footer, etc.*). Users of a screen reader have shortcuts to jump from region to region and can pull up a tree from the page structure generated from this information.

- **Live Region:** *alert, log, marquee, status, and timer.* They define the function of the "live" areas of the page, that is, those that are changed automatically, without the intervention of the user, and without taking the focus.
- **Window:** *alertdialog and dialog,* for layers that open as windows and receive focus, waiting for some action from the user.

You can learn their details in the "[Definition of Roles](#)"<sup>83</sup> of the WAI-ARIA 1.2 specification or refer to the [Schematic graph of relationships between roles](#).<sup>84</sup>

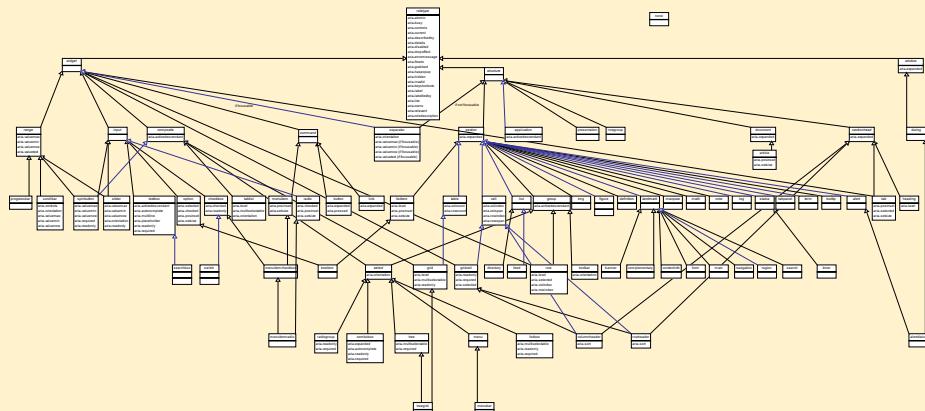


Illustration 37 WAI-ARIA 1.2 Schematic Graph of Roles Relationships

<sup>83</sup> [https://www.w3.org/TR/wai-aria-1.2/#role\\_definitions](https://www.w3.org/TR/wai-aria-1.2/#role_definitions)

<sup>84</sup> [https://www.w3.org/TR/2023/REC-wai-aria-1.2-20230606/img/rdf\\_model.svg](https://www.w3.org/TR/2023/REC-wai-aria-1.2-20230606/img/rdf_model.svg)

# States and properties

In addition to roles, ARIA defines the states and properties of the various controls. The conceptual difference between "state" and "property" is subtle: properties tend to change less (though not always) than states, which change frequently due to user interaction. In practice, it is unnecessary to differentiate between a property and a state, as all of them will begin with *aria-*.

There are **46 states and properties** that fall into four categories:

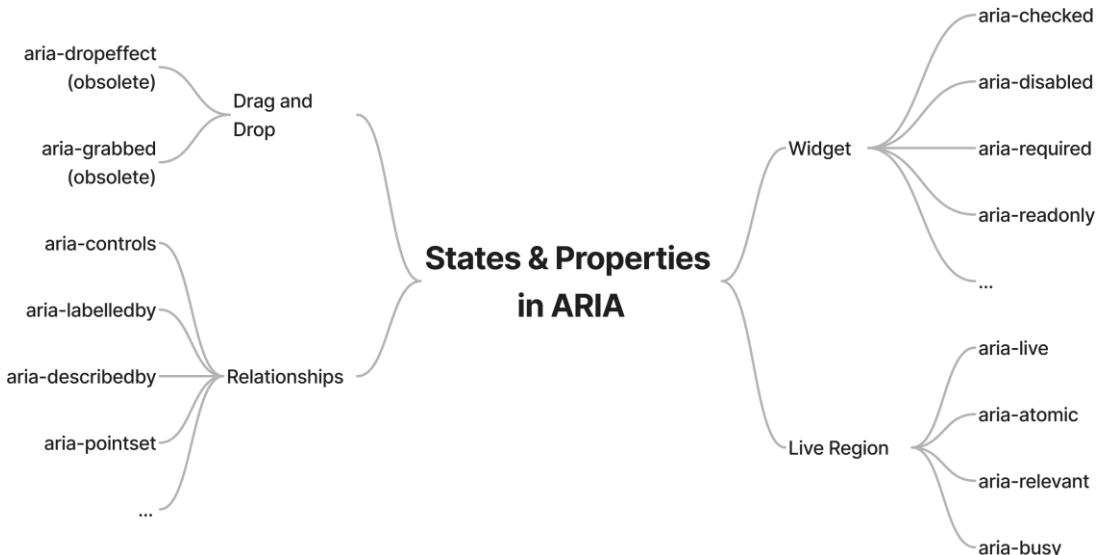


Illustration 38 Schema of states and properties in ARIA

- **Widget attributes:** *aria-checked*, *aria-disabled*, *aria-required*, *aria-selected*, *aria-readonly*, *aria-expanded*, *aria-pressed*, *aria-label*, etc., to express attributes of common components that usually receive input from the user or process their actions (checked, disabled, mandatory, read-only, expanded, pressed, their name, etc.).
- **Live Region attributes:** *aria-live*, *aria-atomic*, *aria-relevant*, and *aria-busy*, which allow you to define when changes to areas that update themselves without being targeted will be announced to the user of the assistive technology; which part will be announced; what type of update we want to be announced; or if we want them to temporarily stop being announced. For more information and examples, see the [article "Live Regions and WAI-ARIA"](#).<sup>85</sup>
- **Relationship Attributes:** which express relationships or associations between elements that cannot be easily determined from the structure of the document: *aria-controls*, *aria-labelledby*, *aria-describedby*, *aria-posinset*, *aria-setsize*, etc.

<sup>85</sup> <https://olgacarreras.blogspot.com.es/2013/11/live-regions-y-wai-aria-como-mejorar-la.html>

- **Drag-and-Drop attributes:** i.e., components that are dragged and dropped across the screen. There are only two attributes (*aria-dropeffect* and *aria-grabbed*) and they are currently considered obsolete, awaiting replacement by others in the future.

States and properties can be changed with JavaScript; in fact, it is particularly important to change them when a user interacts with them.

If the user opens or closes the drop-down tree thanks to the associated JavaScript events, the functions must also include the dynamic change from the *aria-expanded="false"* attribute to *aria-expanded="true"* so that the screen reader correctly announces the state change to the user. Conversely, the value of the *role* attribute, which indicates the component's function, must remain fixed.

There are **three fundamental properties** referenced by many of the ARIA techniques in WCAG 2.2 that we must use continuously and do well. They are those that allow you to tag or associate descriptions with the elements:

- *aria-label*,
- *aria-labelledby*
- *aria-describedby*.

We will see them in depth in the following pages.



You can access the definition and detail of each state or property in the "[Definitions of States and Properties \(all aria-\\* attributes\)](#)"<sup>86</sup> of the WAI-ARIA 1.2 specification.

<sup>86</sup> [https://www.w3.org/TR/wai-aria-1.2/#state\\_prop\\_def](https://www.w3.org/TR/wai-aria-1.2/#state_prop_def)

---

## aria-label

With this attribute, we directly indicate **the text of the item's label**:

```
<button aria-label="Close">X</button>
```

It is particularly important to know that *aria-label* overrides the native label. In other words, the screen reader will not read "X button" or "X Close button" but "Close button" which, in this case, is what we want.

We can also use it to **expand the information** of a link that could be confusing:

```
<a href="..." aria-label="Read more about web
accessibility">Read more
```

However, we **should not** use this attribute in a link as a *title*:

<b>Incorrect</b>
<pre>&lt;a href="" aria-label="Opens in new window" target="_blank"&gt;Read more about web accessibility&lt;/a&gt;</pre>

Another common use is to **distinguish regions of the page** of the same type, for example, two navigation zones:

```
<div role="navigation" aria-label="Secondary menu">
```

The three examples given are some cases for which WCAG 2.2 supports using *aria-label* as a sufficient technique.

It is common to find inappropriate texts or in the wrong language within the *aria-label* attribute. It is necessary to check that it is used correctly and that the text it includes is appropriate and is in the correct language.

---

## aria-labelledby

Both *aria-label* and *aria-labelledby* are used to **label an element**, that is, to give it an accessible name that is announced by the assistive technologies. The difference is that with *aria-label* we directly include the text that we want to function as an accessible name, while with *aria-labelledby* we reference the ID of the element (or elements) that function as the label of the component.

The *aria-label* and *aria-labelledby* attributes cancel each other out and it doesn't make sense to use them together. In addition, like *aria-label*, *aria-labelledby* also overrides the native label.

In the following example, the screen reader announces the link as "Annual Report 2017 Download PDF (25 KB)" instead of "PDF link (25 KB)", meaning it concatenates the label of the two referenced elements in the *aria-labelledby* attribute:

```
<h2 id="report">Annual Report 2017</h2>
<p><a aria-labelledby="pdf report" href="" id="pdf">
 Download PDF (25 KB)
</p>
```

As we can see, the *aria-labelledby* attribute can refer to one ID or to several, in which case, they are separated by spaces.

You must always refer to the ID of the element itself first, to ensure that the link is accessible via voice access (criterion 2.5.3 A).

Another example of common use is in **search forms**. In the following case, the screen reader announces the field as "Search", taking advantage of a label already available on the page, the adjacent button.



```
<input name="searchtxt" type="text" aria-labelledby="btn">
<input name="searchbtn" id="btn" type="submit" value="Search">
```

We can also use *aria-labelledby* to leverage the **value of a form field**. In this example, the screen reader announces the field as "Extend time up to 20 minutes" (instead of "Extend time to 20").

Extend time up to 20 minutes

```
<label for="duration" id="timeout">Extend the time to </label>
<input type="text" id="duration" value="20" aria-
labelledby="timeout duration unit">
minutes
```

In addition, we can use *aria-labelledby* to **label areas of the page**. It can also be used to **label images**, as shown in the following example.:



```
<div role="img" aria-labelledby="points">

</div>
<div id="points" aria-hidden="true">
 score
 4 of 5
</div>
```

In this case, the screen reader ignores the stars (because they have *alt=""*) and announces the *div* as an image (thanks to *role="img"*) with the label "score 4 out of 5", associated with the image through the ID referenced in the *aria-labelledby* attribute.

The *aria-hidden="true"* attribute is used so that the assistive product does not announce the contents of the *div* after the image, as this would be unnecessary and redundant. In addition, the text "score" is visually hidden.

This example also allows us to explain that, even if the elements referenced by the *aria-labelledby* attribute are hidden using any technique, this does not affect their behavior, the screen reader will continue to announce "score 4 out of 5".

These are several of the examples that include WCAG 2.2 for the use of *aria-labelledby* as a sufficient technique.

---

## aria-describedbyby

It allows us to reference the ID of the element that we want to function as a description of another, that is, to provide **additional information** beyond its tag.

Let's imagine a dialog window with a button to close it, certain content, and a warning message at the end.



```
<button aria-label="Close" aria-
describedby="descClose">X</button>

[...]

<div id="descClose">
<p>Closing this window will discard any changes made and you
will be redirected to the homepage.</p>
</div>
```

The *aria-describedbyby* attribute of the close button allows us to associate additional information with its label, so that when it receives focus, the screen reader announces that additional description, ensuring it does not go unnoticed by the user: "Close button. Closing this window will discard any changes made and you will be redirected to the homepage."

The *aria-describedbyby* can also be used to associate the **extensive description of an image**:

```


<div id="descWCAG">
<p>The WCAG 2.2 guidelines are made up of 4 principles:</p>

Perceivable
Operable
Understandable
Robust

<div>
```

It should be noted that the screen reader will read the description continuously, without announcing its semantic markup.

Another way to use *aria-describedby* is to **associate a text field with its contextual help:**

Password (required)

Passwords must have at least 6 characters.

```
<label for="against">Password (required)</label>
<input name="contra" id="contra" type="password" aria-
describedby="descripcionContra" />

<p id="descripcionContra" class="help">
 Password must have at least 6 characters.
</p>
```

The screen reader announces the field labeled "Password (required)" and adds the description "Password must have at least 6 characters."<sup>87</sup> This way, the field requirements will not go unnoticed by a screen reader user, who typically navigates through the form by jumping from field to field.

These are several examples that include WCAG 2.2 as sufficient techniques for using *aria-describedby*.

---

<sup>87</sup> The NVDA screen reader advertises exactly that field as: "Password required. Password edit field. The password must be at least 6 characters long. Blank." Depending on the screen reader and browser combination, the ad may be different.

## Example 2. Accessible Tabbed Navigation

Let's say we have content that is visually structured with tabs and laid out using the code below:

The diagram shows a horizontal row of three rectangular buttons labeled "Chapter 1", "Chapter 2", and "Chapter 3". The "Chapter 1" button has a thick blue border, while the others have a thin grey border. Below this row is a rectangular box containing the text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris."

```
<!-- tabs -->

 Chapter 1
 Chapter 2
 Chapter 3

<!-- contents -->
<section id="cap1">
 <p>Lorem ipsum... </p></section>
<section id="cap2"> ... </section>
<section id="cap3"> ...
</section>
```

When a tab is clicked, its appearance changes to visually reflect that it is selected, and the content of the corresponding chapter is displayed. To do this, we use non-intrusive JavaScript and CSS styles.

However, people who sign in with an assistive product, such as a screen reader, won't understand the true function of the link list, what happens when they are clicked, or how they relate to sections.

Let's improve its accessibility with ARIA.

- First, we must indicate that the list (`<ul>`) will behave like a list of tabs (`<ul role="tablist">`) and assign it an accessible name with `aria-label` or `aria-labelledby`. We start from a list `<ul>` and not from nested `<div>` because in this way, they will retain their relationship even when the presentation mode is changed, for example, if CSS is deactivated.

- Then, it should be indicated that each item in the list (`<li>`) serves a presentational role (`<li role="presentation">`), so the screen reader will announce its content, but not its native role. When `role="presentation"` is applied to an image, it's like applying an `alt=""` or an `aria-hidden="true"`, i.e. it's ignored by the screen reader. But when applied to another element, the screen reader announces its textual content but removes the semantic information from its `role`, i.e., without announcing that it is a list, table, or heading.
- Now, let's add several attributes to the links within each list item:
  - The tab role (`<a role="tab">`) that overrides its native binding role.
  - The `aria-controls="cap1"` attribute that relates the tab to the section it controls.
  - The `aria-selected="true"` attribute in the selected tab and `aria-selected="false"` in the others. The value of this attribute will need to be dynamically updated by JavaScript each time the user selects a tab.
  - The `aria-posinset` and `aria-setsize` attributes to indicate where the tab is positioned and how many tabs there are.
  - Following the principle of progressive enhancement, the default link in the tabs (later overridden by JavaScript) will be an anchor to its section (`<a href="#cap1"...>`).
- Finally, we must add several attributes to each content section:
  - `role="tabpanel"` to indicate its function.
  - the `aria-labelledby` attribute to label it.
  - the `tabindex="0"` attribute so that it can take keyboard focus (more on how keyboard access should look in the tab pattern).
  - the `aria-hidden="true"` attribute in the hidden `tabpanel`s and `aria-hidden="false"` in the visible `tabpanel`. This attribute's value must be dynamically updated by JavaScript each time the user selects a tab. This step is not necessary if they are hidden with the `display:none` style, but it is necessary if they are hidden with other styles, such as `text-indent:-1000px`.
  - We will include a header within each section, which can be visually hidden, so that the section's content is clear from other access contexts, such as without CSS.

In this way, the **final improved code** would be as follows:

```

<!-- tabs -->
<ul role="tablist" aria-label="Chapters of Don Quixote">
 <li role="presentation">Chapter 1

 <li role="presentation">Chapter 2

 <li role="presentation">Chapter 3

<!-- contents -->
<section id="cap1" role="tabpanel" aria-hidden="false" tabindex="0" aria-labelledby="cap1"><h2>Chapter 1</h2>
<p>Lorem ipsum...</p></section>
<section id="cap2" role="tabpanel" aria-hidden="true" tabindex="0" aria-labelledby="cap2">
<h2>Chapter 2</h2>...
<section id="cap3" role="tabpanel" aria-hidden="true" tabindex="0" aria-labelledby="cap3"><h2>Chapter 3</h2>
...
</section>
```

Why are `tabindex="0"` and `tabindex="-1"` added to tab links if they are elements that take focus by default? The reason is related to how tabs should behave with keyboard navigation. If the same type of component, such as tabs, is handled differently on each website, users who rely on keyboard navigation will have to learn how to use it on each page. For this reason, **the ARIA standard also defines how keyboard access should be** in tab, menu, tree, and other components.

In the case of tabs, once the first tab takes focus, navigation to the other tabs should be done using the right and left arrow keys (not with the Tab key); the Home and End keys should allow you to move to the first and last tabs; pressing the Enter or Space key should activate the tab; and the Tab key should allow you to access the content of the tab.

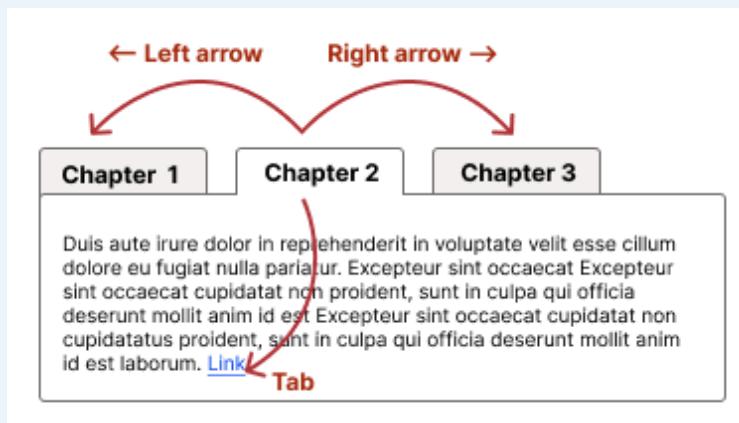


Illustration 39 Keyboard navigation within the tabs.

For this reason, the first tab is given the `tabindex="0"` attribute, so that it takes focus while the others are removed from the tab sequence with `tabindex="-1"` (an attribute that indicates the element can only take focus through scripting). As the user selects the tabs with the arrow keys on the keyboard, the attribute is dynamically changed by JavaScript from `tabindex="-1"` to `tabindex="0"`. This is known as the "roving tabindex". In other components, as we will see later, focus is managed using the `aria-activedescendant` attribute.

With the improved code, when the first tab of the example takes focus, the NVDA screen reader announces: "Chapters of Don Quixote, tabs. Chapter 1 selected tab 1 of 3." Depending on the screen reader and browser combination, the announcement may vary.

The screen reader user, therefore, will be able to navigate from one tab to another using the keyboard, having perfect knowledge of where they are and what is happening on the page.

We recommend the article "[Developing a Keyboard Interface](#)"<sup>88</sup> to learn more about how the browser interacts with the keyboard.



The articles "[Tabs Pattern](#)" by ARIA Authoring Practices Guide<sup>89</sup> by the W3C and the article "[Tabbed interfaces](#)"<sup>90</sup> by Heydon Pickering explain the tab pattern in detail.

---

<sup>88</sup> <https://www.w3.org/WAI/ARIA/apg/practices/keyboard-interface/>

<sup>89</sup> <https://www.w3.org/WAI/ARIA/apg/patterns/tabs/>

<sup>90</sup> <https://inclusive-components.design/tabbed-interfaces/>

# Best practices for ARIA

**Misused, ARIA is a danger.** ARIA has the power to provide semantics to the interface, to describe almost any component in a way that can be interpreted by assistive products... but annulling the original semantics, which, if done inadvertently or through incorrect application of the standard, can confuse rather than help.

To mitigate this problem, the W3C recommends:

1. **Do not use ARIA if it is not necessary.** Use HTML tags in the standard way whenever you can. If you can use `<input type="checkbox">` or `<button>` use them instead of `<div role="checkbox">` or `<div role="button">`. Remember that the ARIA role overrides the native role.
2. **A role is a promise.** If you indicate that a `<div>` is a button (`role="button"`) as we have seen in the first example of this chapter, this does not make browsers provide that element with the style or behavior of a button, we only get it to be advertised as a button, but it will be your responsibility to behave as such.
3. **Use roles and properties as per specification.** You must mark the structure of the widget (*menubar*, *tablist*...) and the relationships between its elements (*aria-labelledby*, *aria-controls*...). Remember that the role should not change dynamically, only the properties and states are changed.
4. **Avoid conflicts.** Don't add ARIA to tags if they may conflict with your own semantics. For example, if we redefine the behavior of a *radiobutton* as if it were a *checkbox* (`<input type="radio" role="checkbox" />`), each user agent could implement it in a different way, the behavior would be chaotic, and the robustness of the code would be lost.
5. **Avoid redundancy.** Don't add ARIA to native controls with the same value, as it's redundant. For example, these examples would be absurd:  
`<input type="checkbox" role="checkbox" />`  
`<img alt="Send" aria-label="Send" />`
6. **Change states and properties in response to events.** If we indicate that a tree is open with `aria-expanded="true"`, when the user closes it, we must change the property to `false` using JavaScript, so that the assistive technology can announce that it is now closed.
7. **Keyboard accessible.** In HTML, the elements that can take keyboard focus by default are form elements, buttons, and links. Any other element that we want to provide interaction with (a *div*, a list element, etc.) must include the attribute `tabindex="0"` (or `tabindex="-1"` if we only want it to take the focus programmatically).
8. **Synchronize the visual interface with the accessible interface.** If an item changes its status from selected to unselected, it changes its style as well:

```
.treeitem[role="treeitem"] [aria-selected="true"] {
 color: #000;
 background-color: #f2f2f2; }

.treeitem[role="treeitem"] [aria-selected="false"] {
 color: #f2f2f2;
 background-color: #000; }
```

9. Although it is no longer necessary for the page to work without active JavaScript, it is enough to make it natively accessible, program using **non-intrusive JavaScript**, and follow the principle of **progressive enhancement**.
10. If you apply the ARIA standard, **always review it** using some of the tools listed in the next section, which must include listening to the page with a screen reader.



[\*\*The ARIA Authoring Practices Guide \(APG\)\*\*](#)<sup>91</sup> provides examples for each type of widget, with the correct application of the related roles, properties and states, and their proper keyboard behavior.

<sup>91</sup> <https://www.w3.org/WAI/ARIA/apg/>

## Example 3. Validation of a required field

Let's imagine that we validate the fields of our form when they lose focus, and if we detect a validation error, we display a layer with an alert message. This involves performing field-by-field validation.

To make it as accessible as possible, we can use ARIA in the following way:

```
<!--Field that caused the error -->
<label for="c1">Name*:</label>
<input type="text" id="c1" name="c1" aria-invalid="true"
aria-required="true" value="" aria-errormessage="error">
<!--Error layer -->
<div id="error" aria-live="assertive" role="alert">
 <p>
 [...]
 The Name field is required.
 </p>
</div>
```

The error layer has *role="alert"* and the attribute *aria-live="assertive,"* which means that as soon as the layer's content is modified, the screen reader announces "Alert" and reads its content (although the focus is already on the next field).

A vector icon in SVG format precedes the error message. This element has the attributes *role="img"* and *aria-label="error:"*. This causes the icon to be announced as an image labeled "error:".

The field has the attributes *aria-required="true"* and *aria-invalid="true"* so that, when it gains focus, the screen reader announces that the field is mandatory and that its input is invalid.

We associate the error with the field using the *aria-errormessage* attribute so that when the field gains focus again, the reader will announce its error.

It is important that the error description includes the name of the field "The <Name> field is required", because it reads it when the next field gains focus. If the error message is "The field is required" it may seem to refer to the current field, and not to the field that has lost focus, which is the one that has been validated.

# How to Review ARIA

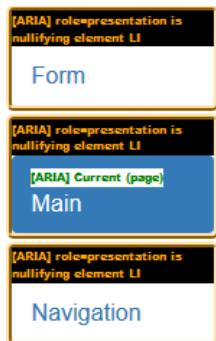
## With a screen reader

The best way to check our page is **to access it with a screen reader** such as NVDA, JAWS, or VoiceOver. Only then can we hear how each element is announced, if it is understandable, if it needs more information, and the differences when including ARIA roles and attributes. In the chapter [Work tools](#) we recommend different screen readers depending on the platform.

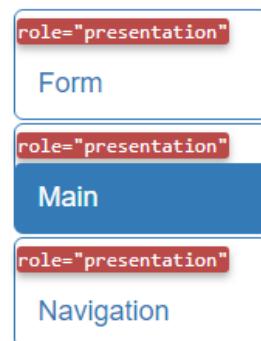
## Reviewing ARIA's attributes

Another option is **to audit** whether we correctly apply ARIA according to the specification. For example, [Lighthouse](#)<sup>92</sup> or [AXE DevTools](#)<sup>93</sup>, which are installed as tabs in the Chrome browser's "Developer Tools," evaluate the correctness of the ARIA code, that is, the formal aspects specific to ARIA: whether we are using an attribute that is not allowed for a given role, whether the role has its mandatory attributes, whether the attributes have correct values, whether the role has the necessary parents or children, etc.

We also have browser extensions that **highlight the ARIA roles and attributes** that have been applied on the page, such as [Visual ARIA](#)<sup>94</sup> or the [Web Developer Toolbar](#)<sup>95</sup>.



Screenshot 22 Visual ARIA  
highlighting roles, states, and  
properties



Screenshot 23 Web Developer Toolbar  
highlighting roles

---

92

<https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=es>

<sup>93</sup> <https://chrome.google.com/webstore/detail/axe-devtools-web-accessib/lhdoppojmngadmnnindnejefpokebjdd>

<sup>94</sup> <https://whatsock.com/>

<sup>95</sup> <https://chrispederick.com/work/web-developer/>

## Inspecting the Accessibility Tree

Another tool for reviewing ARIA is the direct inspection of the accessibility tree. To introduce this concept, we must first explain how browsers work in more depth.

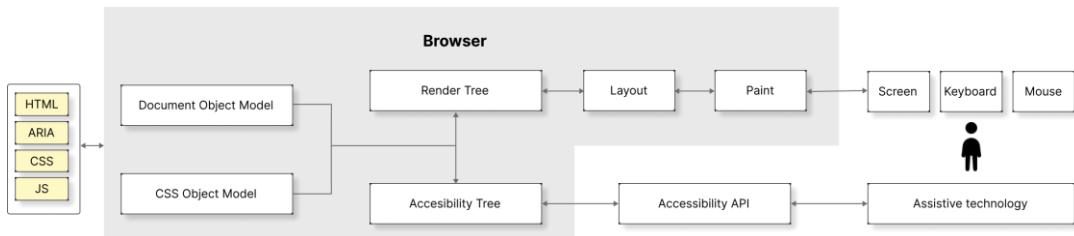


Illustration 40 Complete scheme of operation of the browser and the supporting product through the Accessibility API; and the rest of the peripherals through the render tree.

First, browsers convert HTML markup into an internal representation called a **DOM** (*Document Object Model*). The DOM contains objects that represent all the elements, attributes, and text nodes in the markup.

Similarly, browsers convert CSS properties into an internal representation called **CSSOM** (*CSS Object Model*) that allows JavaScript to manipulate the CSS.

Browsers then create a **render tree** and an **accessibility tree** based on the DOM tree and CSSOM.

To create the **render tree**, the browser loops through each visible node starting from the root. Nodes that are not visible (such as metadata) or hidden by CSS are ignored because they are not reflected in the rendered output.

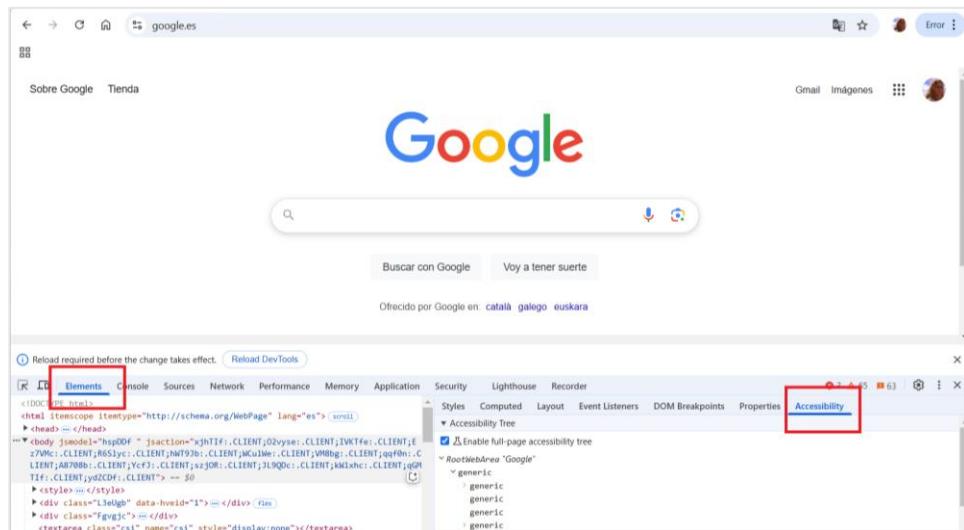
Next, the render tree moves on to the "**Layout**" phase or definition of the positions and sizes of the elements to be displayed, and then to the "**Paint**" phase where the screen pixels are changed color according to what the render tree commands.

Browsers create an accessibility tree, based on the DOM tree, that is used by platform-specific accessibility APIs to provide a representation that can be understood by assistive products, such as screen readers. To create the **accessibility tree**, the browser collects accessibility-related information for most elements in the DOM. Note that when ARIA roles override native language semantics, there are no changes to the DOM, only to the accessibility tree.

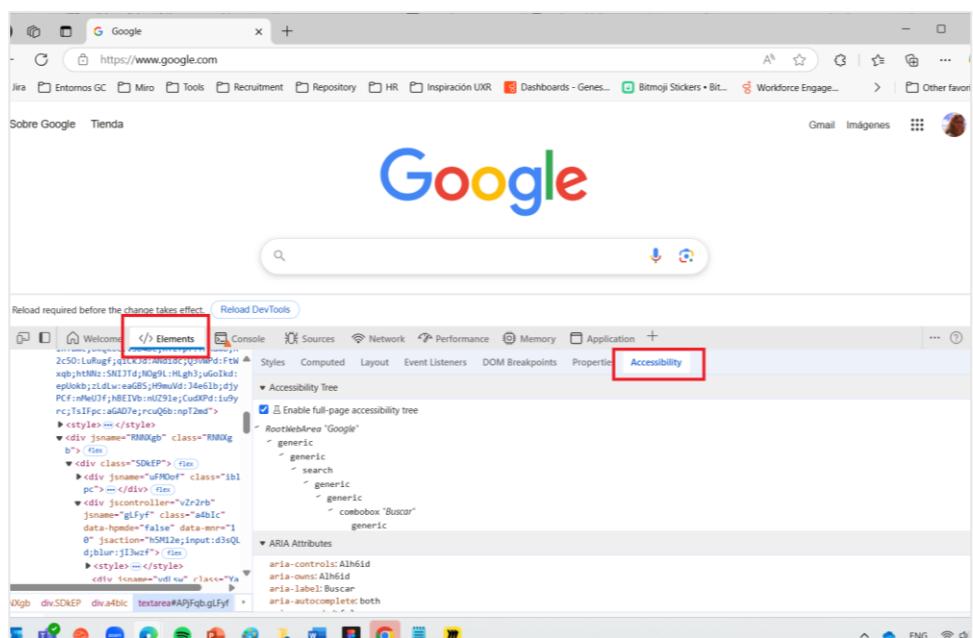
Specifically, there are four relevant properties in the objects in the accessibility tree:

- **Name of the object.** For example, a link with the text "Read more" will have "Read more" as its name (if no other name has been indicated with ARIA).
- **Description.** For example, a table can have a description with the attribute "*summary*" (deprecated in HTML 5) or the attribute *aria-describedby*.
- **Role.** For example, an object can have a button, image, link, header function, and so on associated with it.
- **State.** For example, in a checkbox whether it is checked or cleared.

It is very useful to manually inspect the accessibility tree to check all the information associated with an object. To do so, we must open the developer options offered by browsers. You can find it in both Google Chrome and Microsoft Edge in the "Elements > Accessibility" tabs.



Screenshot 24 Google Chrome Accessibility Tree



Screenshot 25 Microsoft Edge Accessibility Tree

In the image below, we show the calculated properties *of the Google search field*.

The screenshot shows the element inspector of a browser. On the left, the HTML code for a search input field is visible, including attributes like `jsname="yZiJbe"`, `class="gLfyf"`, `jsaction="paste:puy29d;"`, `id="APjFqb"`, `maxlength="2048"`, `name="q"`, `rows="1"`, `aria-activedescendant`, `aria-autocomplete="both"`, `aria-controls="Alh6id"`, `aria-expanded="false"`, `aria-haspopup="both"`, `aria-owns="Alh6id"`, `autocapitalize="off"`, `autocomplete="off"`, `autocorrect="off"`, `autofocus role="combobox"`, `spellcheck="false"`, `title="Buscar"`, `type="search"`, and `value="Buscar"`. The `aria-label="Buscar"` attribute is present. On the right, the calculated properties panel lists various properties such as `Name: "Buscar"`, `aria-labelledby: Sin especificar`, `aria-label: "Buscar"`, `De label: Sin especificar`, `title: "Buscar"`, `placeholder: Sin especificar`, `aria-placeholder: Sin especificar`, `title: "Buscar"`, `Descripción: "Buscar"`, `Rol: combobox`, `Entrada de usuario no válida: false`, `Enfocable: true`, `Especifica: true`, `Editable: plaintext`, `Puede establecer el valor: true`, `Tiene Autocompletar: both`, `hasPopup: listbox`, `Obligatorio: false`, `Expandida: false`, and `Controles:`.

Screenshot 26 Google Search Field Properties Detail

The field has a "`title`" attribute and an ARIA "`aria-label`" attribute with the same "Search" value. To determine the name, the "`aria-label`" takes precedence, while in this case the description comes from the "`title`" attribute.

To understand this process, two recommendations from the W3C are essential, as referenced in the WAI-ARIA standard, which all user agents should respect to ensure a consistent experience across all of them:



[Accessible Name and Description Computation 1.1](#)<sup>96</sup>: This document describes how user agents determine the name and description of objects.

[Core Accessibility API Mappings 1.1](#)<sup>97</sup>: This document describes how user agents should expose the semantics of web content languages to the Accessibility API.

We continue to analyze the properties of the field and see that its role is "combobox". The tag used is "textarea", not "input" as you would expect, but once we apply it with ARIA the attribute `role="combobox"` becomes a "combobox".

<sup>96</sup> <https://www.w3.org/TR/accname-1.1/>

<sup>97</sup> <https://www.w3.org/TR/core-aam-1.1/>

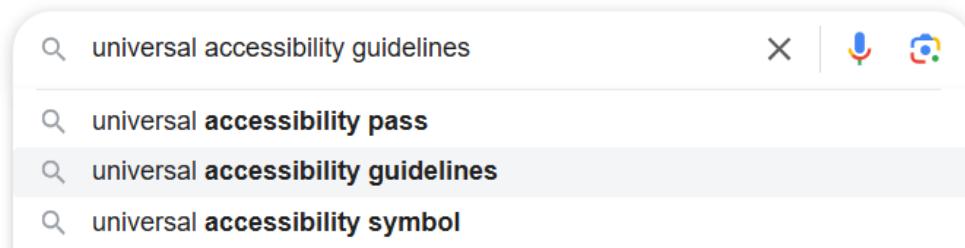
A combobox is an input that controls another item, such as a drop-down list or pivot table, that can appear dynamically to help the user set the value of that input. In this case, the Google search field shows suggestions when the field gains focus or is typed within it.

Some of its attributes are inferred; for example, it is not considered a mandatory control because the attribute `aria-required` is not included (its default value is `false`, meaning not mandatory). Other properties are explicitly defined with the attributes included in the field.

Let's review its ARIA attributes:

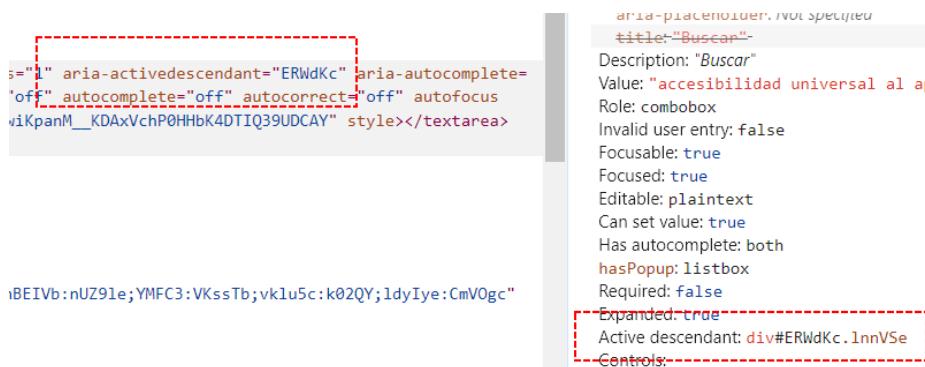
- **`aria-autocomplete`**: This indicates that entering text will trigger the display of suggestions. The `both` value means that it offers two possible models: a prediction of completion of the value written in the field (*inline*) and the list model (*list*).
- **`aria-controls`**: This associates the field with the component it deploys by code.
- **`aria-expanded`**: This indicates whether it is expanded or contracted, a value that changes dynamically.
- **`aria-haspopup`**: This indicates the type of element that is displayed. It has the value `both`, which is not one of the values supported by the standard, so the browser interprets its value as `listbox` and reflects it in the accessibility tree. You can detect this syntax error with the *AXE DevTools* validator, which we discussed earlier.
- **`aria-owns`**: This reflects the visual, functional, or contextual parent/child relationship with another element (in this case the element it displays) when this relationship cannot be represented in the DOM hierarchy.
- **`aria-activedescendant`**: In the tabs example we explain how to handle keyboard focus using "roving tabindex". Well, `aria-activedescendant` provides an alternative method to manage keyboard focus on interactive elements that can contain multiple focusable descendants, such as `combobox`. Instead of moving the focus between the elements, we set the DOM focus on the `combobox`, and using `aria-activedescendant` we indicate the active element in the list.

When you access the component with the keyboard, the focus remains on the field, although with the arrow keys on the keyboard you can scroll through the suggested options in the list:



Screenshot 27 Using the keyboard to select an option from a list

When you scroll with the arrow keys on the keyboard through the items in the list, the *aria-activedescendant* attribute changes to reflect where the effective focus is, although the DOM focus remains on the field:



Screenshot 28 Active-descendant attribute in the Accessibility tree inspector

## Inspecting the information that comes into the Accessibility API

---

In the previous section, we explained how we can consult the accessibility tree generated by the browser.

Another option is to verify the information provided by accessibility APIs in various operating systems.

There are several accessibility APIs on Windows, including *UI Automation* and *Microsoft Active Accessibility* (MSAA). You can try Microsoft's free tool [Inspect](#)<sup>98</sup> to inspect the information.

["Accessibility Inspector"](#)<sup>99</sup> is a developer tool provided by Apple as part of the Xcode developer tools, which provides detailed information about UI elements and their properties.

In Linux environments, [Accerciser](#)<sup>100</sup> is an open-source accessibility debugging tool for GNOME that allows you to test and verify the accessibility of applications.

---

<sup>98</sup> <https://learn.microsoft.com/en-us/windows/win32/winauto/inspect-objects>

<sup>99</sup> <https://www.deque.com/blog/intro-accessibility-inspector-tool-ios-native-apps/>

<sup>100</sup> <https://help.gnome.org/users/accerciser/stable/introduction.html.en>

# The specific techniques of ARIA

WCAG 2.2 includes 23 techniques<sup>101</sup> specific to ARIA technology:

1. Describe UI controls with the *aria-describedby* property.
2. Identify required fields with the *aria-required* property.
4. Use roles to report the function of each UI component.
5. Report the status of each interface component with ARIA states and properties.
6. Label objects with *aria-label*.
7. Define the purpose of links with *aria-labelledby*.
8. Define the purpose of links with *aria-label*.
9. Create a label by concatenating multiple text nodes with *aria-labelledby*.
10. Use *aria-labelledby* to provide alt text to non-text content.
11. Use *landmarks* roles to identify areas on the page.
12. Identify headers with *role="heading"*.
13. Name regions and *landmarks* with *aria-labelledby*.
14. Provide invisible labels with *aria-label* when you can't use visible labels.
15. Describe the images with *aria-describedby*.
16. Provide a name to UI controls with *aria-labelledby*.
17. Use grouping roles (such as *role="group"* and *role="radiogroup"*) to identify related form controls.
18. Identify errors with *role="alertdialog"*.
19. Identify errors with *role="alert"* or *live regions*.
20. Identify regions of the page with *role="region"*.
21. Identify fields with errors with *aria-invalid*.
22. Present status messages with *role="status"*.
23. Identify sequential information updates with *role="log."*
24. Semantically identify font icons with *role="img"*.

---

<sup>101</sup> Technique 3 was integrated into technique 5 in 2014.

## ARIA Guides

- [\*Full specification WAI-ARIA 1.2\*](#)<sup>102</sup>
- [\*Official documentation of ARIA techniques\*](#) <sup>103</sup>
- [\*"WAI-ARIA 1.2 Cheat Sheet" by Raghavendra Satish Peri\*](#)<sup>104</sup>

---

<sup>102</sup> <https://www.w3.org/TR/wai-aria>

<sup>103</sup> <https://www.w3.org/WAI/WCAG22/Techniques/#aria>

<sup>104</sup> <https://www.digitala11y.com/wai-aria-1-1-cheat-sheet/>



# Accessible Single Page Applications

On many websites, clicking on links doesn't actually navigate to new pages; instead, much or all of the content on the page is dynamically updated within the same URL.

You may not even realize it... unless you access it with a screen reader or just the keyboard.

A blind user's experience in a *Single Page Application* (SPA) where accessibility hasn't been addressed can be disastrous: they hear only silence when browsing, they are unsure of their location, and the keyboard focus becomes unpredictable.

In this chapter, we explain how to remedy it in a simple way.



# What is a SPA

A *Multiple Page Application* (MPA) and a *Single Page Application* (SPA) are two different approaches to building websites and web applications.

The traditional and most common approach is a *Multiple Page Application* (MPA), where each page is loaded separately from the server when the user requests a new page. That is, when you navigate from one page to another, the application makes a request to the server to get the new page complete, so that browsing involves reloading the entire page.

In contrast, in a *Single Page Application* (SPA), instead of loading entire pages from the server, a single page is loaded in the browser, and as the user interacts with the application, the necessary data is loaded, and the user interface is dynamically updated. You can dynamically change only part of the content, for example, the main region, or sometimes the entire page's content.

Therefore, the main difference lies in handling navigation and content loading. An MPA involves reloading entire pages when browsing, while a SPA loads the app once and dynamically updates content based on people's interactions.

An example of a SPA web application is Gmail, but also websites such as Zara, or Pinterest.

The main accessibility problems of SPAs are:

- a. Status messages **are not included** when the page is modified or are not announced by screen readers.
- b. The **page title** does not change when new content is uploaded.
- c. Keyboard **focus** is not handled properly when the page is modified.

Now let's see how to fix them.

# Status messages

When you access a traditional website with a screen reader, it announces the title of the loaded page. However, in a SPA, there is only silence because the page has not really been loaded; only the content has been modified dynamically.

To avoid this problem, we must work with status messages.



WCAG 2.2 does not require us to have status messages, only to mark them properly with ARIA so that screen reader announces them.

If you are limited to strict compliance with WCAG 2.2, SPAs may be inaccessible with the screen reader. So, we need to include status messages.

The good practices to follow are:

1. **Include a status message whenever you dynamically modify content.** It is best to make the message visible to everyone.
2. **Include ARIA attributes** so that the screen reader announces the message, even if the message is not in focus (see [Criterion 4.1.3 Status Messages](#)):

```
<div role="status" aria-live="polite"><p>Loading data
from the page... </p></div>
```

If the message contains only one animation, it must have *alt* text:

```
<div role="status" aria-live="polite"></div>
```

3. **Dynamically modify the status message** to "Page [page title] has loaded" when the data is loaded.

The message that the page has already loaded is not usually left visible to everyone, only to people who access it with a screen reader. In these cases, remember that you should NOT hide the message with the `display:none` or `visibility:hidden` styles because they also hide the content for the screen reader.

# Dynamic page title

On a traditional MPA website, each page that loads has its own page title in the `<title>` tag of the `<head>`.

However, in an SPA, there is only one page whose content is dynamically modified; therefore, although it seems that different pages are loaded, there is only one with its initial title. As a result, **it is essential that the page title is dynamically modified**, i.e., the content of the `<title>` tag of the `<head>`.



Dynamically changing the page title **doesn't automatically cause the screen reader to announce it as a status message**, so we still need to include the status messages we've already discussed.

Public access SPA websites often dynamically modify the title of pages for search engine optimization purposes. However, **it is common to find errors when the page title is not relevant to that positioning**. For example, we can find several pages of a purchase process that maintain the same title or that the pages within the customer area, with restricted access, always maintain the title of the initial page.

The page title is important for a screen reader user, who can ask for it at any time to navigate, for example, with NVDA or JAWS, using the shortcut `Insert+t`.

A clear title is also important for anyone who accesses a website, since the title is the page's name displayed in the browser tab, when you bookmark it, or when you share it on social media.

There's another reason why the title may be relevant in a SPA. We've already indicated that we should include a page-loaded message: "Page [page title] has loaded." If we use the content of the `<title>` tag to dynamically generate the page-loaded message, it will be very confusing when the screen reader announces that different pages have been loaded with the same name.

A unique, descriptive title for each page provides clearer status messages when the title is used to generate them: "The 'Checkout. Step 1 of 3. Customer data' has loaded", "The page 'Purchase process. Step 2 of 3 Shipping address' has loaded", etc.

# Keyboard Focus Control

Many people access pages using pushbuttons or keyboards only, including those using a screen reader. Many of these people are blind and can't see where the focus is. These people do not know that they are in a SPA, and they do not have to know.

On a traditional website, the keyboard focus is at the beginning of the page when the page loads. However, navigating in a SPA where accessibility has not been worked on can be very confusing when you depend on the keyboard and cannot see the screen. Keyboard focus often becomes unpredictable, leaving you uncertain where it will go.

It is common to find yourself in situations where keyboard focus is lost when new content is loaded, as the element with the focus disappears and is no longer present. Other times, the focus stays where it is after the loading of a new page is announced, generating the expectation that it should be located at the beginning of it.

Therefore, the third key to making a SPA accessible is **to handle correctly the keyboard focus when the content is loaded**. To do it well, you must apply common sense, know how users of screen readers and keyboards access; and involve them in testing whenever possible.

---

## Option 1. Send keyboard focus to the top of the page

If you are dynamically modifying all the core content, one option may be **to send the focus to the top of the page**, which is what the user expects. In particular, it is advisable to place it in the first link of the <body>, i.e. the "Skip to content" link (see [criterion 2.4.1 "Bypass blocks"](#)).

This way, after the status message announces "Page [page title] has loaded", the focus is always predictably placed at the top of the page, on the link to jump to the central content.

Therefore, the screen reader user has the necessary feedback, and it is predictable behavior. On the other hand, the keyboard user who views the page and does not use a support product does not stay focused on a position from which it is difficult to return to the top of the page or to the main region. For example, the keyboard focus does not remain in the footer after clicking one of its links.

## Option 2. Focus on the page title

In other cases, sending the focus to the page title, that is, to the `<h1>` in the `<main>`, can provide a better user experience.

In the following example, you can see two pages of an accessible course from *isEazy*. When you click "Start Course" or "Next Section," it does not navigate to another page but dynamically modifies the content of the page:

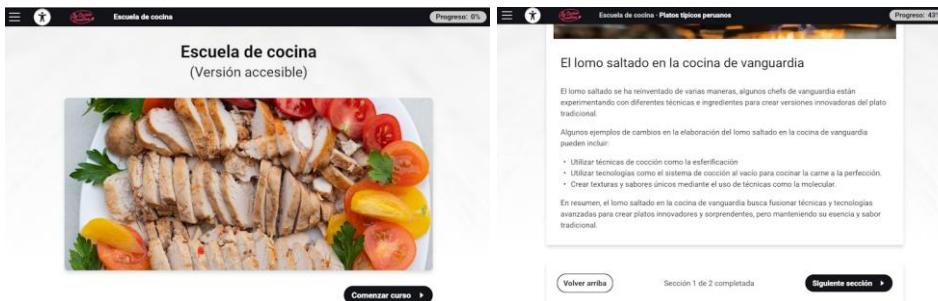


Illustration 41 Pages of an accessible course *isEazy*

Considering the specific elements of the course header and that the student's objective is to advance through the contents of the course, a suitable alternative when dynamically changing the page's content is to send the focus to the `<h1>`. In this way, the user advances quickly, fluidly, and consistently.

In this case, it is also appropriate to simplify the page loading message to "The page has loaded" or "The content has loaded", since the page title (the `<h1>`) then takes the focus, so the screen reader will already announce it and it is not necessary to repeat it in the status message.

When sending keyboard focus to an element, keep these points in mind:

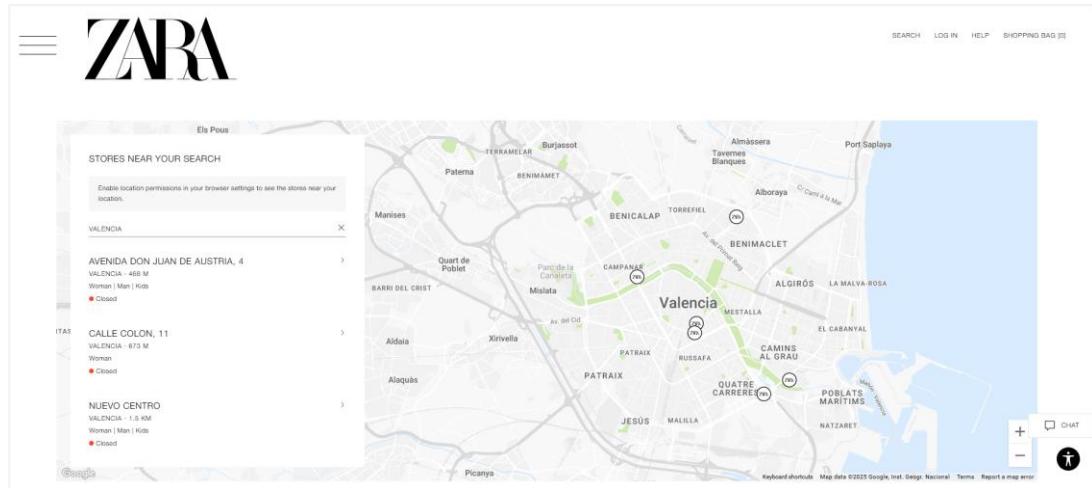
- **If the element that gains focus is NOT an interactive element**, as in the case of `<h1>`, it must have the `tabindex="-1"` attribute, never `tabindex="0"`.
- If you add `tabindex="0"` it will take the focus when tabbing around the page.
- If you add `tabindex="-1"` it will only take the focus programmatically, but never when tabbing around the page, which is the correct behavior.
- The **visual indication of which element has the focus must be clear**. Avoid limiting yourself to a simple color change. For example, the element could be highlighted with a border.

---

### Option 3. Maintain keyboard focus on the tapped interactive element

In other cases, when dynamically modifying only a portion of the content, moving the focus to the top of the page or to the title is not the best solution.

Imagine you have a search engine in the main region of the page, and only the search results are dynamically modified, like on this page:



Screenshot 29 An internal search engine on a page that dynamically reloads the results.

In this case, it may be more appropriate for the **keyboard focus to stay on the "Search" button**, as shown in the example above, so that the next tab takes us to the first search result.

On the other hand, the "Showing 9 stores" message must have the ARIA status message attributes that we discussed earlier. This way, the screen reader announces the message, even if the focus has not moved.

# Final recommendations

**Three key recommendations** to make a SPA accessible are:

- **Inform** the user of what is happening through status messages correctly implemented with ARIA, so that they can be announced to users of assistive products.
- **Dynamically modify** the <title> of the <head> to reflect the content displayed in each case.
- **Control** keyboard focus when loading new content so it's predictable and consistent, and creates the best user experience for keyboard users, who may or may not see the screen.

**Other recommendations** to consider when addressing the accessibility of a SPA are:

- Remember **to modify the breadcrumb** appropriately to reflect the current page.
- If you mark the **current menu or step**, make sure to mark the correct one when editing the content.
- You'll need **to set a delay** for the screen reader to announce everything properly. We must ensure that the screen reader loads the new content into the buffer; that gives you time to read the status message before reading the content that takes focus, without interruptions or abrupt changes.



# **Accessible Design Systems**

Digital transformation has brought organizations the need to create countless online services.

Maintaining consistency between them is an almost impossible task without a system that helps them reuse components from one project to another.

If we make these components accessible, we will take a significant step toward building accessible websites almost effortlessly.



# What is a Design System

In large (and not so large) organizations, it is common for several UX/UI designers and developers to collaborate for different projects, and many times they will have to face similar situations: forms, headers, navigation menus, links... In the absence of clear and shared guidelines, every UX/UI designer and developer will be forced to spend a considerable amount of time defining, deciding, designing, and programming every single element of an interface, even the seemingly simple ones.

These uncertainties pose challenges to UX/UI designers and developers when creating digital products, resulting in wasted time searching for specific styles for each element or component.

1. How much **time and effort has been invested** in the design and development of each element?
2. How frustrated do you think the members of the design and development team will feel? And the project managers when they see that theirs **doesn't look like that of other** colleagues?
3. And how do the people who will access the website feel that they must dedicate part of their mental resources to **learning how each interface is handled**?
4. Is the image that the organization is transmitting good or bad?

To avoid these problems, design systems were born, a tool that allows the team to establish reusable patterns to create new projects and extend functionalities. Design systems are much more than a library of elements shared by UX/UI designers and developers, they **are a way of working, processes, documentation, and a team, which, based on shared principles, allow the scalability and standardization of projects from small pieces.**

And, if these small pieces take accessibility into consideration, we can say that the creation of accessible projects becomes easier, since we will have a large part of the way done by building the pages with elements already validated.

# How to Design and Develop an Accessible Design System

The design systems are organized following Brad Frost's model of atomic design, a methodology composed of six stages, as shown in the Illustration 42, from the smallest to the largest.

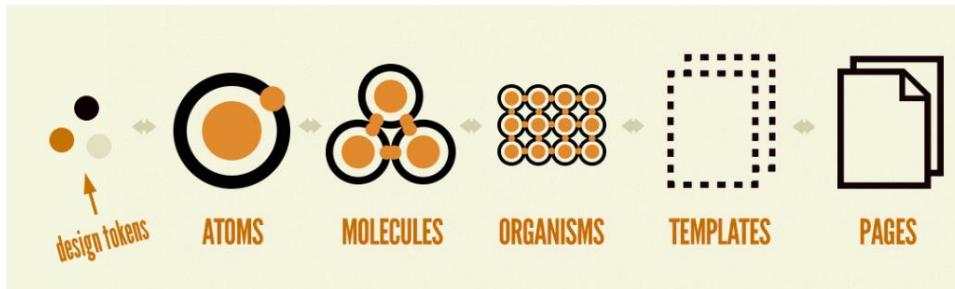


Illustration 42 Stages of atomic design. Source: [Extending Atomic Design | Brad Frost](#)<sup>105</sup>

Although atoms are the smallest units of matter, **design tokens** would be the specific values of the characteristics of these elements, such as the weight of a font or the hexadecimal value of a color.

Atoms are the bases on which the design is based and would correspond to the identity of the organization. Acting on the accessibility of atoms is essential, as it is the first stone on the road. We must select:

- **A combination of colors** (foreground and background) with sufficient contrast.
- **Fonts** that facilitate reading.
- **Icons** that are explanatory and have sufficient contrast with the background. In development, they must be implemented in such a way that it can be decided whether their alternative text is mandatory or not, depending on the context in which they are used.
- A simple **tone of voice** that is compatible with clear writing.
- **Sufficient spacing** to organize the information clearly with an obvious hierarchy.

The next level would be **molecules**, i.e., the union of several atoms that together would form the building blocks of HTML. Some of the variables that we must monitor in this step, following what is defined in WCAG 2.2, would be:

- **Headings** should be larger and/or heavier the more important they are, i.e., H1 should be more prominent than H2; H2 more than H3; and so on until H6.
- The **texts** must be:
  - large enough to be read;
  - able to be resized;
  - structured avoiding long blocks.

<sup>105</sup> <https://bradfrost.com/blog/post/extending-atomic-design/>

- **Links**, in addition to being visually highlighted in order to distinguish them from normal text, must include a way to identify their purpose.
- Accessible **data tables** are usually one of the most challenging elements to achieve, especially if they are complex, due to the relationships between headers and cells or how they should respond to different screen sizes, especially mobile phones.
- **Form elements**: input fields, checkboxes, buttons, error notices... Although they are not complicated, they are laborious due to the different cases they have (active, inactive, focused, error...). In terms of design, you have to consider the contrast of colors, the simplicity of use, the operating instructions, the validations, the autocomplete, and the gestures that must be made if they have a dragging movement... In development, they must take care of the labels that identify the purpose of the fields, the ARIA attributes if they need them, how they behave...
- **Images**, both decorative and content, how they should be displayed, whether they should have *alt* text or not, and how to implement it.

Once the molecules have been decided, the next step is the **organisms**, that is, the components that unite several elements in a single concept, such as the page header, the navigation menus, the breadcrumbs... Here, the accessibility cases are even larger since each component must be evaluated for its own characteristics, also following WCAG 2.2. For example, a carousel must be able to be operated by keyboard; it must be able to stop; the structure must be able to be programmatically determinable; transitions should not have annoying blinking or jerky movements; changes must be announced by ARIA to screen readers; images must (or should not) have *alt* text...

Another example of a complex component is forms. Not only do you have to consider the complexity of each field separately, but also how they interact with each other, how they are grouped, how they are validated, if they have a time limit, how they are divided into different pages...



If you want to check the accessibility requirements of carousels, forms, and other common components, we recommend the [W3C component tutorials](#).<sup>106</sup>

<sup>106</sup> <https://www.w3.org/WAI/tutorials>

The next level would be the construction of **templates**, pre-made, and already developed designs ready to introduce the content. Here we must verify how the different items behave together: their hierarchy, their keyboard operation, the order of the focus, and that it is not trapped in hidden layers. In addition, we must check that the language of the page is identified, that it works correctly on different screen sizes and orientations, that the help is always in the same position, that the template's title describes its purpose... That is, what has already been seen in the WCAG 2.2 criteria.

The last level would be the **pages** that are effectively built from all the items created. If all the previous steps have been taken with certainty, much progress will have been made in the accessibility of the final set. Still, it cannot be 100% guaranteed since it is necessary to verify that the specifications given in the documentation have been followed and that the orchestration of the items is correct. For this reason, it should be checked, both with automatic tools and manually, and, if possible, with people with various disabilities.

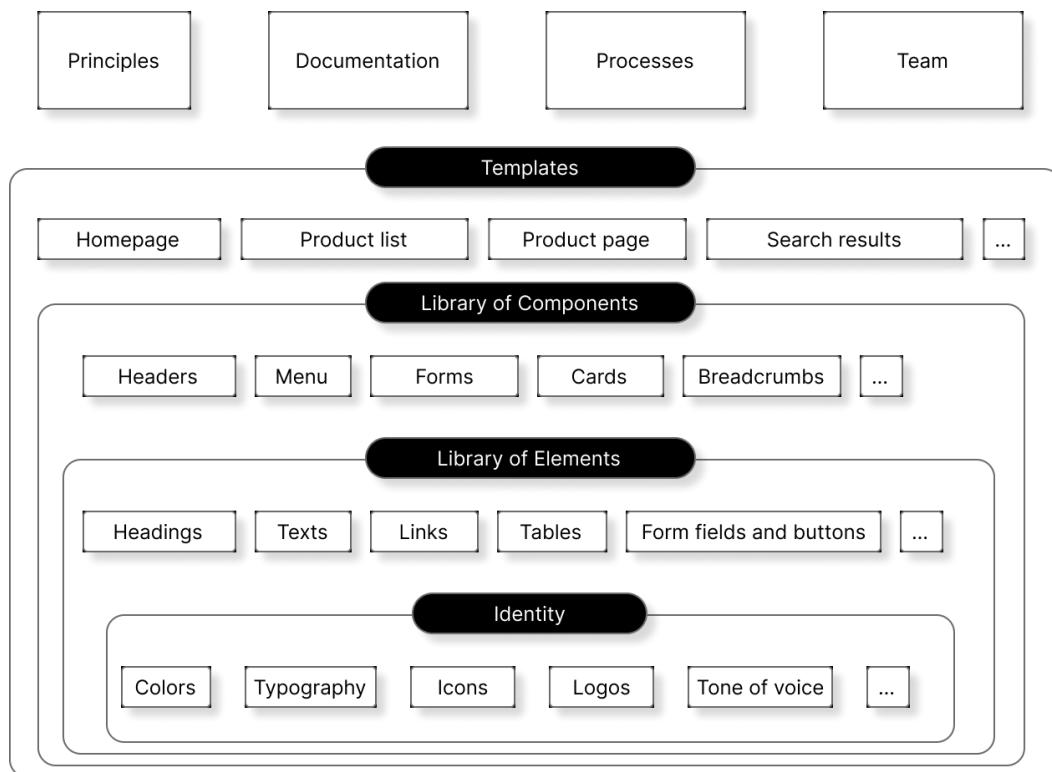


Illustration 43 Basic components of a design system

# Documenting the Design System

Design systems are much more than a ready-to-use UI kit in Figma, Sketch, or other interaction design tools. They include documentation that explains the use of each item, both from a design and development standpoint. Above all, design systems facilitate communication between both parties, so they must present the most relevant information in a clear and easy-to-understand way.

The most important design systems consider accessibility one of their pillars and make real efforts to ensure that this is reflected in the final product. Let's see how various design systems reflect this through a commonly used element, such as checkboxes.

[Google's Material 3 design system](#)<sup>107</sup> displays information on how the item is designed, technical specifications, how it is used, and behavioral guidelines. In addition, it dedicates a page to the accessibility characteristics of this element: how it is used with assistive technology, how it is operated with a keyboard, and what code characteristics must be considered to make this happen.

[Microsoft Fluent 2 design system](#)<sup>108</sup> describes the layout, behavior, and content associated with the check boxes. With respect to accessibility, it only includes a paragraph explaining the use of *label* in the groups of boxes. Although it only dedicates a brief paragraph, the rest of the page describes the best practices expected from a checkbox to make it usable and accessible: short texts, easy to scan, predictable operation, logical order...

[Apple's Human Interface Guidelines](#)<sup>109</sup> present all the selector elements on the same page, and the checkboxes only present some generic use cases, without commenting on anything specific about accessibility. However, it does incorporate extensive [developer documentation](#)<sup>110</sup> and how to include accessibility features in general.

Another way of applying accessibility in a design system is the way used by the [DESY, from the Government of Aragon \(Spain\)](#)<sup>111</sup>. As a product of a public administration in the European Union, DESY is required to comply with the EN 301 549 standard, based on WCAG 2.1. To this end, DESY integrates numerous casuistic, documented in clear language and includes a section of good practices for each element.

<sup>107</sup> <https://m3.material.io/components/checkbox/accessibility>

<sup>108</sup> <https://fluent2.microsoft.design/components/web/react/checkbox/usage>

<sup>109</sup> <https://developer.apple.com/design/human-interface-guidelines/toggles>

<sup>110</sup> <https://developer.apple.com/documentation/Accessibility>

<sup>111</sup> <https://desy.aragon.es/componente-checkboxes.html>

In the last example, the [Government of Spain's design system, Dintel](https://dintel.redsara.es/DINTEL/elementos-de-interfaz/componentes/checkbox.html)<sup>112</sup> indicates that it meets the necessary criteria according to WCAG 2.1, although changes in content and styles may affect accessibility. It also offers a handy checklist to ensure that the item meets the success criteria associated with it.

Finally, websites that document DESY and Dintel must comply with public accessibility regulations, so they are a great testing ground for their own elements and components. We recommend reviewing other [Government Design Systems](https://www.supernova.io/blog/top-10-government-design-systems-enhancing-digital-public-services)<sup>113</sup> to check how they apply the accessibility requirements.

As we can see, each organization maintains different documentation adapted to its needs, with more or less detail and with more or less focus on accessibility. The important thing is that each individual item is validated and, when it is used through different webpages, it remains as accessible as conceived in the system. In addition, if an accessibility improvement is made to the item, this improvement will be automatically propagated to the pages where that item has been used.

---

<sup>112</sup> <https://dintel.redsara.es/DINTEL/elementos-de-interfaz/componentes/checkbox.html>

<sup>113</sup> <https://www.supernova.io/blog/top-10-government-design-systems-enhancing-digital-public-services>

The screenshot displays the Google Material 3 design system interface, specifically the 'Components' section. On the left, a sidebar lists various UI components like App bars, Badges, Buttons, Cards, Carousels, Checkboxes, Chips, Date pickers, Icons, Inputs, Lists, Menus, Navigation, Progress indicators, Radio button, Search, Sheets, Sliders, Snackbars, Switch, Tabs, Text fields, Time pickers, and Tooltips. The 'Checkboxes' item is highlighted.

The main content area is titled 'Checkbox' and contains the following sections:

- Overview:** A brief description: "Checkboxes let users select one or more items from a list, or turn an item on or off".
- Specs:** A detailed specification table.
- Guidelines:** Accessibility guidelines.
- Use cases:** A list of user requirements:
  - Users should be able to:
    - Navigate to a checkbox with assistive technology
    - Toggle the checkbox on and off
    - Get appropriate feedback based on input type documented under [interaction & style](#)
- Interaction & style:** Examples of checkboxes being selected via text labels and groups, with notes on their states (selected, unselected, indeterminate).
- Keyboard navigation:** A table of keyboard shortcuts:
 

Keys	Action
Tab	Moves to the next element in a group
Space	Toggles a focused checkbox between selected and unselected
- Labeling elements:** A diagram showing the accessibility label for a checkbox, which clearly states the text label of the checkbox.

Screenshot 30 Checkbox in Google's Material 3 design system

**Fluent 2** Search / Q Go to Fluent 1

- > Get started
- > Design language
- > Guidelines
- Web**
- > Components
  - Overview
  - Accordion
  - Avatar
  - Avatar group
  - Badge
  - Button
  - Card
  - Checkbox**
  - Combobox
  - Dialog
  - Divider
  - Dropdown
  - Field
  - Fluent provider
  - Icon
  - Image
  - Info button
  - Input
  - Label
  - Link
  - Menu
  - Persona
  - Popover
  - Progress bar
  - Radio group
  - Select
  - Skeleton
  - Slider
  - Spin button
- Preview

Email me

Daily updates  
 Updates from my groups  
 New followers  
 Likes on my posts  
 Messages in my inbox  
 Weekly updates  
 Highlights from my organization  
 Highlights from my groups  
 Suggestions for people to follow

**Indeterminate state**

If one checkbox can control a subset of options, selecting or clearing the parent checkbox selects or clears all of the children. Use an indeterminate state to show when some, but not all, of the options are selected.

Create a visual separation between parent and child checkboxes to increase scannability. For example, indenting the children makes it easy to infer what will happen if someone selects the parent.

## Accessibility

Checkbox groups should be associated with a descriptive `label` component. Do this by rendering a separate `label` and either nesting the checkbox within it or using `for` and `id` attributes to associate the elements.

Passing the `label` prop within the checkbox provides a text label for single checkboxes.

## Content

1. Group label (optional) describes a group of checkboxes.  
 2. Checkbox label (required) clearly communicates what will happen if the checkbox is selected.

Screenshot 31 Checkbox in the Microsoft Fluent 2 Design System

**Design**

Overview What's new Guidelines Resources

particular, avoid using these components in a toolbar or status bar.

**Switches**

Avoid using a switch to control a single detail or a minor setting. A switch has more visual weight than a checkbox, so it looks better when it controls more functionality than a checkbox typically does. For example, you might use a switch to let people turn on or off a group of settings, instead of just one setting.

In general, don't replace a checkbox with a switch. If you're already using a checkbox in your interface, it's probably best to keep using it.

**Checkboxes**

A checkbox is a small square button that's empty when the button is off, contains a checkmark when the button is on, and can contain a dash when the button's state is mixed. Unless a checkbox appears in a checklist, a descriptive label follows the button.

Use a checkbox instead of a switch if you need to present a hierarchy of settings. The visual style of checkboxes helps them align well and communicate grouping. By using alignment — generally along the leading edge of the checkboxes — and indentation, you can show dependencies, such as when the state of a checkbox governs the state of subordinate checkboxes.

Consider using radio buttons if you need to present a set of more than two mutually exclusive options. When people need to choose from options in addition to just "on" or "off," using multiple radio buttons can help you clarify each option with a unique label.

Consider using a label to introduce a group of checkboxes if their relationship isn't clear. Describe the set of options, and align the label's baseline with the first checkbox in the group.

Accurately reflect a checkbox's state in its appearance. A checkbox can be in a selected, deselected, or mixed state. If you use a checkbox to globally turn on and off multiple subordinate checkboxes, show a mixed state when the subordinate checkboxes have different states. For example, you might need to present a text-style setting that turns all styles on or off, but also lets people choose a subset of individual style settings like bold, italic, or underline. For developer guidance, see [allowsMixedState](#).

**Radio buttons**

A radio button is a small, circular button followed by a label. Typically displayed in groups of two to five, radio buttons present a set of mutually exclusive choices.

A radio button's state is either on (a filled circle) or off (an empty circle). Although a radio button can also display a mixed state (indicated by a dash), this state is rarely useful because you can

Screenshot 32 Checkbox in the design system *Human Interface Guidelines* from Apple

**GOBIERNO DE ARAGÓN**

**DESY**  
Sistema de Diseño del Gobierno de Aragón

Buscar en DESY

Inicio Cómo empezar Estilos Componentes Patrones Plantillas Desarrollo

**COMPONENTES**

- + Componentes principales
- Formularios
- Árbol
- Área de texto
- Bloques de datos
- Botón de radio
- Calendario
- Carga de archivos
- Casilla de verificación**
- Entrada de texto
- Selector
- + Navegación
- + Mostrar y ocultar
- + Datos
- + Avisos
- + Información visual

**Formularios**

## Casilla de verificación

Este elemento ofrece la posibilidad de seleccionar varias opciones de las que se ofrecen.

Uso [Código Nunjucks y HTML](#) [Código Angular](#)

**Anatomía**

Se componen de una caja y de una etiqueta con el nombre de la opción.

La alineación puede ser tanto en vertical como en horizontal, siendo preferible siempre que sea posible la alineación vertical.

**Variantes**

Por defecto	Condicional	Con líneas divisorias
<input type="checkbox"/> Item 1	<input type="checkbox"/> Item 1	<input type="checkbox"/> Item 1 Lorem ipsum dolor sit amet
<input type="checkbox"/> Item 2	<input type="checkbox"/> Item 2	<input type="checkbox"/> Item 2 Lorem ipsum dolor sit amet
<input type="checkbox"/> Item 3	<input type="checkbox"/> Item 3	<input type="checkbox"/> Item 3 Lorem ipsum dolor sit amet

**Por defecto**  
Es el que se recomienda usar en primera instancia.

**Condicional**  
Se usa en los casos en los que hay contenido adicional. En este caso, una vez se marca la opción, despliega un input abrazado con una linea a la izquierda.

**Con líneas divisorias**  
En general, no se recomienda su uso. Se emplea cuando el contenido es extenso y se requiere hacer una división visual de los elementos.

**Tamaños**

Tamaño base	Tamaño pequeño
<input type="checkbox"/> Item 1	<input type="checkbox"/> Item 1
<input type="checkbox"/> Item 2	<input type="checkbox"/> Item 2
<input type="checkbox"/> Item 3	<input type="checkbox"/> Item 3

**Tamaño base**  
Utilizar por defecto para cualquier tipo de situaciones y layouts. Siempre que sea posible se recomienda usar este tamaño por motivos de accesibilidad.

**Tamaño pequeño**  
Se recomienda su uso cuando el elemento estándar sea demasiado grande para el espacio donde debe de encajar, por ejemplo dentro de tablas, cards, grupos de filtros o similares.

**Accesibilidad**

**Cómo funciona**

- Grupo rodeado con la etiqueta `<fieldset>` y su `<legend>`.
- Cada ítem del grupo es un `<input>` con `type="checkbox"`. Deben tener asociado un `<label>` a través de un `id` único.
- Cuando se navega con teclado, la persona puede desplazarse por la lista con el tabulador y activar el ítem con la barra espaciadora.

Basado en el [patrón checkbox de W3C](#)

Más información sobre los inputs en el apartado [accesibilidad de formularios](#).

**Qué tener en cuenta**

Qué usar	Dónde	Función
fieldset	Parámetro Nunjucks dentro del componente	Utiliza el <code>legend</code> de su interior como título. Debe ser visible.
idprefix	Parámetro Nunjucks dentro del componente	Permite las llamadas entre <code>label</code> , <code>hint</code> , <code>error</code> y el <code>input</code> . El propio componente se encarga de construir un <code>id</code> único para cada ítem.

**Buenas prácticas**

- Por defecto, todas las casillas de verificación deben mostrarse desseleccionadas.
- Liste las opciones en un orden racional y que tenga sentido para el usuario.
- Si el conjunto de opciones es mayor a 5, es preferible emplear un elemento de selección o desplegable.
- El input de entrada no debe limitarse al botón de caja. La persona también debe poder seleccionar la opción pulsando en la etiqueta.

Screenshot 33 Checkbox in the DESY design system of the Government of Aragón

The screenshot shows the Dintel design system's component documentation for the 'Casilla de verificación' (Checkbox). The top navigation bar includes links for 'Español', 'Inicio', 'Primeros pasos', 'Fundamentos', 'Elementos de interfaz', 'Actualizaciones', and 'Contacto'. A sidebar on the left lists various UI components: Elementos de interfaz, Módulos, Plantillas, Componentes (Accordion, Alerta, Enlace Volver, Botón Ir Arriba, Miges de par, Botones, Tarjeta), Casilla de verificación, Expendible, Selector de fecha, Divisor, Desplegable, Cargador de archivos, Menú horizontal, Campo de texto, Enlace, Lista, Cuadro de diálogo, Campo numérico, Paginación, Barra de progreso, Radio, Búsqueda, Selector, Spinner, Pasos, Tabla, Pestanas, Etiqueta, Textarea, Selector de hora, Notificación, Tooltip, Árbol, Menú vertical, and Casilla de verificación (highlighted in blue).

**Casilla de verificación**

Las casillas de verificación o checkboxes se utilizan cuando hay opción de seleccionar varios elementos de una lista. Los usuarios pueden seleccionar cero, uno o cualquier número de elementos.

**Vista previa**

Label \*  Option 1  Option 2  Option 3  Option 4  Option 5  Option 6

**Usos**

**Cuando usar**

- En formularios: a página completa, modales o paneles laterales.
- En filtros de página, menú o dentro de un componente.
- En tablas de datos para edición por ítems.
- En páginas de términos y condiciones para indicar si está de acuerdo con los términos.
- En el recorriatorio de contraseñas para el login.
- En listas con subsecciones.

**Cuando no usar**

- Si sólo puede seleccionarse una opción de la lista, debe utilizarse el botón de opción para que el usuario pueda seleccionar sólo una opción.

**Comportamiento**

**Tipos de casillas de verificación**

Label *	Simple	Label *	Button
<input type="checkbox"/> Option 1 <input checked="" type="checkbox"/> Option 2		Option 01	Option 02

**Orientación**

Label *	Horizontal	Label *	Vertical
<input type="checkbox"/> Option 1 <input checked="" type="checkbox"/> Option 2		<input type="checkbox"/> Option 01 <input checked="" type="checkbox"/> Option 02	

**Tipos de casillas de verificación en formularios**

Label *	Con texto
<input type="checkbox"/> <input checked="" type="checkbox"/>	Option 1 <input checked="" type="checkbox"/> Option 2

**Estados**

Default	Hover
<input type="checkbox"/> Option 1	<input type="checkbox"/> Option 1

Focus	Select
<input checked="" type="checkbox"/> Option 1	<input type="checkbox"/> Option 1

Select All	Disabled
<input type="checkbox"/> Option 1	<input type="checkbox"/> Option 1

Select Disabled	Select All Disabled
<input type="checkbox"/> Option 1	<input type="checkbox"/> Option 1

Error
<input type="checkbox"/> Option 1 ● Error message

**Accesibilidad**

Este componente ha sido validado para cumplir con los requisitos de accesibilidad la UNE-EN 301549:2022 en la que se engloban los criterios de la WCAG 2.1 (AA), sin embargo, los cambios del contenido y estilos pueden afectar al cumplimiento de la accesibilidad. Debemos de asegurarnos de revisar y seguir las directrices de esta sección cuando actualicemos o añadamos nuevo contenido o estilos a este componente.

• La propiedad label del checkbox tiene role="checkbox".

• El checkbox tiene el atributo type="checkbox" en el elemento con role="checkbox".

• Cuando está marcado, el elemento checkbox tiene el estado aria-checked establecido en true. Mientras que cuando no lo está, se establece en false.

• Si el checkbox o grupo de checkboxes incluye texto estínguo descriptivo adicional relevante, tiene la propiedad aria-describedby establecida en el ID del elemento que contiene la descripción. Se usa para indicar la identificación de un componente que contiene texto descriptivo y/o relacionado con el checkbox.

• La propiedad disabled hace que el componente checkbox esté deshabilitado. Esto evita que los usuarios puedan interactuar con el checkbox y transmitir su estado inactivo a las tecnologías de asistencia.

Screenshot 34 Checkbox in the Spanish Government's Dintel design system



# **Caring for the words**

WCAG 2.2 Guideline 3.1 shows how we should make content easy to read and understand.

In this chapter, we go a step further and explain three different and complementary ways of increasing the quality of information, citizen participation, and respect for diversity.

First, we explain the concepts of communication and plain language, and the keys to achieving them.

Next, we show a step further, easy reading or easy language, a way to reach almost all people.

Finally, we show the power of words to create a better society through inclusive language.



# The importance of words

Often, we have to reread a text because it is difficult to understand. We may not make sense of it; it may include words we do not know, or it may be constructed in a complex way.

This problem affects people who have to deal with legal, financial, administrative, or medical texts on a daily basis without being professionals in these fields. This difficulty is greater for people with intellectual, developmental, or hearing disabilities, people with low literacy, people who are learning the language, or children who do not understand complex concepts, among other profiles.

As a reaction to this problem, two movements have emerged in defense of cognitive accessibility through plain language and easy reading, which, although they share origin, are differentiated concepts.

**Plain language** is part of a larger movement, called **clear communication**. Both methods put the people who are going to read the text first by considering:

- what people want and need to know;
- the level of interest, experience and literacy skills of the people;
- the context in which people will use the document.

**Easy reading** (also called **easy language**) is a method of making information easier to understand for most people, including people with intellectual or developmental disabilities.

That is, plain language can be used for a general audience, while easy reading is used for people who struggle with reading comprehension.

On the other hand, **inclusive language** is a form of expression that seeks to avoid discrimination or exclusion of people because of their characteristics as a result of the way in which language is used. The inclusive language aims to reflect the diversity and equality of society and promote respect and visibility for minority or vulnerable groups. It supports both clear language and easy reading.

# Clear communication

The [Clear Communication Guide](#)<sup>114</sup> (in Spanish) define clear communication as "transmitting relevant information to citizens in an easy, direct, transparent, simple and effective way through any of the different channels [...] and adapted to their particularities".

Plain language is one component within this model, but not the only one. Design, neurolanguage, and the digital field also play a significant role. A new factor appears at the intersection of these fields: artificial intelligence, which learns from interacting with people and converses in our own register.

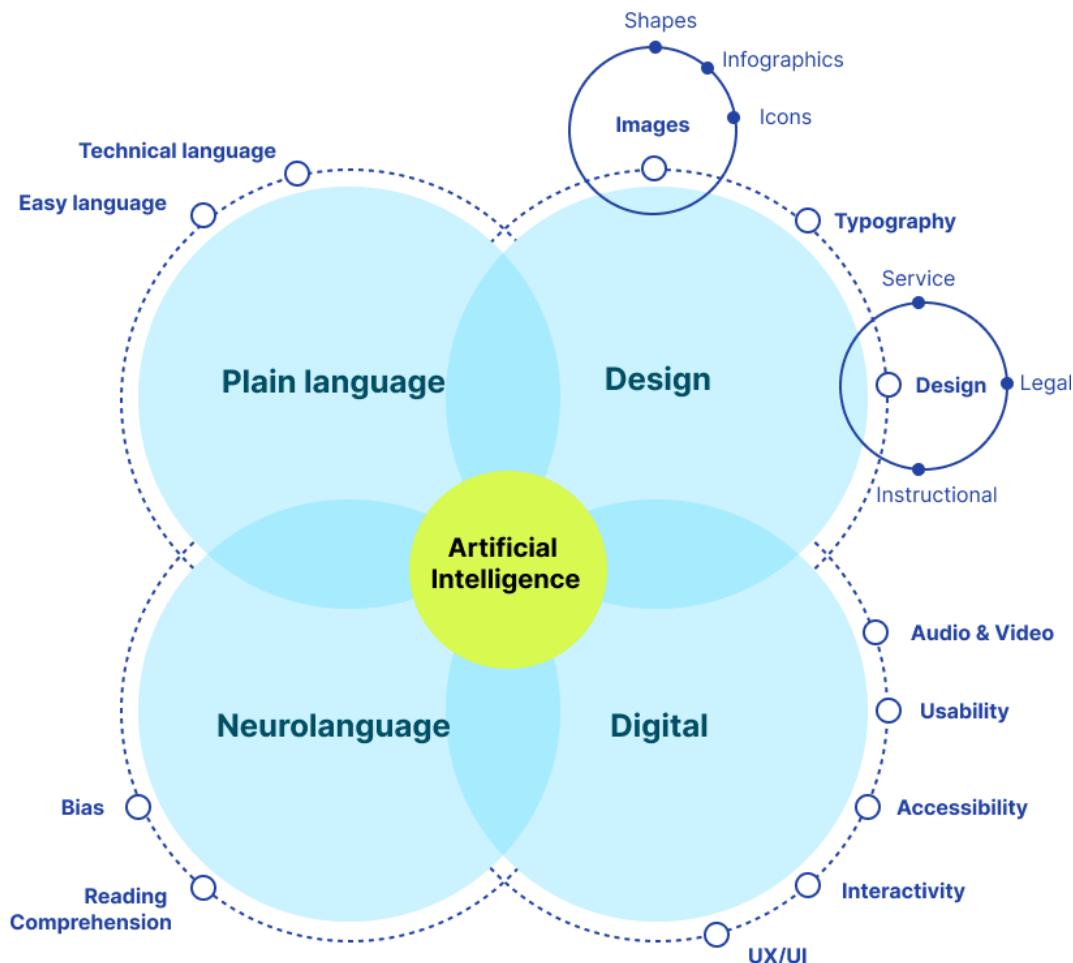


Illustration 44 Clear Communication Components  
(Source: Clear Communication Guide)

<sup>114</sup> <https://comunicacionclara.com/docs/guia-comunicacion-clara-prodigioso-volcan.pdf>

There are nine steps to practicing clear communication, in which people must be at the center of the process and the focus of attention.

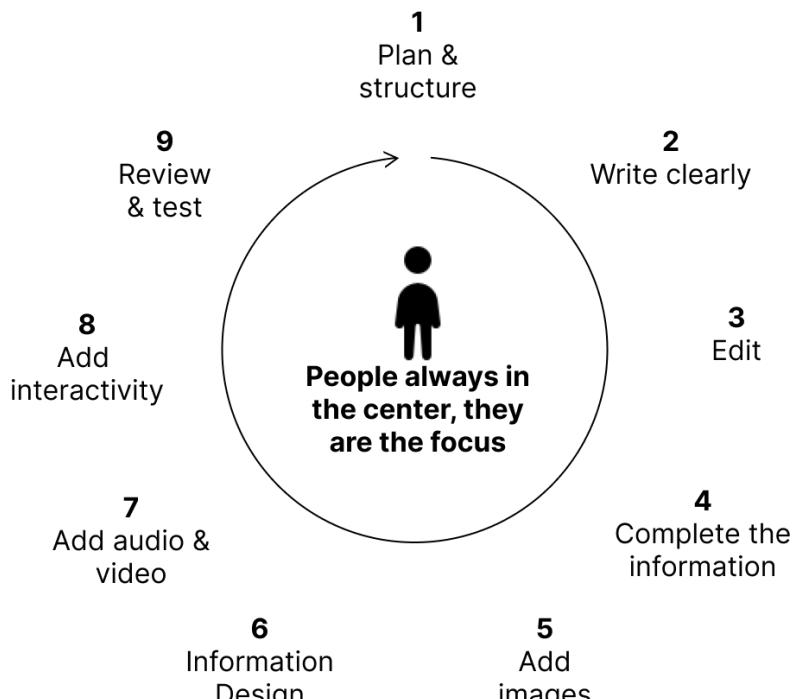


Illustration 45 Clear Communication Steps  
(Source: Clear Communication Guide)

- **Step 1. Plan and structure.** First, we must answer the following questions: What do we want to say? What elements should this information include? How many parts will it have? How will the parts be organized?
- **Step 2. Write clearly.** Writing is difficult because it requires working with ideas as well as words.
- **Step 3. Edit.** It's time to use the typographic resources at our disposal to highlight elements: capital letters, bold, italics, underlines... Always with moderation so that the text does not seem like a catalog of words struggling to stand out.
- **Step 4. Fill out the information.** Use examples, footnotes, tables of contents or indexes...
- **Step 5. Add images.** Icons, graphics, illustrations, or photographs that reinforce the message and make the content more attractive.

- **Step 6. Design the information.** It is time for our information to look clean, orderly, legible, and hierarchical. The composition, the use of spaces, the width of the paragraphs, or the color are some of the tools that we will use to give an impression of simplicity in the design.
- **Step 7. Integrates audio and video.** In the audiovisual era, complementing textual information with any of these resources will improve its attractiveness, usability, and accessibility.
- **Step 8. Be interactive.** On the web there is no linearity that can be presumed to a book, but people enter and leave without control, so our texts must catch them so that they decide to read them.
- **Step 9. Review and test.** Whether it's A/B testing, user testing, or focus groups, the important thing is to know people's opinions first-hand and integrate their feedback if deemed appropriate.

# Writing in plain language

The International Organization for Standards (ISO) has published a specific standard for plain language, [ISO 24495-1:2023<sup>115</sup>](#) (in English and French). Its recommendations fall into four categories:

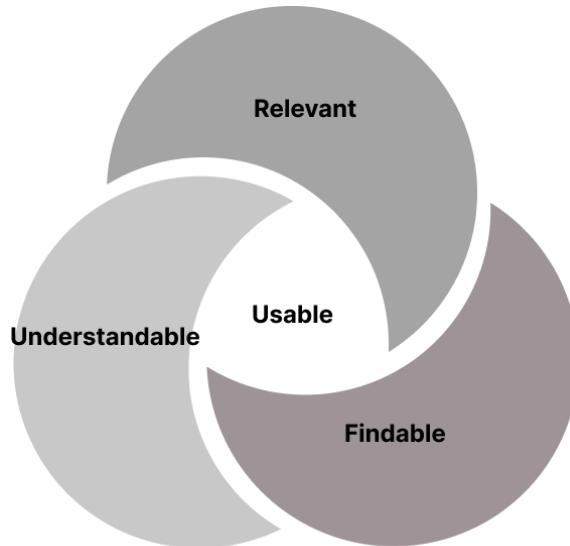


Illustration 46 Plain Language Recommendation Categories According to ISO 24495-1:2023

1. The text is **relevant** to the user's needs. To do this, it is necessary to identify the user who will read the text, its objectives, and its context. It is also necessary to select the type or types of documents that we will present to you and the content that you will need.
2. The reader can **easily find** what they need. To do this, you have to work on the document's structure, use information design techniques, include headings that anticipate the content of text blocks, and separate important information from supplementary information.
3. The reader can easily understand what they find. To do this, familiar words should be used in clear, concise sentences and paragraphs. The tone of voice must be respectful, and the final document must be checked for coherence. In addition, the use of images and multimedia that complement the content of the text should be considered.
4. The reader can **easily use** the information. At this point, it is necessary to evaluate that what is written complies with the above recommendations.

More specifically, the *Clear Communication Guide* gives the following more accurate tips for writing clearly:

<sup>115</sup> <https://www.iso.org/obp/ui/en/#iso:std:iso:24495:-1:ed-1:v1:en>

- Use examples, comparisons, analogies, or metaphors.
- When writing for a screen, synthesize as much as possible. Think that people mainly use mobile phones to access information.
- Control the length of paragraphs and sentences.
- Follow the natural order of the sentences.
- Use the active voice instead of the passive voice.
- Do not abuse gerunds, infinitives, and participles.
- Connect ideas.
- List with ordered lists.
- Choose the right words.
- Avoid technical terms.
- Watch out for the use of abbreviations and acronyms.
- Prevent the accumulation of negation elements.
- Respect the rules of the language.
- Divide the text into sections and blocks.
- Introduce explanatory headings before the sections.
- Ensure a coherent relationship between written text and images.

# Example of communication improvement

The [City Council of Madrid is reviewing the information](#)<sup>116</sup> on its institutional portals and communications with citizens to apply clearer, more concise, and understandable language for citizens. Below are two versions of the same information. It is a letter that is sent to citizens explaining how to pay taxes.

Are you able to identify changes in the text? And in the design of the letter?

## ORIGINAL<sup>117</sup>



## REVISED<sup>118</sup>



Illustration 47 Before and after the letter (text in Spanish; translation to English below)

<sup>116</sup>

<https://www.madrid.es/portal/site/munimadrid/menuitem.0c57021e0d1f6162c345c34571f1a5a0/?vgnextfmt=default&vgnextoid=a01f1905bacde510VgnVCM1000001d4a900aRCRD&vgnextchannel=59af566813946010VgnVCM100000dc0ca8c0RCRD&idCapitulo=10559854&vgnextlocale=es>

<sup>117</sup> Complete original available in

[https://www.madrid.es/UnidadesDescentralizadas/Calidad/LenguajeClaro/ComunicacionClara/Ficheros/ModelocartaliqPAC\\_Inicial\\_Web.pdf](https://www.madrid.es/UnidadesDescentralizadas/Calidad/LenguajeClaro/ComunicacionClara/Ficheros/ModelocartaliqPAC_Inicial_Web.pdf)

<sup>118</sup> Complete original available in

[https://www.madrid.es/UnidadesDescentralizadas/Calidad/LenguajeClaro/ComunicacionClara/Ficheros/ModelocartaliqPAC\\_Final\\_Web.pdf](https://www.madrid.es/UnidadesDescentralizadas/Calidad/LenguajeClaro/ComunicacionClara/Ficheros/ModelocartaliqPAC_Final_Web.pdf)

## **ORIGINAL LETTER**

Dear Mr/Mrs,

You figure as the holder of the À la Carte Payment System (PAC), registration no. 1621529500217, for the 2017 financial year, according to the data contained in our database.

With this communication we inform you of the situation of the receipts attached to it, with an indication of those that have been paid in full and those that remain to be paid, once the payments on account made by you have been applied. And the corresponding bonuses.

Proof of payment of the bills that have been fully paid is sent. The bills that remain unpaid, either in full or partially, will be sent to your bank or savings bank, so that they can be debited from your account on December 15, with the bank sending you the proof of payment. We send you the images of the outstanding bills with all the data and we recommend that you keep this document so that you can complete the data that your financial institution will send you.

Information on the payment schedule for the 2018 financial year will be available soon.

Kind regards

XXXXX

**REFUND OF UNAPPLIED AMOUNTS** Article 55 of the General Tax Ordinance on Management, Collection and Inspection determines that in the event that the advance payments are greater than the sum of the amounts of the settlements adhered to the PAC, the refund will be made ex officio by bank transfer to the same account in which the charges were made.

## **REVISED LETTER**

From the Tax Agency of the Madrid City Council, we inform you about the status of your 2018 On-Demand Payment.

As you may remember, you chose this payment option, by direct debiting your bills from your bank. In this shipment we explain in detail:

- > How much you have paid
- > How much has been saved
- > How much you have left to pay

In addition, we include the information of the outstanding bills and the proof of payment of the bills paid in full through this option. We advise you to keep all this information together with the proof of payment that the bank will send you. You will receive the payment schedule for next year shortly.

Kind regards

XXXXX

How we do the calculation> In order to offer you the Payment on Demand, we calculate the periodic fee to be paid taking as a reference the amounts paid last year. In case this year's installment is lower, don't worry: we will refund it by transfer to the bank account where you have paid by direct debit.

# Text easy to read and understand

The main recommendations published in the standard [ISO 23859:2023<sup>119</sup>](#) on making written text easy to read and understand are:

## Index

- Place a table of contents at the beginning of the document.
- Make it clear on which page each content is.

## Vocabulary

- Use simple and common words suitable for the document's readers.
- Avoid:
  - abstract words;
  - very long or with difficult syllables;
  - exaggerations;
  - words from another language unless they are well known;
  - abbreviations;
  - acronyms, unless they are well known and you explain them;
  - unclear words;
  - words with several meanings;
  - idioms, sayings, ironies, and metaphors.
- Use the same word for the same object or concept throughout the document.
- If you need to use difficult words, use glosses and glossaries to explain them.

## Numbers

- Write the numbers in numbers.
- Avoid:
  - the big numbers. If you have to use them, write them down in print and with examples of comparisons;
  - ordinal numbers;
  - fractions and percentages. Use ideas that are easier to imagine;
  - Roman numerals. If you have to use them, explain them;
  - Write the hours in 24-hour format.
- Split phone numbers into smaller numbers.
- Write the dates in full and with the day of the week written.

---

<sup>119</sup> <https://www.iso.org/obp/ui/en/#iso:std:iso-iec:23859:ed-1:v1:en>

## Punctuation marks

- Avoid:
  - write entire words or phrases in capital letters;
  - rare and unusual signs, such as ( ) % & ... ; " ";
  - "etcetera", you can use other words such as "many more" or "and others".
- Use the 2 dots in front of a list with more than 3 words or phrases.

## Phrases

- Use simple, short sentences, each with a unique idea. If long sentences are necessary, break them down into natural reading points.
- Look for each sentence to occupy a line to facilitate reading.
- Be sure to specify who you're addressing, using pronouns like you, me, they, you, or people's names if any.
- Use simple verbs and, when possible, the present tense.
- Avoid:
  - explanations with commas within a sentence;
  - use 2 or more verbs together;
  - gerunds;
  - giving orders, if you have to, make it clear to whom they are addressed;
  - the passive voice. Use the active voice better;
  - negative phrases. Never use two negative phrases together.

## Organization of the text

- Provide all the necessary information without leaving gaps so that the user does not have to make assumptions.
- Avoid the inclusion of unnecessary details, keeping texts concise and avoiding excessive length.
- Structure the information chronologically, using terms such as "before," "after," or "next" to guide the user through the text.
- Maintain direct communication with the user whenever possible.
- Group related information under the same topics and use white space between sections for clarity.
- Use descriptive titles that indicate the content of the text.
- Organize lists using bullet points for easy reading.
- Present the dialogue in a theatrical format, using scripts and making sure to clearly identify who is speaking.

## Documents layout

- Organize the content of documents by sections.
- Make sure that the lines end in approximately the same place.
- Set a line spacing of at least 1.5 and use wide margins.
- Choose an unadorned typeface (sans serifs) and a minimum size of Arial 14, adjusting it according to your chosen typeface.
- Pick a font color that contrasts appropriately with the background of the page.
- Clearly indicate when the text continues on the next page.
- Number pages with a bigger font and place it consistently on each page.
- Distinguish headings from the rest of the text, either by different sizes or colors.
- Highlight words explained in glosses using bold.
- If there are sections that don't follow the Easy Reading style, let them know.
- Decide on the orientation of the pages (landscape or portrait) and maintain consistency throughout the document.
- Avoid:
  - dividing the entire document into very small sections;
  - writing vertically;
  - letters with ornaments, italics, underlined, shadows, or difficult to read;
  - writing on images and backgrounds with various colors and tones;
  - justifying the text and writing in columns. It's better to align sentences to the left than to the center or right of the page;
  - separating a hyphenated word between 2 lines.

## Images

- Select sharp and specific images, avoiding element saturation. Make sure they have good quality and preferably use them in color.
- Place the images on or to the left of the text they accompany, keeping them close but without hindering reading. Avoid placing them behind the text.
- Use the same image throughout the document to represent a consistent idea.
- Omit usage rights information next to images. In case the image requires explanation, please provide this information in Easy Reading format.

## Graphics

- Include a title or explanation that clarifies the content of the chart.
- Add a legend that indicates how to interpret the information in the chart.
- Limit the explanation in a chart to just two items. If there's more information, consider creating additional charts.

## **Maps and plans**

- Include a title or explanation on the map to guide your content.
- Clearly highlight itineraries and paths.
- Clearly indicate the key elements that the user must identify, including important places. You can use real photos of recognizable spaces and elements.
- Add a legend to make it easier to interpret the map.

## **Glosses or explanations in the margin**

- Use glosses to clarify complicated words that are essential in the document.
- Enter the gloss the first time the word difficult appears in the text.
- Place the gloss as close as possible to the difficult word you're explaining, ensuring they're both on the same page.
- Highlight the word complicated in bold both in the text and in the gloss, clearly differentiating the word from its explanation. You can present the word on one line and the explanation on separate lines.

## **Glossary**

- Use a glossary to define words that the user must understand before tackling the document. Remember that the glossary is not a compendium of all the explanations in the text.
- Place the glossary at the beginning of the document.
- Avoid the word "glossary" if it's complicated; instead, opt for titles like "Dictionary" or "Keywords."
- Organize the words in the glossary in alphabetical order.
- Distinguish the word from its definition using bold fonts and a clear design.
- Illustrate difficult words with examples to make them easier to understand

## **Summaries**

- Use summaries in lengthy documents and repeat key information when necessary.
- Place the summaries at the end of the corresponding section.
- Differentiate the main text and the abstract by using colors, shapes, or other visual elements.

## **Exercises and activities**

- Add memory activities and exercises to address the content.

## Example of easy reading

Take a look at the following news published by the NGO Inclusion-Europe written in easy reading. Inclusion-Europe is an association whose aim is equal rights and the full inclusion of people with intellectual disabilities and their families in all aspects of society.

### Speech of Helena Dalli at the European Day of Persons with Disabilities 2021

Click on a word which is in **blue and bold** to read what it means.



 **Helena Dalli** is Commissioner for Equality at the [European Commission](#).

 A **Commissioner** is a person who is responsible for a field of work at the European Commission.

 Helena Dalli gave a speech at the conference for the [European Day of Persons with Disabilities](#).

 Helena talked about the [EU Disability Rights Strategy](#). The strategy describes what problems people with disabilities face in the [European Union](#).

 The strategy says what the European Union will do to help with these problems.

 **AcessibleEU** will bring together ideas and tools to improve access to places, services, information.

Screenshot 35 News written in easy reading (Source: inclusion-europe.eu)

This story tells a politician's intervention at a conference and covers the main questions of a news story: what happened, who was involved, why is important for the reader, the content of the speech....

Observe some of the characteristics of the "Easy Reading" style in this example:

- Text:
  - Simple words, without technicalities.
  - The style is amazingly simple: subject, verb, and complements.
  - Short sentences, distributed in several lines to facilitate scanning.
  - No abbreviations are used.
  - No unusual punctuation marks are used, just the period at the end of each sentence.
  - No figures are given.
  - The active voice is used, not the passive voice.
- Design and layout:
  - Simple drawings in the margins to reinforce the text.
  - Important words are highlighted in bold.
  - Large print size.
  - The typeface, without serif.
  - Wide margin, clear and simple layout.
  - A single news item per page.
  - Generous line spacing.
  - Aligned to the left.
  - Flat color background, no drawings.
  - High contrast of text and background colors.
  - The number of the page on which it is located is clearly identified.
  - Page size matches a folio
  - Links to definitions of complex concepts

In addition, the photo of the protagonist is in the foreground, reducing the visual noise around her and making her easier to identify. There are no photos of the audience listening to the conference, nor general shots that are more difficult to process.

Finally, the Easy Reading logo is included.

## Process of creating a document in easy language

The process has two stages: the adaptation phase and the validation phase. The standard includes an informative annex that provides examples of activities for the validation phase, which is fundamental in the process and receives great attention in the standard's content.

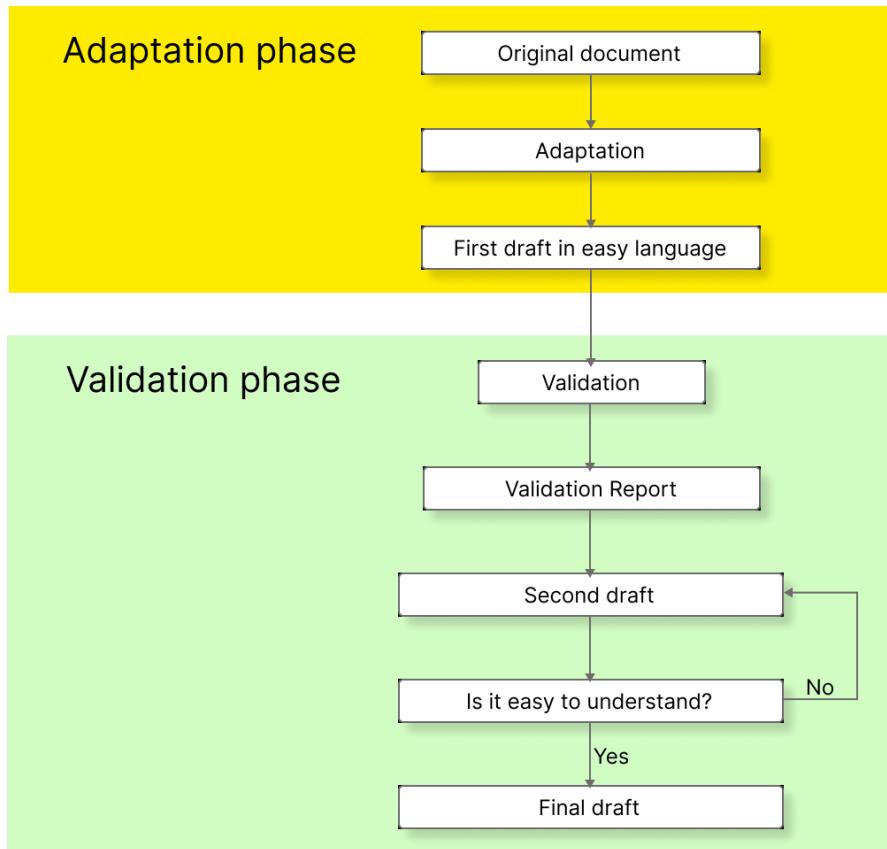


Illustration 48 Process of creating a document in easy reading

In the first phase, the original document is collected, adapted with the participation of the designer and the layout designer, and a first draft is generated that is easy to read.

From there, the validation phase is carried out by expert validators and coordinated by a facilitator. The facilitator generates a validation report with the contributions of the group of validators. The adapter, designer, and layout designer integrate these contributions into a new draft.

The validators recheck this new draft, coordinated by the facilitator and with the collaboration of the adapter, designer, and layout designer. This test can give a negative result, and a new document must be regenerated with the test's contributions.

If the result is positive, it becomes the final draft, ready for publication and the easy-to-read European logo can be used (some terms apply<sup>120</sup>).



Illustration 49 Easy Reading Icon

## Easy-to-read guides by the NGO Inclusion Europe

- [Information for all. European standards for making information easy to read and understand](#)<sup>121</sup>
- [Easy to read Checklist](#)<sup>122</sup>
- [Frequent asked questions about Easy to read](#)<sup>123</sup>

<sup>120</sup> <https://www.inclusion-europe.eu/wp-content/uploads/2021/02/How-to-use-ETR-logo..pdf>

<sup>121</sup> [https://www.inclusion-europe.eu/wp-content/uploads/2017/06/EN\\_Information\\_for\\_all.pdf](https://www.inclusion-europe.eu/wp-content/uploads/2017/06/EN_Information_for_all.pdf)

<sup>122</sup> <https://www.inclusion-europe.eu/wp-content/uploads/2020/06/Easy-to-read-checklist-Inclusion-Europe.pdf>

<sup>123</sup> <https://www.inclusion-europe.eu/frequently-asked-questions-about-easy-to-read-etr/>

# Inclusive language

Through language, societies transmit their beliefs, traditions, and cultural norms from generation to generation. Using words and expressions loaded with prejudice or stereotypes can contribute to the marginalization of certain groups.

We can distinguish inclusive languages for different groups, such as anti-racist, feminist, LGTBIQ+, or people with disabilities, on which we will focus.

Look at the headlines of these news stories. They use non-inclusive language to attract attention. All of them mention people in a pejorative way.



A Crazy Man and His Crazy Gun: Would-Be Trump Assassin's Rifle Details Revealed



Sep 24 • By Octavius Mayhem



Schizophrenic Leicestershire man who killed mother given life sentence



14 Years Later, Judge Drops Charges Against Retarded Man

Oct 8, 2007

## Illustration 50 Headlines using non-inclusive terms

Fortunately, language is an entity in continuous transformation, permeable to different sensitivities and social realities. Aware of the power that language exerts on society, social movements try to advance a language that does not perpetuate negative images or stigmatize them.

However, like any challenge to the established norm, there are conflicting positions and resistance to change, so the evolution of language is sometimes slow. For example, in the Spanish Constitution of 1978, protection was introduced for the "physically, sensory and mentally handicapped", a term that is currently considered

inappropriate, discriminatory, and offensive<sup>124</sup>. Since 2008 it has been recommended to use "person with disability" instead and it has finally been possible to replace the term in the Spanish Constitution in 2024<sup>125</sup>.

It should be avoided to use language in a pejorative way, but to do so, it is necessary, on the one hand, to be aware of the misuse of language and, on the other, to know how to use it more appropriately. In the Disability Inclusive Language Guidelines<sup>126</sup> by the United Nations, guidelines are set out for the use of correct, respectful and consensual language when referring to people with disabilities.

In summary, **people with disabilities**:

- **First and foremost, they are people:** disability is just one more characteristic.
- **They are part of society;** that is, they are members of a group.
- **They are normal:** people without disabilities are not normal, they are only "people without disabilities" or "the rest of the population".
- **They are not supermen or superwomen;** they just do their tasks when they have the appropriate means and their rights are respected.

That's why their top language recommendations are:

- Avoid words with **negative connotations** such as invalid, handicapped or retarded.
- **Avoid negative or sensationalist descriptions**, as they generate compassion rather than "social acceptance based on respect."
- **Avoid euphemisms** when talking about disability, as they are loaded with condescension, generate confusion and legal uncertainty, and reduce the protection of their rights and freedoms.

The following table lists inclusive terms and their equivalent to avoid.

---

<sup>124</sup> <https://sid-inico.usal.es/noticias/una-lucha-de-20-anos-para-reformar-el-articulo-49-de-la-constitucion-y-eliminar-la-palabra-disminuidos/>

<sup>125</sup> <https://boe.es/boe/dias/2024/02/17/> (in Spanish)

<sup>126</sup> <https://www.unigeveva.org/en/about/accessibility/disability-inclusive-language>

Table 12 Examples of inclusive and non-inclusive language

Inclusive language	Non-inclusive language
person with disability	disabled person, handicapped, person with special needs, handicapable, atypical, person living with a disability, differently abled, people of all abilities, people of determination, person living with a disability
person with [type of impairment]	
persons with disabilities	
people with disabilities (only in Easy Read documents, informal text, and oral speech)	
person without disability	normal, healthy, able-bodied, typical,
the rest of the population	whole of sound body/mind
have [disability/impairment/condition]	suffer from, afflicted by, stricken by, troubled with
person with an intellectual disability	retarded, simple, slow, afflicted, brain-damaged, intellectually challenged, subnormal, of unsound mind, feeble-minded, mentally handicapped
person with an intellectual impairment	insane, crazy, maniac, psycho, hypersensitive, lunatic, demented, panicked, agitated, mentally deranged, mentally ill
person with a psychosocial disability	
deaf person	
person who is deaf	
person with a hearing disability	
person with a hearing impairment	
person with hearing loss	
hard-of-hearing person	
deafblind person	
blind person	
person who is blind	
person with a vision/visual disability	
person with a vision/visual impairment	
person with low vision	
deafblind person	
person with a physical disability	
person with a physical impairment	
wheelchair user	
person who uses a wheelchair	
person with a mobility disability	
person with a mobility impairment	
person using a mobility device	
person with a mental health condition	
person with cerebral palsy	
person of short stature	
little person	

Inclusive language	Non-inclusive language
person with achondroplasia (only if the person has this condition)	
person with Down syndrome	
person with trisomy-21	mongoloid, special person, Down
person with albinism	albino
person who uses a communication device	
person who uses an alternative method of communication	non-verbal, can't talk
person with epilepsy, diabetes, depression, or someone who has epilepsy, diabetes, depression	an epileptic, diabetic, depressive
person with a drug/alcohol addiction	
person with alcoholism	addict, alcoholic, junkie
person in recovery and/or remission	
accessible parking	
parking reserved for persons with disabilities	disabled/handicapped parking handicapped bathroom
accessible bathroom	
seizures	fits, spells, attacks

## Inclusive Language Guides

The following documents have been the basis for the previous table, and we recommend reading them:

- [Disability Inclusive Language Guidelines](https://www.un.org/en/about/accessibility/disability-inclusive-language) by the United Nations<sup>127</sup>
- [Inclusive language: words to use and avoid when writing about disability](https://www.gov.uk/government/publications/inclusive-communication/inclusive-language-words-to-use-and-avoid-when-writing-about-disability) by the Government of UK<sup>128</sup>
- [Disability Language Guide](https://disability.stanford.edu/sites/g/files/sbiybj26391/files/media/file/disability-language-guide-stanford_1.pdf) by the Stanford University<sup>129</sup>

<sup>127</sup> <https://www.un.org/en/about/accessibility/disability-inclusive-language>

<sup>128</sup> <https://www.gov.uk/government/publications/inclusive-communication/inclusive-language-words-to-use-and-avoid-when-writing-about-disability>

<sup>129</sup> [https://disability.stanford.edu/sites/g/files/sbiybj26391/files/media/file/disability-language-guide-stanford\\_1.pdf](https://disability.stanford.edu/sites/g/files/sbiybj26391/files/media/file/disability-language-guide-stanford_1.pdf)



# **Validation tools**

**There are no tools that detect automatically all accessibility errors on a website; some criteria require manual testing because they involve meaning, usability, or intent that automated tools cannot fully evaluate yet, even with the AI raise.**

**In this chapter, we present the most relevant applications that help check specific criteria and record the evaluation, both manually and automatically.**



## WCAG 2.2 Audit Tool

It is an Excel tool that allows you to collect the data obtained during the automatic and manual review of a website following WCAG 2.2. The tool generates charts and statistics of compliance and non-compliance by page, level, principle, or criterion of conformance. It also compares the sample's compliance with WCAG 2.0 and 2.1.

This information is helpful for preparing the executive report and presenting the results, comparing websites or results over time, identifying the pages and criteria that present the most problems, or assessing the effort required to comply with the new version of the standard.

In addition, the previous version, in Spanish and English, can still be downloaded for evaluation according to WCAG 2.1.

**Link:** [Audit Tool WCAG 2.2 \(English\)](#)<sup>130</sup>

Audit Tool WCAG 2.2							HELP	Author: Olga Carreras
6. Evaluation results by level and principle, averaging results by page								
* Level AA Compliance % is only displayed if you have indicated on Sheet 1 that you evaluate according to Level AA								
* The % of compliance per principle is calculated based on the number of criteria that each principle has at the level you are evaluating								
Page	% Level A	% Level AA	% Perceivable	% Operable	% Understandable	% Robust		
Alias1	65,22	82,98	88,24	88,24	72,73	50,00		
Alias2	65,22	82,98	88,24	88,24	72,73	50,00		
Alias3	65,22	82,98	88,24	88,24	72,73	50,00		
Alias4	59,26	78,43	84,21	78,95	72,73	50,00		
Alias5	59,26	78,43	84,21	78,95	72,73	50,00		
Alias6	59,26	78,43	78,95	73,68	81,82	100,00		
Alias7	47,62	75,56	78,95	61,54	81,82	100,00		
Alias8	42,86	73,33	73,68	61,54	81,82	100,00		
Alias9	41,18	75,61	73,68	72,73	80,00	100,00		
Alias10	30,43	65,96	73,68	47,06	80,00	100,00		
Alias11	30,43	65,96	73,68	47,06	80,00	100,00		
Alias12	44,00	71,43	77,78	55,56	81,82	100,00		
Alias13	80,00	89,80	83,33	100,00	81,82	100,00		
Alias14	80,00	89,80	83,33	100,00	81,82	100,00		
Alias15	64,00	81,63	83,33	88,89	72,73	50,00		
Average	55,60	78,22	80,90	75,38	77,82	80,00		

Note that in tabs 3.1 and 4.1 the total compliance percentage is made by assessing the sample as a whole. If a single page doesn't meet a criterion, the entire sample doesn't. However, here the percentage is made by mediating the compliance percentages of each page individually, so the % result can vary quite a bit from that of the sample as a whole.

**% Average Compliance Level**

Level	Average Compliance Level (%)
Level A	55,60
Level AA	78,22

**% Average Compliance Percentage by Principle**

Principle	Average Compliance Percentage (%)
Perceivable	80,90
Operable	75,38
Understandable	77,82
Robust	80,00

\* These graphs exclude "not applicable"

Screenshot 36 WCAG 2.2 Audit Tool

<sup>130</sup> [https://www.usableyaccesible.com/recurso\\_descargas.php#informeWCAG2AA](https://www.usableyaccesible.com/recurso_descargas.php#informeWCAG2AA)

# Automatic global validation tools

There are many applications, extensions, and browser bookmarks that perform automatic validations of a web page. We review several of them below.

## Microsoft Accessibility Insights

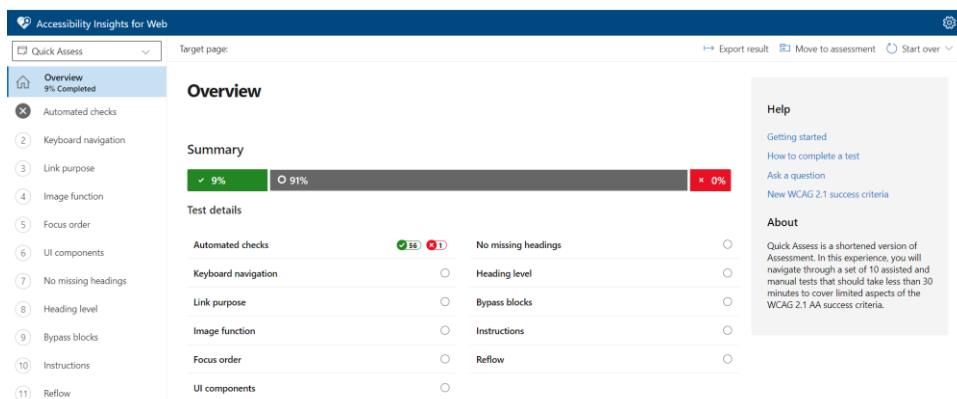
*Microsoft Accessibility Insights* is available for free as a Chrome and Edge extension, and as a Windows app.

The tool supports three main scenarios:

- *FastPass*: This is a quick process that helps developers identify common accessibility issues. The tool performs an automatic evaluation, gives instructions to validate keyboard access and a list of possible errors that must be reviewed manually.
- *Quick Assess*: In addition to automatic assessment, it provides step-by-step instructions for performing ten types of manual checks in an assisted manner: keyboard access, purpose of links, function of images, etc., which can be useful for less experienced testers.
- *Assessment*: In this case, there are 24 types of guided validations.

The report can be exported as HTML and JSON.

Link: [Accessibility Insights](https://accessibilityinsights.io/)<sup>131</sup>



Screenshot 37. *Accessibility Insights*

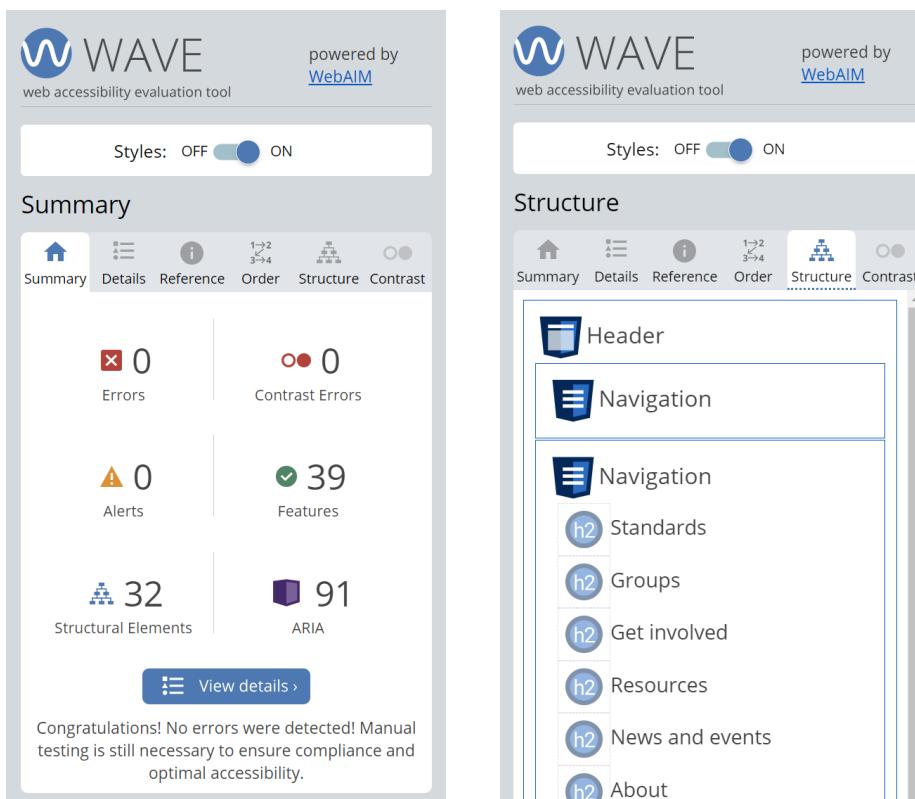
<sup>131</sup> <https://accessibilityinsights.io/>

## WAVE

WAVE is a suite of tools developed by WebAIM, such as the browser extension for Chrome, Firefox, and Edge; or a subscription to WAVE API and local installation.

The browser extension, in addition to performing automatic validations, allows you to review the contrast, disable styles, and inspect the tab order and page structure through their regions and headings.

Link: [WAVE](https://wave.webaim.org/)<sup>132</sup>



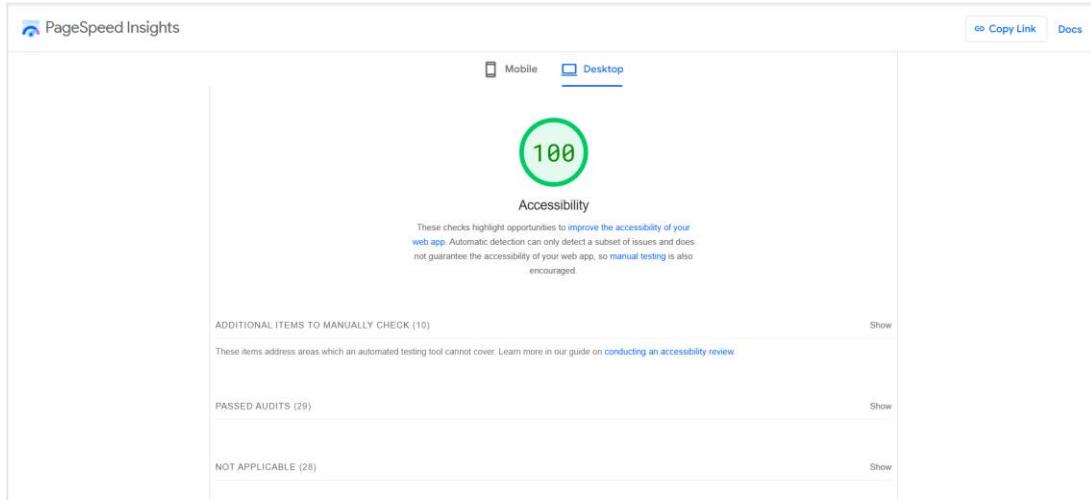
Screenshot 38. WAVE extension.

<sup>132</sup> <https://wave.webaim.org/>

## Lighthouse

This Chrome extension runs a barrage of tests (evaluates performance, accessibility, best practices, and SEO) against the page and then generates a report on how well the page did.

Link: [Lighthouse](#)<sup>133</sup>



The screenshot shows the Lighthouse report interface for a desktop page. At the top, there are tabs for 'Mobile' and 'Desktop', with 'Desktop' being the active tab. A large green circle in the center displays a '100' score. Below the score, the word 'Accessibility' is written. A small note below the title states: 'These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.' Below this, there are three sections: 'ADDITIONAL ITEMS TO MANUALLY CHECK (10)' with a 'Show' link, 'PASSED AUDITS (29)' with a 'Show' link, and 'NOT APPLICABLE (28)' with a 'Show' link.

Screenshot 39. Lighthouse report

133

<https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjmppbjk?hl=es>

## Sitelimprove

The **Sitelimprove Chrome browser extension** is an automatic and free accessibility validator that validates **some criteria (not all)** according to, for the moment, WCAG 2.0. It allows you to filter the results by level (A, AA, AAA), type (error, warning, and manual review), and profile of the user who needs to make the changes (UX/UI designer, webmaster, and developer). It is very reliable in the results, with no false positives.

Link: [Sitelimprove Accessibility Checker](#)<sup>134</sup>

The screenshot shows the Sitelimprove extension in a browser window. On the left, the extension's sidebar displays various accessibility issues categorized by WCAG guidelines (Text Alternatives, Adaptable, Distinguishable, Enough Time, Navigable) with their respective counts. On the right, the main page features a large diagram titled "Las WCAG 2.1 en forma de esquema" (WCAG 2.1 in schematic form) from April 2018. The diagram is a hierarchical tree structure showing the relationship between the four principles of WCAG 2.1: Perceivable, Operable, Comprehensible, and Robust. Each principle branches into specific success criteria, which further break down into detailed guidelines. A watermark at the bottom of the diagram reads "WCAG 2.1 DE FORMA SENCILLA". Below the diagram, a note says "Spoiler: este esquema estará presente en "WCAG 2.1 de forma sencilla", junto con más sorpresas :-)".

Screenshot 40 Sitelimprove Extension (2021 Version)

Sitelimprove also exists as a professional paid online application that allows you to schedule periodic evaluations of entire sites (including PDF documents), integrate with various content managers, generate detailed reports, define custom analyses, in addition to user management, action history and other types of analysis (spelling, search engine positioning, inventories, etc.).

Link: [Sitelimprove website](#)<sup>135</sup>

<sup>134</sup> <https://chromewebstore.google.com/detail/siteimprove-accessibility/djcglobmbegflehmbleechkjhmcdcopn>

<sup>135</sup> <https://siteimprove.com/en/content-accessibility/>

 Siteimprove

30/09/2016

INFORME

## WCAG 2.0 AA Detallado

SITIO  
www.usableyaccesible.com

### Visión general



Total	11 Páginas	10 Problemas A	1 Problemas AA	3 Problemas AAA	4 PDF con problemas
Nombre	Páginas	Problemas A	Problemas AA	Problemas AAA	PDF con problemas
www.usableyaccesible.com http://www.usableyaccesible.com	11	10	1	3	4

### Páginas con problemas

Los datos de esta tabla pueden distorsionar la problemática del sitio. El número de errores que se indica en cada página es el sumatorio de todas las instancias de todos los errores, advertencias y 'necesita revisión manual' de esa página, muchos de los cuales no serán realmente problemas.

Título de página	A	AA	AAA	Nivel de página
Curriculum de Olga Carreras http://www.usableyaccesible.com/index.php	12	10	88	1
Terminos y condiciones de Usable y accesible http://www.usableyaccesible.com/aviso.php	10	0	7	2
Accesibilidad, abajo de heelado y opciones de personalización de Usable y accesible http://www.usableyaccesible.com/accesibilidad.php	11	10	13	2
Mapa web de Usable y accesible http://www.usableyaccesible.com/mapa_web.php	16	9	28	2
Enlaces de Interés sobre usabilidad y accesibilidad web http://www.usableyaccesible.com/recurso_enlaces.php	18	9	124	2
Validadores de accesibilidad y usabilidad web http://www.usableyaccesible.com/recurso_invalidadores.php	31	26	366	2
Libros UX y accesibilidad recomendados y con reseña de Olga Carreras http://www.usableyaccesible.com/recurso_libras.php	48	26	63	2
Glosario de usabilidad y accesibilidad web http://www.usableyaccesible.com/recurso_glosario.php	141	32	1.380	2
Herramientas y material de apoyo para consultorías de accesibilidad o usabilidad http://www.usableyaccesible.com/recurso_descargas.php	14	31	39	2
Consultoría y servicios de accesibilidad y usabilidad web http://www.usableyaccesible.com/servicios.php	10	9	8	2

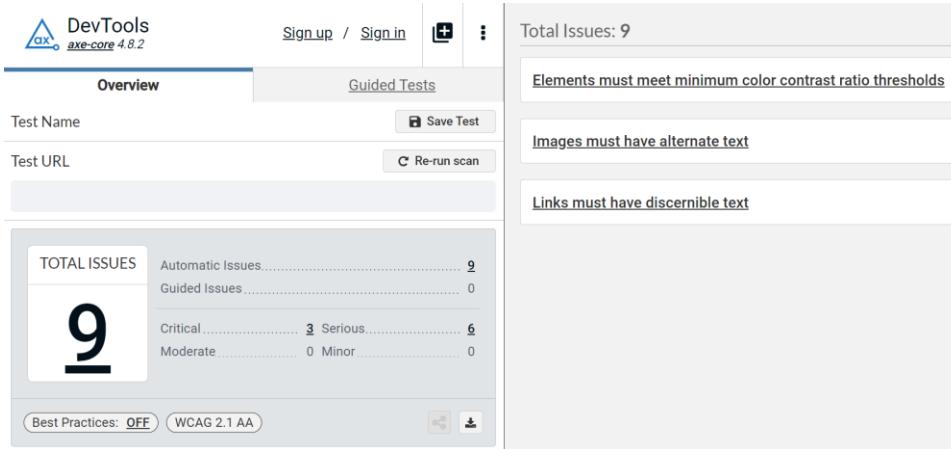
Screenshot 41 [Sample Report](#)<sup>136</sup> generated by the Siteimprove paid app

<sup>136</sup> [http://www.usableyaccesible.com/archivos/informe\\_ejemplo\\_siteimprove.pdf](http://www.usableyaccesible.com/archivos/informe_ejemplo_siteimprove.pdf)

## Axe DevTools

Axe DevTools is a browser extension developed by Deque, offering both free and paid features. Validation following WCAG 2.2 is already included in the paid version.

Link: [Axe DevTools](https://www.deque.com/axe/browser-extensions/)<sup>139</sup>



Screenshot 43. Axe DevTools

Deque also has other tools such as [axe Auditor](https://www.deque.com/axe/auditor/)<sup>140</sup> that helps manual review using heuristics, [axe Monitor](https://www.deque.com/axe/monitor/)<sup>141</sup> for auditor and monitor entire websites, and [axe-core](https://github.com/dequelabs/axe-core)<sup>142</sup>, an accessibility testing engine for websites and other HTML-based user interfaces.

<sup>139</sup> <https://www.deque.com/axe/browser-extensions/>

<sup>140</sup> <https://www.deque.com/axe/auditor/>

<sup>141</sup> <https://www.deque.com/axe/monitor/>

<sup>142</sup> <https://github.com/dequelabs/axe-core>

# Manual validation tools

Although automatic tools and automated accessibility greatly assist in validation, only human reviews can verify all the criteria, even when they have been previously checked with automated tools. For example, a tool can verify that all images have an alternative text, but it cannot determine if that text is suitable and accurately describes the image.

The following table, based on the article “[A Tool’s Errand](#)”<sup>137</sup> by the accessibility specialist Steve Faulkner, outlines the criteria that should be manually checked and the corresponding actions to be performed.

Table 13 Criteria to be validated manually

Criterion	Action
1.1.1 Non-Text Content (A)	Determine if the alternative text is meaningful.
1.3.1 Info and Relationships (A)	Determine if relationships are accurately conveyed.
1.3.2 Meaningful Sequence (A)	Verify if the reading order makes sense.
1.3.3 Sensory Characteristics (A)	Determine if instructions rely solely on sensory characteristics
1.3.4 Orientation (AA)	Confirm that no essential content is lost when orientation changes.
1.3.5 Identify Input Purpose (AA)	Confirm if the assigned purpose is correct.
1.4.1 Use of Color (A)	Verify if color alone is used to convey meaning.
1.4.10 Reflow (AA)	Verify usability when content is reflowed.
1.4.12 Text Spacing (AA)	Ensure readability when spacing is adjusted.
1.4.13 Content on Hover or Focus (AA)	Confirm its usability, persistence, or dismissibility.
2.1.1 Keyboard (A)	Verify full keyboard navigation and operability.
2.1.2 No Keyboard Trap (A)	Confirm if a user is truly trapped.
2.1.4 Character Key Shortcuts (A)	Check for unintended usability issues.
2.2.1 Timing Adjustable (A)	Assess usability and whether time extensions are provided appropriately.
2.2.2 Pause, Stop, Hide (A)	Verify if users can pause or stop the motion.
2.4.3 Focus Order (A)	Confirm logical, intuitive focus flow.

<sup>137</sup> <https://html5accessibility.com/stuff/2025/03/24/a-tools-errand/>

Criterion	Action
2.4.4 Link Purpose (In Context) (A)	Verify that the link text is sensible in context.
2.4.5 Multiple Ways (AA)	Confirm usability.
2.4.6 Headings and Labels (AA)	Determine if they are meaningful and correctly structured.
2.4.7 Focus Visible (AA)	Determine if it is visible enough in various conditions.
2.4.11 Focus Not Obscured (Minimum) (AA)	Confirm usability.
2.5.3 Label in Name (A)	Verify that the label accurately conveys its intended purpose.
2.5.7 Dragging Movements (AA)	Verify if alternative input methods are provided.
2.5.8 Target Size (Minimum) (AA)	Confirm if targets are functionally usable.
3.1.1 Language of Page (A)	Verify if language attributes match the content.
3.1.2 Language of Parts (AA)	Verify if language attributes are correctly applied to multilingual content.
3.2.1 On Focus (A)	Determine if focus-triggered changes are disruptive.
3.2.2 On Input (A)	Assess the predictability of input fields that trigger changes.
3.3.2 Labels or Instructions (A)	Evaluate if form labels are clear, meaningful, and valuable.
3.3.5 Help (AA)	Determine whether contextual help is practical and readily available.
4.1.2 Name, Role, Value (A)	Confirm that these attributes are meaningful and are correctly implemented.

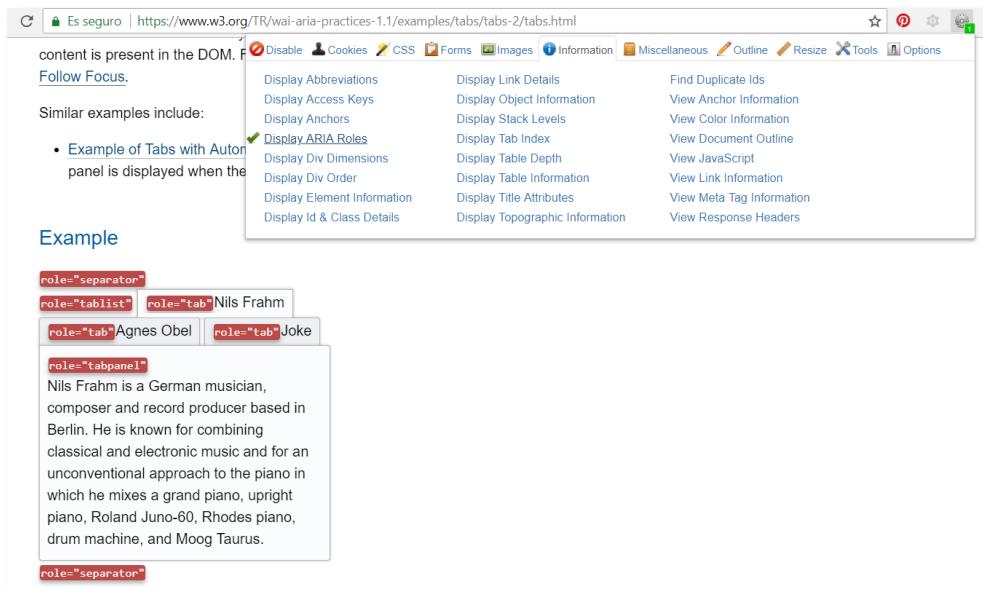
To perform these actions, we recommend using the following browser bars, plugins, and methods.

## Browser Bars

There are various toolbars installed within the browser that help with manual accessibility review, allowing you to disable images or CSS, highlight element attributes, mark the page structure, or display ARIA roles, among other options.

For Chrome, Firefox, and Opera browsers, we recommend the **Web Developer Toolbar**.

Link: [Web Developer Toolbar](https://chrispederick.com/work/web-developer)<sup>138</sup>



Screenshot 42 Browser bar “Web Developer Toolbar”

<sup>138</sup> <https://chrispederick.com/work/web-developer>

## ANDI

ANDI is a tool of the SSA (Social Security Administration) of the United States that is included as a browser bookmark.

It has six functionalities:

- **Elements that receive focus:** You can advance between the different elements that take focus, which are highlighted on the screen. You can see the tab order and the label they have applied.
- **Images:** Identifies images, including SVGs, and allows you to easily validate their code and alt text, telling you how the screen reader announces it.
- **Links and buttons:** You can preview their code and how the screen reader will read it; it also shows you a list of internal and external links with their accessible name and URL.
- **Structure:** You can navigate between headers, lists, *landmarks*, and *live regions* by viewing the code and how it will be announced by the screen reader. You can also check the page title, language, and the "role" and "lang" attributes.
- **Color contrast:** In addition to the contrast ratio and colors used, it tells you the font size. It allows you to view the page in black and white.
- **Hidden content:** You can reveal hidden content and quickly find hidden content with different techniques.

Link: [ANDI](https://www.ssa.gov/accessibility/andi/help/howtouse.html)<sup>143</sup>



Screenshot 44 ANDI Tool

<sup>143</sup> <https://www.ssa.gov/accessibility/andi/help/howtouse.html>

## Chartability

Chartability is a methodology designed to guarantee the accessibility of data visualizations, systems, and interfaces. It is structured around **seven categories**, four of which align with WCAG principles, and three additional categories: *Compromising, Assistive, and Flexible*.

It comprises 50 testable heuristics, 14 of which are critical (listed below):

### Perceivable

- Low contrast
- Content is only visual
- Small text size
- Visual presents seizure risk

### Understandable

- No explanation for the purpose or for how to read
- No title, summary, or caption
- Reading level inappropriate

### Operable

- Interaction modality only has one input type
- No interaction cues or instructions
- Controls override AT controls

### Robust

(no critical heuristics in this category)

### Compromising

No table

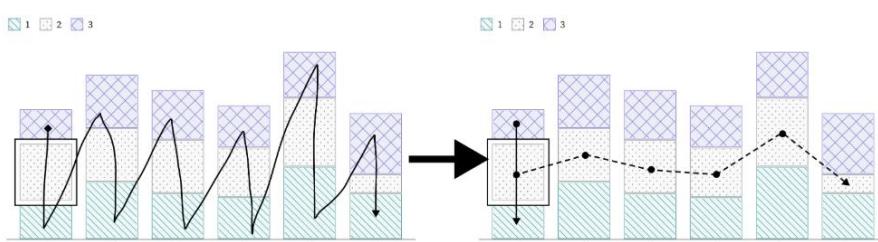
### Flexible

User style change not respected

### Assistive

Data density is inappropriate  
Navigation and interaction is tedious

Link: [Chartability](https://chartability.fizz.studio/)<sup>144</sup>



Screenshot 45 Explanation of how the keyboard should behave in a chart

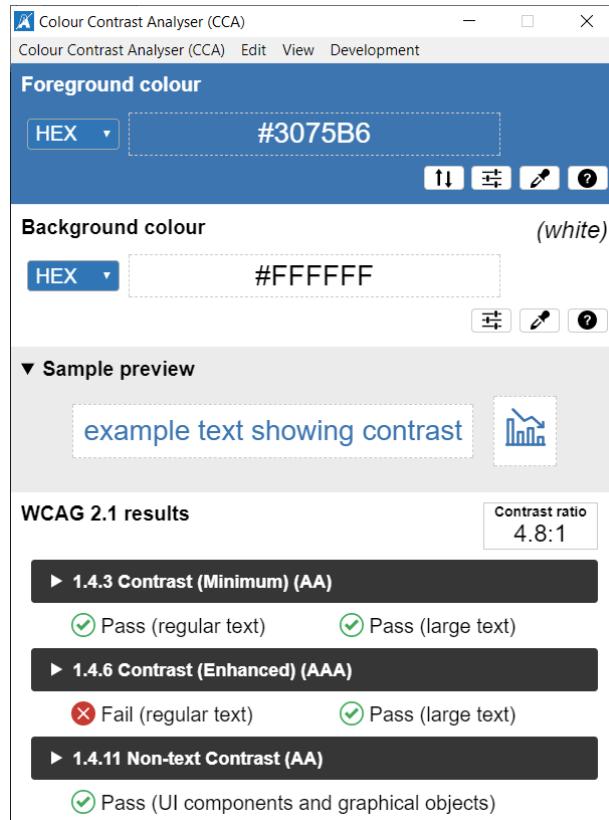
<sup>144</sup> <https://chartability.fizz.studio/>

# Criteria validation tools

## Color contrast validation

To test the contrast ratios between two colors (criteria 1.4.3, 1.4.6, and 1.4.11), the simplest tool is *The Paciello Group's Colour Contrast Analyser*. The eyedropper tool allows you to select colors anywhere on the screen and check the contrast results. If you prefer, you can directly indicate the color codes you want to analyze.

Link: [Colour Contrast Analyser](https://developer.pacielligroup.com/resources/contrastanalyser/)<sup>145</sup> for Windows and Mac.



Screenshot 46 Color Contrast Analyser

<sup>145</sup> <https://developer.pacielligroup.com/resources/contrastanalyser/>

WCAG 2 uses an algorithm for color contrast that sometimes can fail in practical applications.

For example, the following images share the same background color, blue (#3399FF). In the first one, the icon is white (#FFFFFF), while in the second, it is dark blue (#21476E). Perceptually, the first one has high contrast, while the second is more difficult to distinguish. However, the first combination of colors has a contrast ratio of 2.9:1 and the second has 3.2:1, which means that the first image does not pass the Criterion 1.4.11 (minimum 3.0:1), but the second does.



White over Blue  
(#FFFFFF - #3399FF)  
Contrast Ratio: 2.9:1



Dark blue over Blue  
(#21476E - #3399FF)  
Contrast Ratio: 3.2:1

Besides, some combinations that pass the minimum contrast, such as this star (with a contrast ratio of 3.1:1), are perceived with great difficulty by people with color blindness



Normal vision



Deuteranomaly



Protanomaly

To address these and other issues, alternative formulas are being considered. The most popular is the **Accessible Perceptual Contrast Algorithm (APCA)**<sup>146</sup>.

Although it is not officially recognized by the WCAG yet, it may be interesting to check colors with a validator that combines both the current algorithm and APCA, like *Bridge PCA*, created by Andrew Somers in his project [Myndex Perception Research](https://github.com/Myndex/PerceptionResearch).<sup>147</sup>

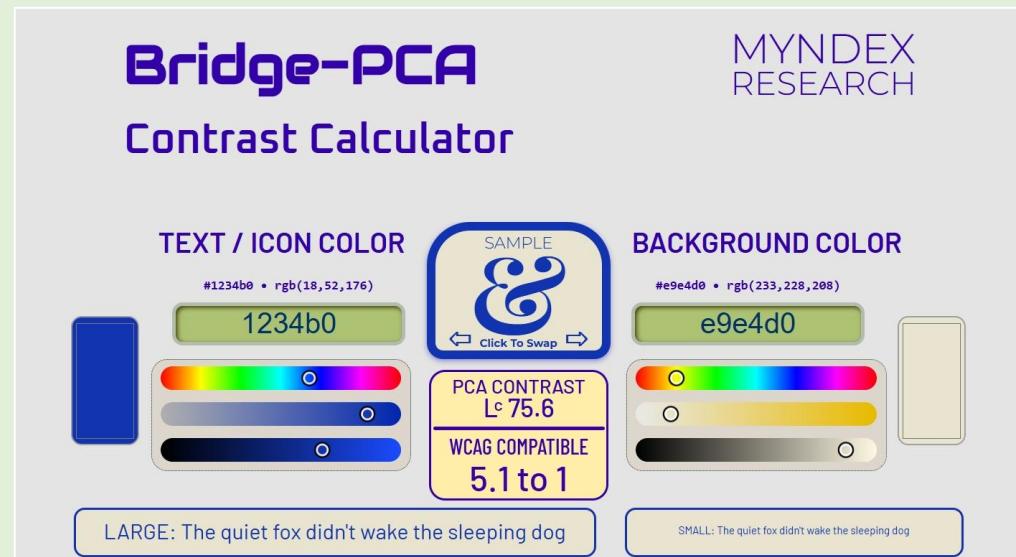
<sup>146</sup> <https://git.apcacontrast.com/documentation/APCAeasyIntro>

<sup>147</sup> <https://github.com/Myndex>

Bridge PCA tool reports contrast as the APCA lightness contrast (Lc) value and the WCAG 2 ratio. The conversion would be :

- Lc 60 exceeds 3:1 ratio in WCAG.
- Lc 75 exceeds 4.5:1 ratio in WCAG.
- Lc 90 exceeds 7:1 ratio in WCAG.

Link: [Bridge PCA](https://www.myndex.com/BPCA/)<sup>148</sup>



Screenshot 47 Bridge PCA



To go deep into the theory of color and contrast, we recommend:

- "[The Realities And Myths Of Contrast And Color](https://www.smashingmagazine.com/2022/09/realities-myths-contrast-color/)"<sup>149</sup> by the specialist is lighting Andrew Somers
- "[Why WCAG Color Contrast Isn't Always Accurate](https://www.linkedin.com/pulse/why-wcag-color-contrast-isnt-always-accurate-evan-levesque-aw7ac/)"<sup>150</sup> by the digital accessibility engineer Evan Levesque.

<sup>148</sup> <https://www.myndex.com/BPCA/>

<sup>149</sup> <https://www.smashingmagazine.com/2022/09/realities-myths-contrast-color/>

<sup>150</sup> <https://www.linkedin.com/pulse/why-wcag-color-contrast-isnt-always-accurate-evan-levesque-aw7ac/>

## Validation of sound contrast



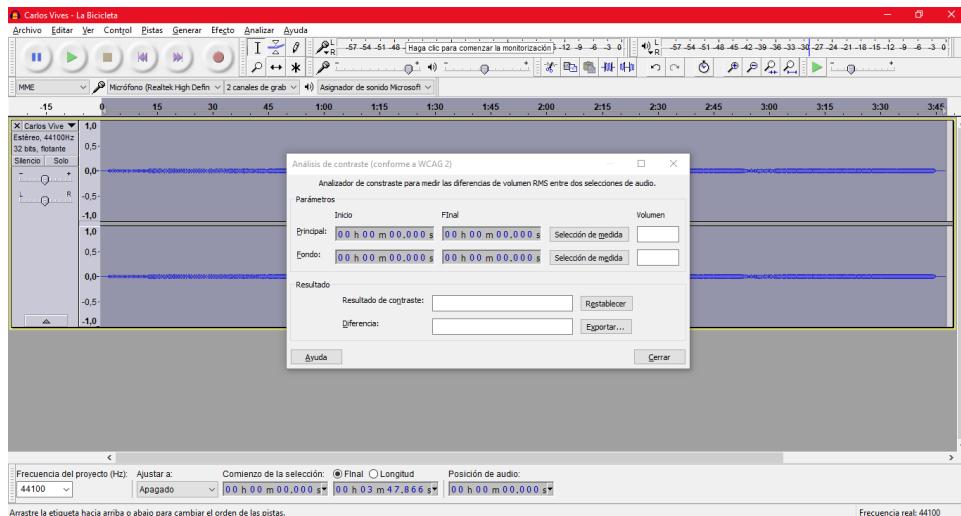
To check if the background and foreground sounds differ by 20 decibels, you can use Audacity®, a free desktop tool (available for Windows, Mac, and Linux) that includes an option to check the contrast according to criterion 1.4.7 (AAA)

Within the "Analyze" menu, you have the "Contrast" option. Select the parts of the audio track you want to check, and you'll get the result.

### Links:

[Audacity Download](https://www.audacityteam.org/)<sup>151</sup>

[Tutorial: How to Check Contrast with Audacity](https://manual.audacityteam.org/man/contrast.html)<sup>152</sup>



Screenshot 48 Sound Contrast Analysis with Audacity

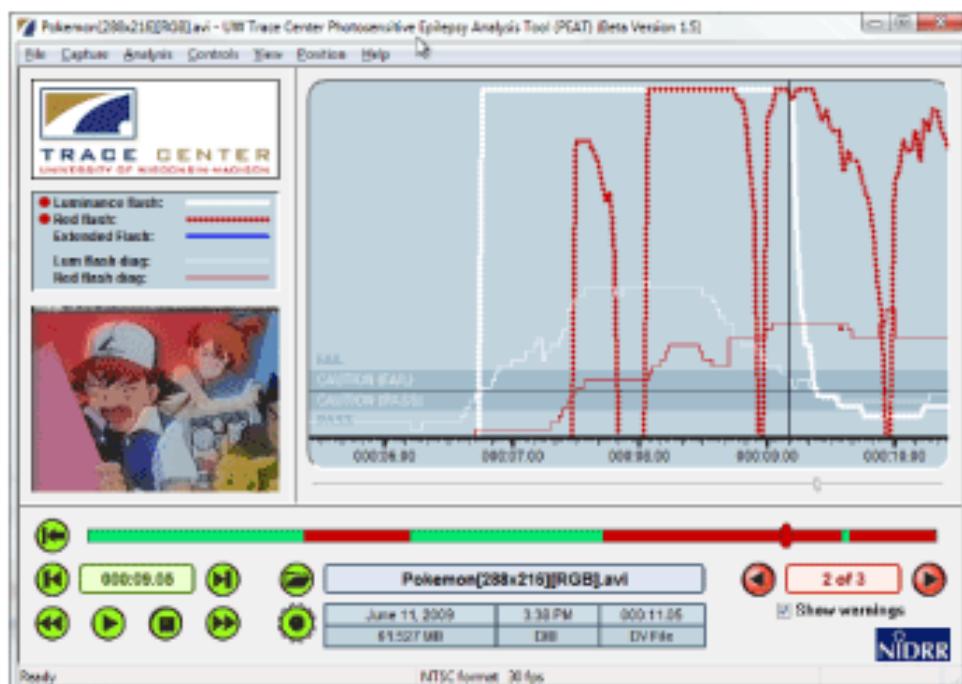
<sup>151</sup> <https://www.audacityteam.org/>

<sup>152</sup> <https://manual.audacityteam.org/man/contrast.html>

## Flash validation

In accordance with criteria 2.3.1 (A) and 2.3.2 (AAA), to test whether a video contains flashes that can cause seizures, we recommend the **PEAT - Photosensitive Epilepsy Analysis Tool**, which visually shows the conflicting moments.

Link: [PEAT](https://trace.umd.edu/peat/)<sup>153</sup>



Screenshot 49 Flash Analysis with PEAT

<sup>153</sup> <https://trace.umd.edu/peat/>

## Readability

To evaluate criterion 3.1.5 (AAA), we can use the [Flesch-Kindaid tests](#)<sup>154</sup>, that assess how challenging a piece of English text is to comprehend depending on the number of words in a sentence and the number of syllables per word. There are two types of tests: the Flesch **Reading-Ease** and the Flesch-Kincaid **Grade Level**. Both tests utilize common metrics, such as word length and sentence length, but they apply different weighting factors. For instance, a reading score between 60 and 70 corresponds to a grade level of 8 to 9, indicating that a 13 to 15-year-old should be able to comprehend a text with this score.

To calculate the Flesch-Kincaid scales and grade level of websites we can use the free online test from Juicy Studio.

Link: [Juicy Studio](#) <sup>155</sup>

### Readability Results

The following table contains the readability results for <http://www.itakora.com>.

Reading Level Results

Summary	Value
Total sentences	89
Total words	467
Average words per Sentence	5.25
Words with 1 Syllable	258
Words with 2 Syllables	85
Words with 3 Syllables	65
Words with 4 or more Syllables	59
Percentage of word with three or more syllables	26.55%
Average Syllables per Word	1.84
Gunning Fog Index	12.72
Flesch Reading Ease	45.90
Flesch-Kincaid Grade	8.16



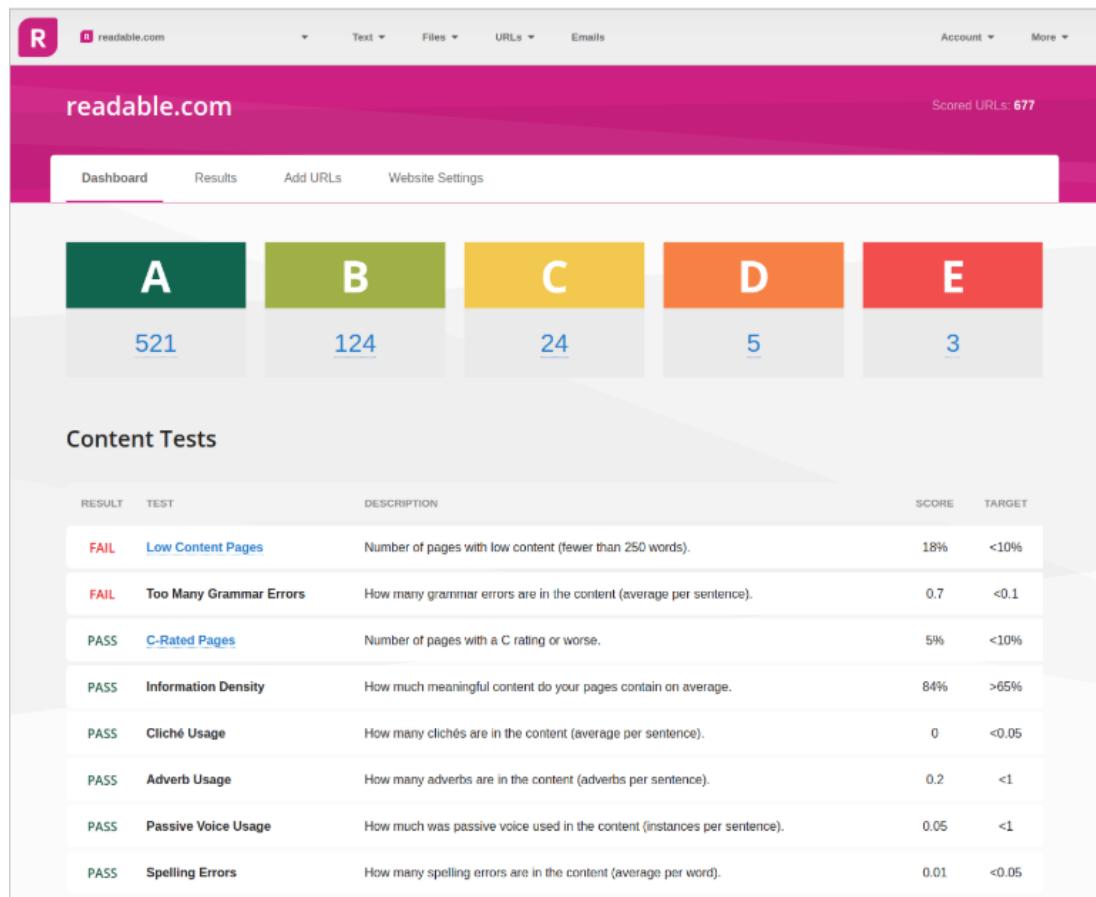
Screenshot 50 Readability Analysis with Juicy Studio

<sup>154</sup> [https://en.wikipedia.org/wiki/Flesch%20Kincaid\\_readability\\_tests](https://en.wikipedia.org/wiki/Flesch%20Kincaid_readability_tests)

<sup>155</sup> <https://juicystudio.com/services/readability.php>

Another program we can use is **Readable**, which scores files, emails, or URLs, according to different readability formulas, including Flesch-Kincaid.

Link: [Readable](https://readable.com)<sup>156</sup>



Screenshot 51 Readability Analysis with Readable

You can also try AI tools like [Grammarly](https://www.grammarly.com)<sup>157</sup>, [ChatGPT](https://chat.openai.com)<sup>158</sup>, [Copilot](https://copilot.microsoft.com)<sup>159</sup> or similar to improve your texts. Just give them precise instructions to correct the spelling and grammar of a text, or to adapt its tone, clarity, or simplicity.

<sup>156</sup> <https://readable.com>

<sup>157</sup> <https://www.grammarly.com>

<sup>158</sup> <https://chat.openai.com>

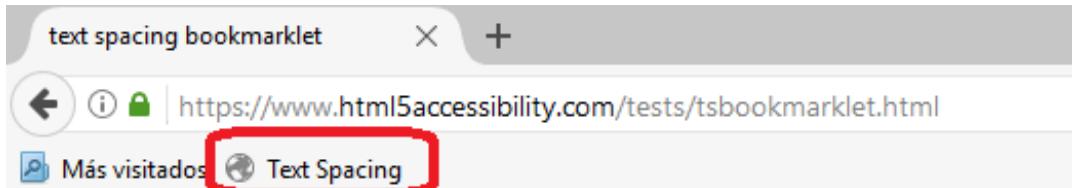
<sup>159</sup> <https://copilot.microsoft.com>

## Text Spacing Validation

The criterion "1.4.12 Text spacing" (AA) requires checking the page with specific spacing values to verify that there is no loss of content or functionality.

To do this, we can use the *bookmarklet "Text spacing"* by the accessibility specialist Steve Faulkner, which is included in the browser as a bookmark.

Link: [bookmarklet "Text spacing"](https://www.html5accessibility.com/tests/tsbookmarklet.html)<sup>160</sup>



Screenshot 52 Bookmarklet "Text spacing" by Steve Faulkner

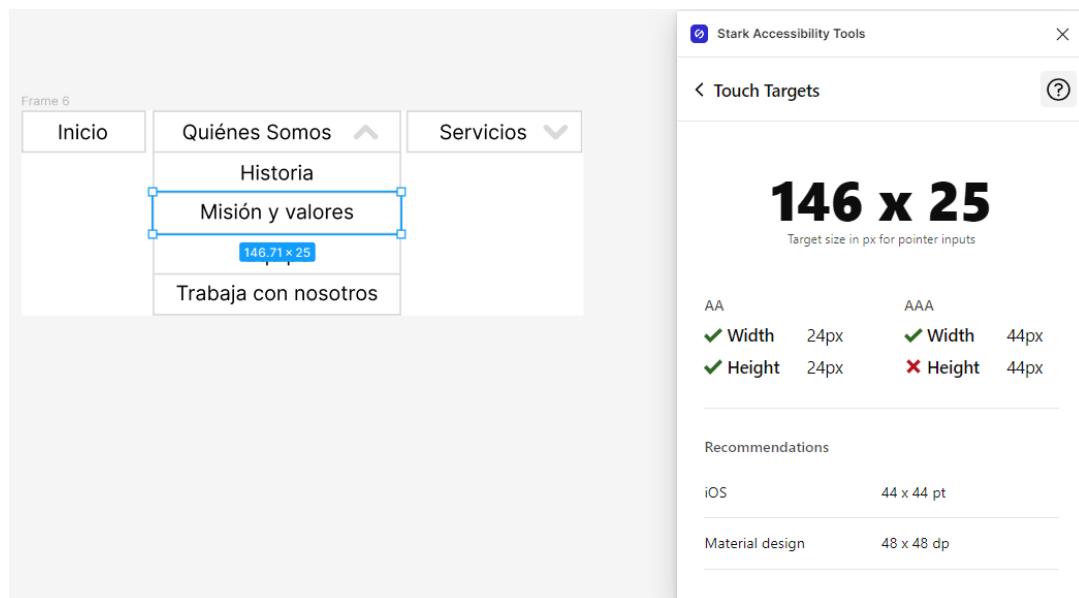
<sup>160</sup> <https://www.html5accessibility.com/tests/tsbookmarklet.html>

## Target Size Validation

Criteria 2.5.8 (AA) and 2.5.5 (AAA) indicate that interactive elements must have a minimum area of interaction to be activated without problems.

With the Figma **Stark Accessibility Tools** plugin, you can check whether or not the interactive element meets the minimum dimensions.

Link: [Figma Stark Accessibility Tools plugin](https://www.figma.com/community/plugin/732603254453395948)<sup>161</sup>



Screenshot 53 Checking Target Sizes with Stark Accessibility Tools

<sup>161</sup> <https://www.figma.com/community/plugin/732603254453395948>

## Code Validation

The **W3C Validation Service** offers two tools to check criterion 4.1.1, one to validate the HTML, XHTML, SMIL, MathML, or SVG code of your website and another to validate CSS stylesheets.

### Links:

- [W3C Validation Service](https://validator.w3.org/)<sup>162</sup>
- [W3C CSS Validator](https://jigsaw.w3.org/css-validator/)<sup>163</sup>

This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

The W3C validators are hosted on server technology donated by HP, and supported by community donations. [Donate](#) and help us build better tools for a better web.

Screenshot 54 W3C Markup Validation Service

Nota: Si deseas validar tu hoja de estilo CSS incluida en un documento (X)HTML, deberás antes comprobar que el (X)HTML utilizado es válido.

Acerca de este servicio Documentación Download Comentarios Créditos

COPYRIGHT © 1994-2009 W3C® (MIT, ERCIM, KEIO). ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

Screenshot 55 W3C CSS Validation Service

<sup>162</sup> <https://validator.w3.org/>

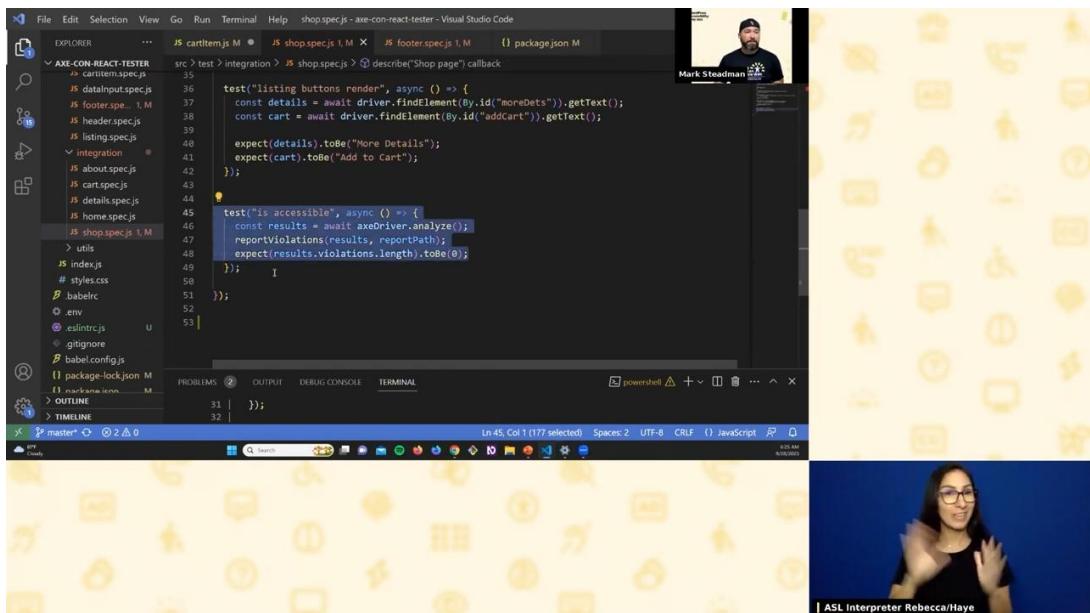
<sup>163</sup> <https://jigsaw.w3.org/css-validator/>

# Automated accessibility testing

During the product development cycle, a testing phase is conducted where software specialists verify for errors or bugs. This testing includes interoperability, security, privacy, and, of course, accessibility issues.

To perform this testing, testers utilize both automatic and manual validation tools, as mentioned earlier, but also employ specialized tools to detect and resolve as many issues as possible.

According to the web developer Mark Steadman, in his talk “[A Practical Guide To Automated Accessibility](#)<sup>164</sup>”, there are 3 types of automated testing tools: Linters, testing libraries and regression testing.



Screenshot 56. Video of the talk “A Practical Guide To Automated Accessibility” by Mark Steadman on YouTube, showcasing different tools.

## Linting

Linters are developer tools that analyze source code for errors, vulnerabilities, and stylistic issues. This detects the problem as early as possible during code writing. They check statically, tag by tag.

Some popular linters are:

- [ESLint](#)<sup>165</sup>

<sup>164</sup> <https://www.youtube.com/watch?v=UB2ljZ6hcI4>

<sup>165</sup> <https://eslint.org>

- [Eslint JSX A11y](#)<sup>166</sup>
- [AXE-Linter](#)<sup>167</sup>

## Automated Testing libraries

---

Automated testing libraries have built-in checks that scan UI content for accessibility issues with a consistent set of rules. They check fully rendered components, or pages, that is, they can be used for unit testing or integration testing.

Some popular libraries are:

- [Axe-core](#)<sup>168</sup>
- [PA11y](#)<sup>169</sup>

## Automated Regression Testing

---

With automated tests cases developers ensure the accessible functionality of the components, for example, if a component is expanded or collapsed when it has the *aria-expanded* property on *true* or *false*.

Some popular tools are:

- [Cypress](#)<sup>170</sup>
- [Selenium](#)<sup>171</sup>

---

<sup>166</sup> <https://www.npmjs.com/package/eslint-plugin-jsx-a11y>

<sup>167</sup> <https://www.deque.com/axe/devtools/linter/>

<sup>168</sup> <https://github.com/dequelabs/axe-core>

<sup>169</sup> <https://pa11y.org>

<sup>170</sup> <https://www.cypress.io>

<sup>171</sup> <https://www.selenium.dev>

# **Work tools**

**Aside from evaluation and validation, there are other tools you can use to ensure the accessibility of your projects at different points in the product cycle.**

**We show you the main screen readers, audio and video players, and disability simulators, so you can experience first-hand how people with disabilities handle themselves on the web.**

**Finally, designers must transfer their creations to developers and other team members, and handover documentation is a critical tool for verifying and communicating the accessibility of created components.**



# Screen readers

Desktop screen readers are operated with keyboard shortcuts, and mobile screen readers are operated with touch gestures.

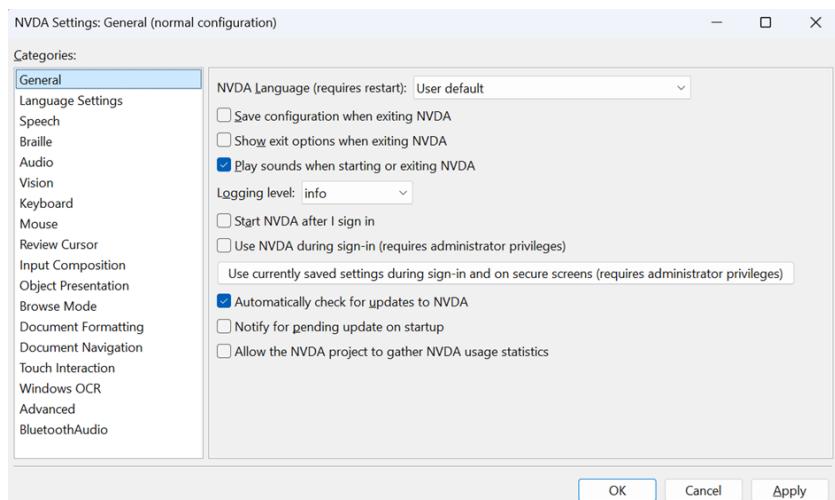
The options are similar among the different screen readers, but each one has its peculiarities, and they do not always state the content in the same way. It is, therefore, important to conform as closely as possible to the standards and norms so that they can interpret the pages and applications as faithfully as possible.

## Desktop screen readers

Many screen readers are out there, but our recommendation for Windows is **NVDA**, which is free and offers many configuration options.

### Links:

- [NVDA](#)<sup>172</sup>
- [NVDA User Guide](#)<sup>173</sup>
- [Top NVDA Keyboard Shortcuts](#)<sup>174</sup>



Screenshot 57 NVDA Desktop Options

Windows includes a default browser in the system: **Narrator**, which can be activated by pressing the Windows + Control + Enter keys.

The default screen reader for macOS is **VoiceOver**, which you can activate with Command + F5.

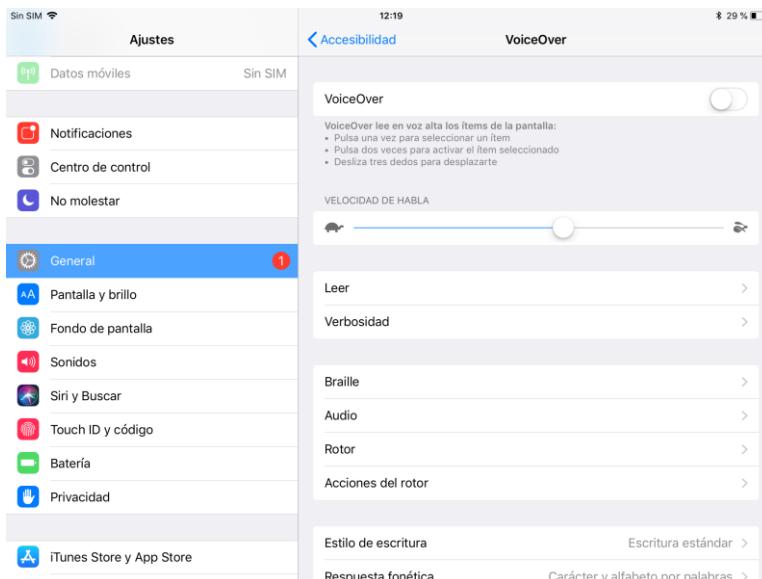
<sup>172</sup> <https://www.nvaccess.org>

<sup>173</sup> <https://www.nvaccess.org/files/nvda/documentation/userGuide.html>

<sup>174</sup> <https://dequeuniversity.com/screenreaders/nvda-keyboard-shortcuts>

## Mobile Device Readers

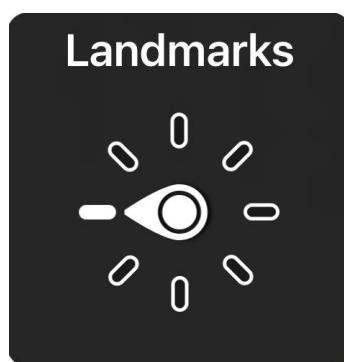
People who use the iOS (Apple) operating system have the **VoiceOver** screen reader built into the system.



Screenshot 58 VoiceOver on iOS for tablet

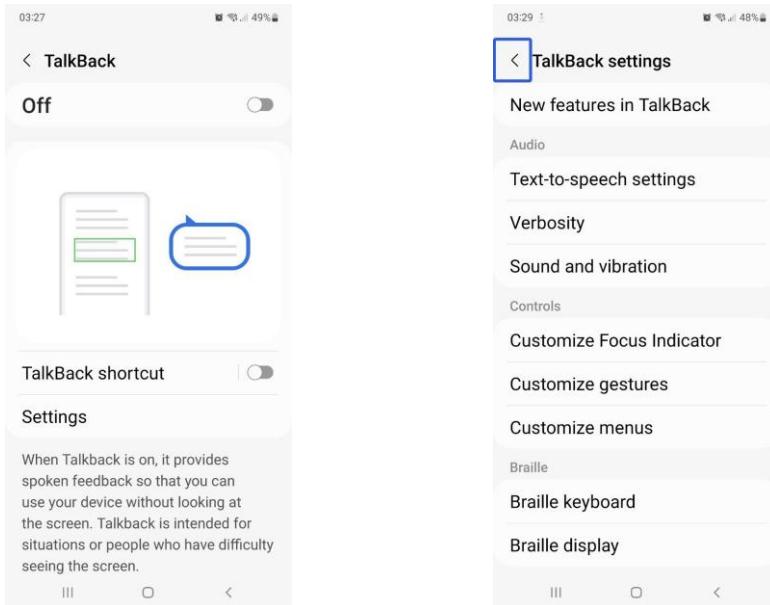
In addition to the simplest gestures, such as swiping right or left to cycle through the screen elements, or two taps to select the element with the focus, the rotor gesture is one of the most useful ones. To activate it, you have to turn two fingers on the screen of the iOS or iPadOS device as if you were turning a dial.

It allows you to select different options, for example, moving around a web page through its headers or its regions:



Screenshot 59 VoiceOver on iOS with the rotor on

On the Android operating system, you can activate **TalkBack** within Settings.



Screenshot 60 Talkback functionality and settings on an Android phone

In addition to the simplest gestures, such as swiping right or left to cycle through screen elements, or two taps to select the element that has the focus, with Talkback you can pull up a context menu by swiping one finger down and right in a continuous motion.

If you're on a web page, you can try swiping up and down quickly and then down to select how to navigate the page (e.g., through its headers):

Android Headings

Mar 13, 2023

Let's have a look at the headings in this post as an ordered list. By listing them out it helps me create a logical flow to the post. Also, it gives the reader an understanding of the progression. And by creating links it's far easier to navigate.

1. Introduction  
2. What is a heading?  
3. Misconceptions  
1. Design

Headings

2. Development  
3. Why does it matter

Screenshot 61 Talkback menu on a web page.

# Accessible audio and video players

HTML5 allows you to include audio and video natively with the `<audio>` or `<video>` elements. Although they are well supported by all current browsers, the same is not true of its `<track>` element, which should allow you to associate subtitles, chapters, or transcripts with them. On the other hand, not all native players are fully accessible with the keyboard and assistive products. In addition, they do not support audio description or synchronization in sign language.

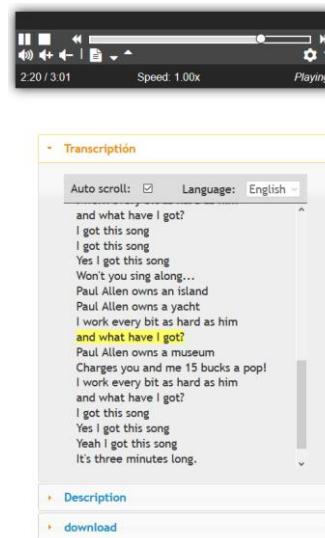
For this reason, it is highly recommended to use other HTML5 players based on the elements `<audio>` or `<video>`, which are fully accessible and support all the necessary alternatives to meet the criteria of Guideline 1.2. Our recommendations are **Able Player** and **OzPlayer**. Both support YouTube videos, and it is more advisable to use these players for their expanded accessibility support than the YouTube player itself. You can see a comparison table of different players and their features in [HTML5 Video Player Comparison](#)<sup>175</sup>

## Links:

- [Able Player](#)<sup>176</sup>
- [OzPlayer](#)<sup>177</sup>



Screenshot 62 Video with transcription in OZ Player



Screenshot 63 Audio with transcription in Able Player

<sup>175</sup> <http://videosws.praegnanz.de/#sws>

<sup>176</sup> <https://ableplayer.github.io/ableplayer/>

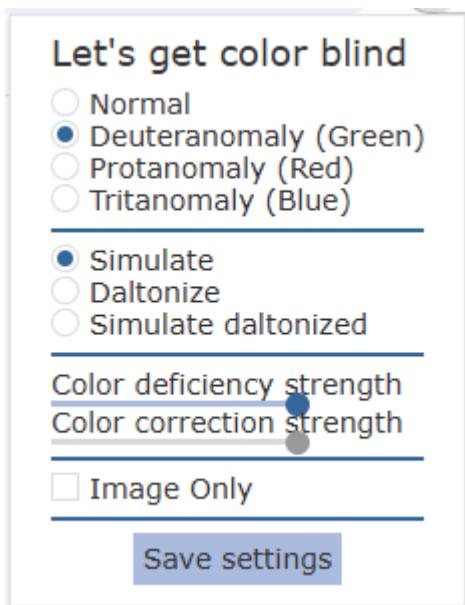
<sup>177</sup> <https://www.accessibilityoz.com/ozplayer>

# Simulating Visually Impaired Access

In the Chrome browser and in the Figma or Sketch programs we can use extensions that modify web pages in such a way that we can put ourselves in the shoes of people with color blindness, low vision and blindness and check how they perceive, navigate, and interact with the web.

## Links:

- ["Let's get color blind" extension<sup>178</sup>](#) for Chrome.
- [Stark](#) for Figma and Sketch<sup>179</sup>.



Screenshot 64 "Let's get color blind"  
Chrome extension



Screenshot 65 Stark Extension for  
Figma and Sketch

<sup>178</sup> <https://chromewebstore.google.com/detail/lets-get-color-blind/bkdgidianpkfahpkmphgehigalpighjck>

<sup>179</sup> <https://www.getstark.co/support/getting-started/using-the-vision-simulator>

# Design-to-development handover documentation

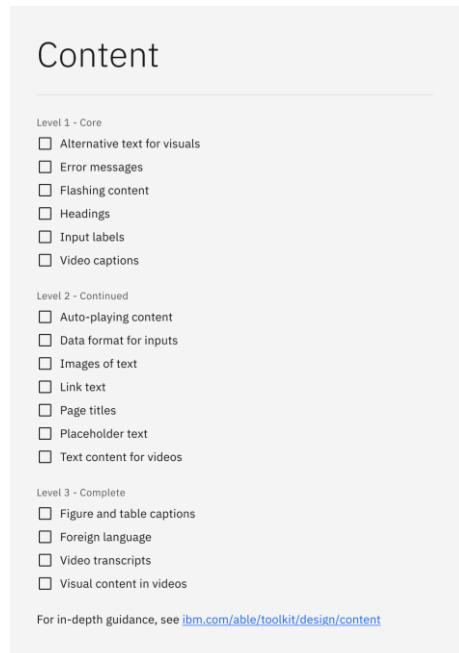
An essential moment in the dynamics of a digital team occurs when UX/UI designers handoff their work to developers. To execute this handoff, UX/UI designers develop certain documentation, including accessibility features, that help developers to understand the designs.

IBM has developed a kit that facilitates this documentation, with a checklist and guides so that you don't forget to document headings, form labels, header order, tab order, keyboard interaction....

The kit is available in both Figma and Sketch.

## Links:

- [Figma: IBM Accessibility Design Kit](#)<sup>180</sup>
- [Sketch: IBM Accessibility Design Kit](#)<sup>181</sup>



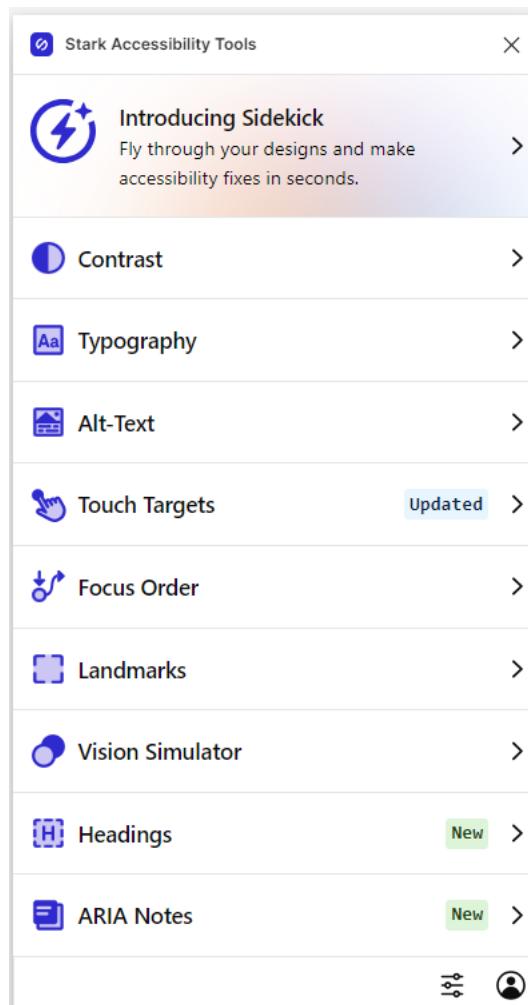
Screenshot 66 IBM Accessibility Kit

<sup>180</sup> <https://www.figma.com/community/file/1118184491812988116>

<sup>181</sup> <https://www.sketch.com/s/f0a04c0d-fb62-4d71-92c6-07c402f8cae7>

In addition, other *plugins*, such as the aforementioned **Stark Accessibility Tools**, can be used to integrate additional comments into the design which allow developers to implement designs in an accessible way, such as the indication of the order of focus, headings, alternative texts of images, ARIA annotations...

Link: [Stark Accessibility](#)<sup>182</sup>



Screenshot 67 Stark Accessibility

<sup>182</sup> <https://www.figma.com/community/plugin/732603254453395948>



# **Summaries and Diagrams**

This section presents some tables and charts that can help you better understand WCAG 2.2.

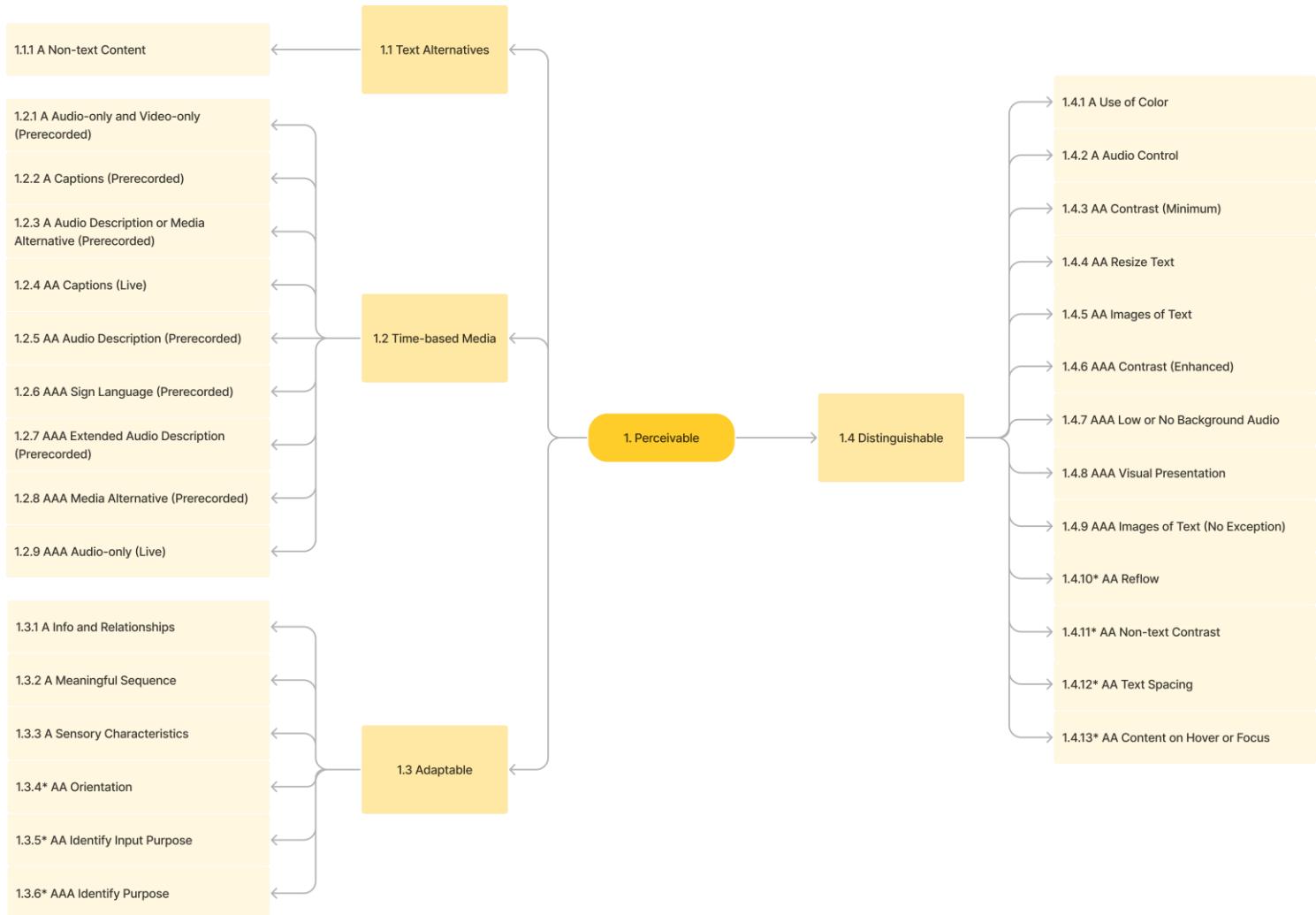
First, we show the conceptual diagrams of the guidelines.

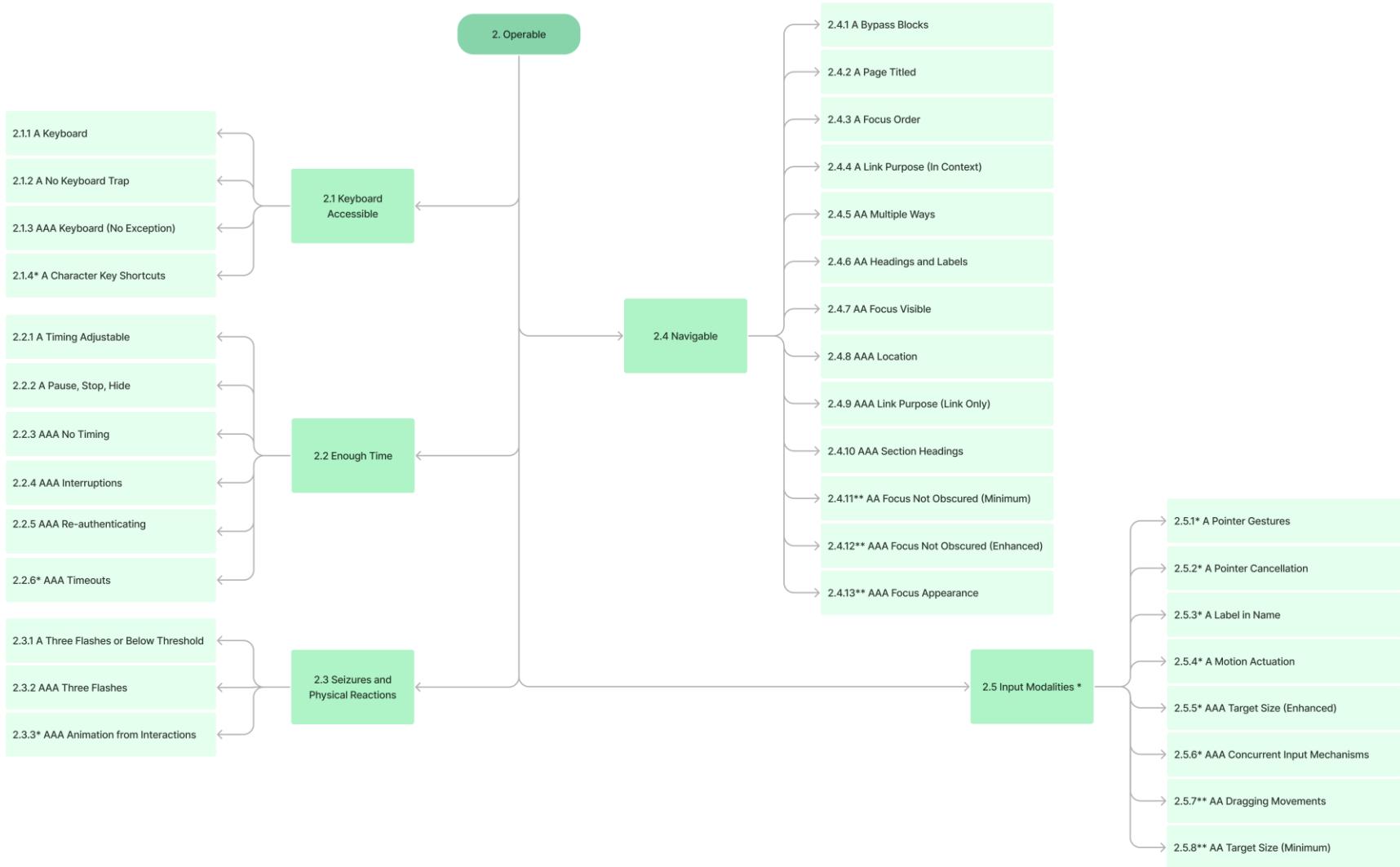
Next, we list the guidelines and criteria grouped in diverse ways so you can use them as a reference according to your needs: by principles, levels, or professions.

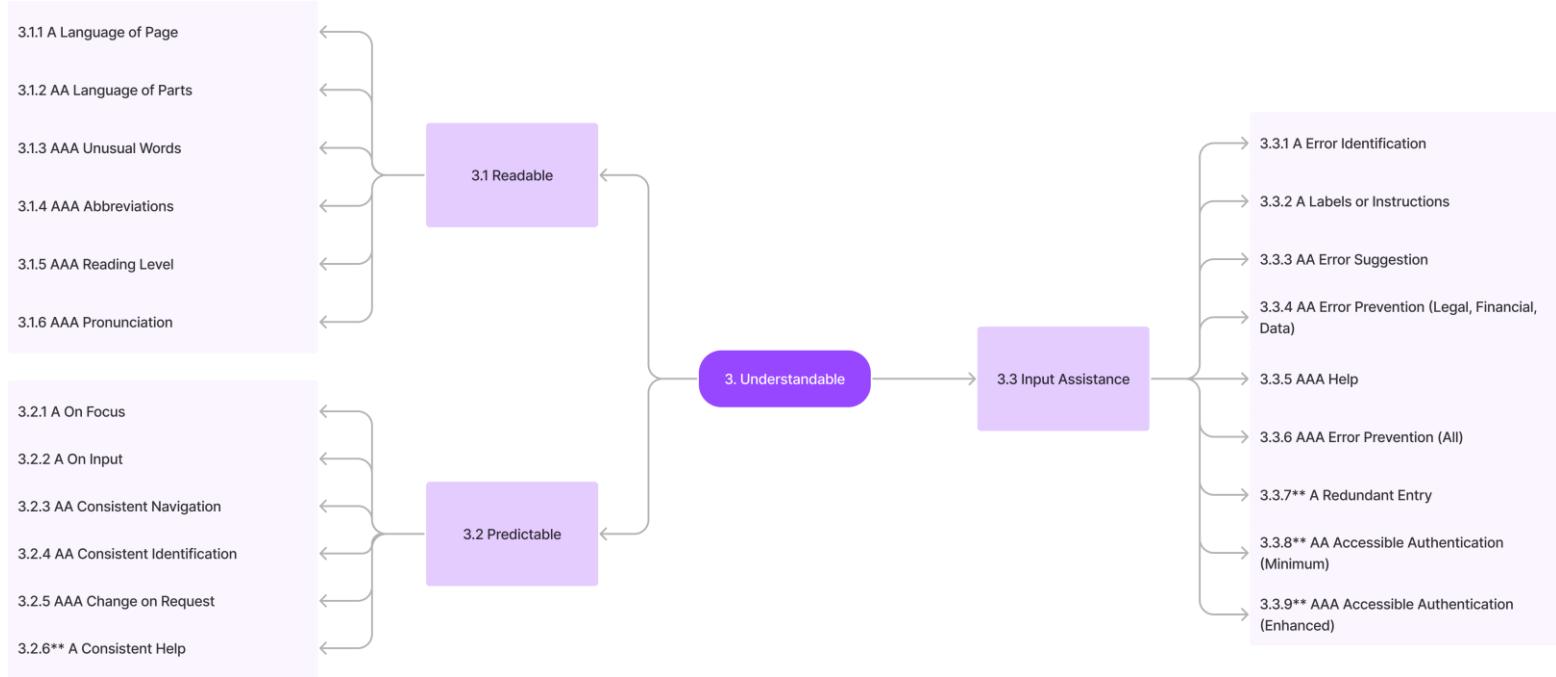
## WCAG 2.2 Concept Diagrams

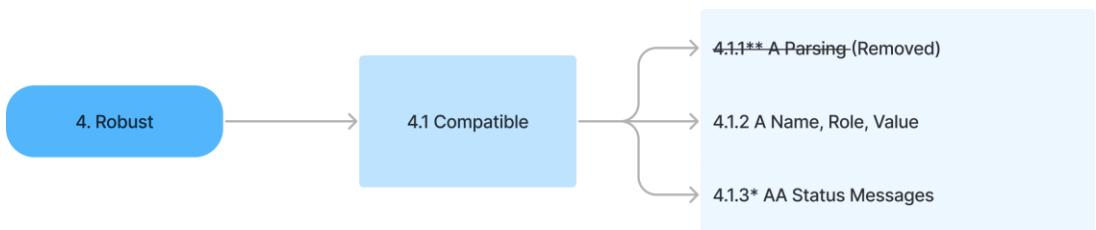
\* New WCAG 2.1 \*\* New WCAG 2.2











# Principles, Guidelines, and Conformity Criteria

\* New WCAG 2.1 \*\* New WCAG 2.2

## Principle 1. Perceivable

---

### Guideline 1.1 Text Alternatives

Criterion	Level	Description
1.1.1	A	Non-Text Content

### Guideline 1.2 Time-based media

Criterion	Level	Description
1.2.1	A	Audio-only and Video-only (Prerecorded)
1.2.2	A	Captions (Prerecorded)
1.2.3	A	Audio Description or Media Alternative (Prerecorded)
1.2.4	AA	Captions (Live)
1.2.5	AA	Audio Description (Prerecorded)
1.2.6	AAA	Sign Language (Prerecorded)
1.2.7	AAA	Extended Audio Description (Prerecorded)
1.2.8	AAA	Media Alternative (Prerecorded)
1.2.9	AAA	Audio-only (Live)

### Guideline 1.3 Adaptable

Criterion	Level	Description
1.3.1	A	Information and Relationships
1.3.2	A	Meaningful Sequence
1.3.3	A	Sensory Characteristics
1.3.4*	AA	Orientation
1.3.5*	AA	Identify Input Purpose
1.3.6*	AAA	Identify the Purpose

## **Guideline 1.4 Distinguishable**

<b>Criterion</b>	<b>Level</b>	<b>Description</b>
1.4.1	A	Use of Color
1.4.2	A	Audio Control
1.4.3	AA	Contrast (Minimum)
1.4.4	AA	Resize Text
1.4.5	AA	Images of Text
1.4.6	AAA	Contrast (Enhanced)
1.4.7	AAA	Low or No Background Audio
1.4.8	AAA	Visual Presentation
1.4.9	AAA	Images of Text (No Exceptions)
1.4.10*	AA	Reflow
1.4.11*	AA	Non-Text Contrast
1.4.12*	AA	Text Spacing
1.4.13*	AA	Content on Hover or Focus

## Principle 2. Operable

---

### Guideline 2.1 Keyboard Operable

Criterion	Level	Description
2.1.1	A	Keyboard
2.1.2	A	No Keyboard Trap
2.1.3	AAA	Keyboard (No Exceptions)
2.1.4*	A	Character Key Shortcuts

### Guideline 2.2 Enough Time

Criterion	Level	Description
2.2.1	A	Timing Adjustable
2.2.2	A	Pause, Stop, Hide
2.2.3	AAA	No Timing
2.2.4	AAA	Interruptions
2.2.5	AAA	Re-Authenticating
2.2.6*	AAA	Timeouts

### Guideline 2.3 Negative physical and psychic reactions

Criterion	Level	Description
2.3.1	A	Three Flashes or Below Threshold
2.3.2	AAA	Three Flashes
2.3.3*	AAA	Animation from Interactions

### Guideline 2.4 Navigable

Criterion	Level	Description
2.4.1	A	Bypass Blocks
2.4.2	A	Page Titled
2.4.3	A	Focus Order
2.4.4	A	Link Purpose (In Context)
2.4.5	AA	Multiple Ways
2.4.6	AA	Headings and Labels
2.4.7	AA	Focus Visible
2.4.8	AAA	Location
2.4.9	AAA	Link Purpose (Link Only)
2.4.10	AAA	Section Headings
2.4.11**	AA	Focus Not Obscured (Minimum)
2.4.12**	AAA	Focus Not Obscured (Enhanced)
2.4.13**	AAA	Focus Appearance

## **Guideline 2.5 Input modalities \***

<b>Criterion</b>	<b>Level</b>	<b>Description</b>
2.5.1*	A	Pointer Gestures
2.5.2*	A	Pointer Cancellation
2.5.3*	A	Label in Name
2.5.4*	A	Motion Actuation
2.5.5*	AAA	Target Size (Enhanced)
2.5.6*	AAA	Concurrent Input Mechanisms
2.5.7**	AA	Dragging Movements
2.5.8**	AA	Target Size (Minimum)

## Principle 3. Understandable

---

### Guideline 3.1 Easy to read

Criterion	Level	Description
3.1.1	A	Language of Page
3.1.2	AA	Language of Parts
3.1.3	AAA	Unusual Words
3.1.4	AAA	Abbreviations
3.1.5	AAA	Reading Level
3.1.6	AAA	Pronunciation

### Guideline 3.2 Predictable

Criterion	Level	Description
3.2.1	A	On Focus
3.2.2	A	On Input
3.2.3	AA	Consistent Navigation
3.2.4	AA	Consistent Identification
3.2.5	AAA	Change on Request
3.2.6**	A	Consistent Help

### Guideline 3.3 Help

Criterion	Level	Description
3.3.1	A	Error Identification
3.3.2	A	Labels or Instructions
3.3.3	AA	Error Suggestion
3.3.4	AA	Error Prevention (Legal, Financial, Data)
3.3.5	AAA	Help
3.3.6	AAA	Error Prevention (All)
3.3.7**	A	Redundant Entry
3.3.8**	AA	Accessible Authentication (Minimum)
3.3.9**	AAA	Accessible Authentication (Enhanced)

## Principle 4. Robust

---

### Guideline 4.1 Compatible

Criterion	Level	Description
4.1.1**	A	<del>Parsing</del> [REMOVED]
4.1.2	A	Name, Role, Value
4.1.3*	AA	Status Messages

# Tiered Conformity Criteria

\* New WCAG 2.1 \*\* New WCAG 2.2

## Level A

<b>1.1.1</b>	Non-Text Content
<b>1.2.1</b>	Audio-Only and Video-Only (Prerecorded)
<b>1.2.2</b>	Subtitles (Recorded)
<b>1.2.3</b>	Audio Description or Media Alternative (Prerecorded)
<b>1.3.1</b>	Information and Relationships
<b>1.3.2</b>	Meaningful Sequence
<b>1.3.3</b>	Sensory Characteristics
<b>1.4.1</b>	Use of Color
<b>1.4.2</b>	Audio Control
<b>2.1.1</b>	Keyboard
<b>2.1.2</b>	No Keyboard Trap
<b>2.1.4*</b>	Character Key Shortcuts
<b>2.2.1</b>	Timing Adjustable
<b>2.2.2</b>	Pause, Stop, Hide
<b>2.3.1</b>	Three Flashes or Below Threshold
<b>2.4.1</b>	Bypass Blocks
<b>2.4.2</b>	Page Titled
<b>2.4.3</b>	Focus Order
<b>2.4.4</b>	Link Purpose (In Context)
<b>2.5.1*</b>	Pointer Gestures
<b>2.5.2*</b>	Pointer Cancellation
<b>2.5.3*</b>	Label in Name
<b>2.5.4*</b>	Motion Actuation
<b>3.1.1</b>	Language of Page
<b>3.2.1</b>	On Focus
<b>3.2.2</b>	On Input
<b>3.2.6*</b>	Consistent Help
<b>3.3.1</b>	Error Identification
<b>3.3.2</b>	Labels or Instructions
<b>3.3.7*</b>	Redundant Entry
<b>4.1.1*</b>	<del>Parsing</del> [REMOVED]
<b>4.1.2</b>	Name, Role, Value

## Level AA

---

1.2.4	Captions (Live)
1.2.5	Audio Description (Prerecorded)
1.3.4*	Orientation
1.3.5*	Identify Input Purpose
1.4.3	Contrast (Minimum)
1.4.4	Resize Text
1.4.5	Images of Text
1.4.10*	Reflow
1.4.11*	Non-Text Contrast
1.4.12*	Text Spacing
1.4.13*	Content on Hover or Focus
2.4.5	Multiple Ways
2.4.6	Headings and Labels
2.4.7	Focus Visible
2.4.11*	Focus Not Obscured (Minimum)
2.5.7**	Dragging Movements
2.5.8**	Target Size (Minimum)
3.1.2	Language of Parts
3.2.3	Consistent Navigation
3.2.4	Consistent Identification
3.3.3	Error Suggestion
3.3.4	Error Prevention (Legal, Financial, Data)
3.3.8**	Accessible Authentication (Minimum)
4.1.3*	Status Messages

## Level AAA

---

<b>1.2.6</b>	Sign Language (Prerecorded)
<b>1.2.7</b>	Extended Audio Description (Recorded)
<b>1.2.8</b>	Media Alternative (Prerecorded)
<b>1.2.9</b>	Audio-Only (Live)
<b>1.3.6*</b>	Identify the Purpose
<b>1.4.6</b>	Contrast (Enhanced)
<b>1.4.7</b>	Low or No Background Audio
<b>1.4.8</b>	Visual Presentation
<b>1.4.9</b>	Images of Text (No Exceptions)
<b>2.1.3</b>	Keyboard (No Exceptions)
<b>2.2.3</b>	No Timing
<b>2.2.4</b>	Interruptions
<b>2.2.5</b>	Re-authenticating
<b>2.2.6*</b>	Timeouts
<b>2.3.2</b>	Three Flashes
<b>2.3.3*</b>	Animation from Interactions
<b>2.4.8</b>	Location
<b>2.4.9</b>	Link Purpose (Link Only)
<b>2.4.10</b>	Section Headings
<b>2.4.12*</b>	Focus Not Obscured (Enhanced)
<b>2.4.13*</b>	Focus Appearance
<b>2.5.5*</b>	Target Size (Enhanced)
<b>2.5.6*</b>	Concurrent Input Mechanisms
<b>3.1.3</b>	Unusual Words
<b>3.1.4</b>	Abbreviations
<b>3.1.5</b>	Reading level
<b>3.1.6</b>	Pronunciation
<b>3.2.5</b>	Change on Request
<b>3.3.5</b>	Help
<b>3.3.6</b>	Error Prevention (All)
<b>3.3.9**</b>	Accessible Authentication (Enhanced)

# Criteria for content creators

\* New WCAG 2.1 \*\* New WCAG 2.2

Concept	Criterion	Level	Description
Images, videos and audios	1.1.1	A	Non-Text Content
	1.2.1	A	Audio-Only and Video-Only (Prerecorded)
	1.2.2	A	Captions (Prerecorded)
	1.2.3	A	Audio Description or Media Alternative (Prerecorded)
	1.2.4	AA	Captions (Live)
	1.2.5	AA	Audio Description (Prerecorded)
	1.2.6	AAA	Sign Language (Prerecorded)
	1.2.7	AAA	Extended Audio Description (Prerecorded)
	1.2.8	AAA	Media Alternative (Prerecorded)
	1.2.9	AAA	Audio-Only (Live)
Simple wording	1.4.2	A	Audio Control
	1.3.3	A	Sensory Characteristics
	2.4.2	A	Page Titled
	2.4.4	A	Link Purpose (In Context)
	2.4.9	AAA	Link Purpose (Link Only)
	2.4.6	AA	Headings and Labels
	2.4.10	AAA	Section Headings
	3.2.4	AA	Consistent Identification
	3.3.2	A	Labels or Instructions
	3.3.5	AAA	Help
Plain language	3.1.3	AAA	Unusual Words
	3.1.4	AAA	Abbreviations
	3.1.5	AAA	Reading Level
Easy formatting	1.3.1	A	Information and Relationships
	1.4.4	AA	Resize Text
	1.4.5	AA	Images of Text
	1.4.9	AAA	Images of Text (No Exceptions)
	1.4.10*	AA	Reflow
	2.4.1	A	Bypass Blocks
Language	3.1.1	A	Language of Page
	3.1.2	AA	Language of Parts
	3.1.6	AAA	Pronunciation
Auditory contrast	1.4.7	AAA	Low or No Background Audio
Ensure contrast	1.4.3	AA	Contrast (Minimum)
	1.4.6	AAA	Contrast (Enhanced)
	1.4.11*	AA	Non-Text Contrast

Concept	Criterion	Level	Description
<b>Don't just use color to convey information</b>	1.4.1	A	Use of Color
<b>Physical discomfort</b>	2.3.1	A	Three Flashes or Below Threshold
	2.3.2	AAA	Three Flashes
	2.3.3*	AAA	Animation from Interactions



If the content is in other formats, such as Word or PDF, for example, you should also integrate the review of the design criteria.

# Criteria for UX/UI Designers

\* New WCAG 2.1 \*\* New WCAG 2.2

Concept	Criterion	Level	Description
Ensure contrast	1.4.3	AA	Contrast (Minimum)
	1.4.6	AAA	Contrast (Enhanced)
	1.4.11*	AA	Non-Text Contrast
	2.4.13**	AAA	Focus Appearance
Don't just use color to convey information	1.4.1	A	Use of Color
Easy-to-identify interactive elements	2.4.7	AA	Focus Visible
	3.2.4	AA	Consistent Identification
Clear navigation	2.4.5	AA	Multiple Ways
	2.4.8	AAA	Location
	2.4.4	A	Link Purpose (In Context)
	2.4.9	AAA	Link Purpose (Link Only)
	3.2.3	AA	Consistent Navigation
	3.2.6**	A	Consistent Help
	3.3.5	AAA	Help
	2.4.6	AA	Headings and Labels
	3.3.2	A	Labels or Instructions
Clear forms	3.3.4	AA	Error Prevention (Legal, Financial, Data)
	3.3.6	AAA	Error Prevention (All)
	3.3.7**	A	Redundant Entry
	3.3.8**	AA	Accessible Authentication (minimum)
	3.3.9**	AAA	Accessible Authentication (Enhanced)
	3.3.1	A	Error Identification
	3.3.3	AA	Error Suggestion
Clear content organization	1.4.8	AAA	Visual Presentation
	2.4.6	AA	Headings and Labels
	2.4.10	AAA	Section Headings
Design for different sizes	1.3.4*	AA	Orientation
	1.4.4	AA	Resize Text
	1.4.10*	AA	Reflow
	1.4.12*	AA	Text Spacing
	2.5.5*	AAA	Target Size (Enhanced)
	2.5.8**	AA	Target Size (Minimum)
Images, videos and audios	1.1.1	A	Non-Text Content
	1.4.2	A	Audio Control
	2.2.2	A	Pause, Stop, Hide

<b>Concept</b>	<b>Criterion</b>	<b>Level</b>	<b>Description</b>
<b>Easy handling</b>	1.4.13*	AA	Content on Hover or Focus
	2.5.1*	A	Pointer Gestures
	2.5.2*	A	Pointer Cancellation
	2.5.4*	A	Motion Actuation
	2.5.6*	AAA	Concurrent input mechanisms
	2.5.7**	AA	Dragging Movements
<b>No time constraints</b>	2.2.1	A	Timing Adjustable
	2.2.3	AAA	No Timing
	2.2.6*	AAA	Timeouts
<b>Do not disturb</b>	2.2.4	AAA	Interruptions
	2.2.5	AAA	Re-Authenticating
<b>Don't make decisions</b>	3.2.1	A	On Focus
	3.2.2	A	On Input
	3.2.5	AAA	Change on Request
<b>Physical discomfort</b>	2.3.1	A	Three Flashes or Below Threshold
	2.3.2	AAA	Three Flashes
	2.3.3*	AAA	Animation from Interactions

# Criteria for developers

\* New WCAG 2.2 \*\* New WCAG 2.1

Concept	Criterion	Level	Description
<b>Valid code</b>	4.1.1**	A	Parsing [REMOVED]
<b>Structure information semantically</b>	1.3.1	A	Information and Relationships
	1.3.2	A	Meaningful Sequence
	1.4.13*	AA	Content on Hover or Focus
	3.3.2	A	Labels or Instructions
	4.1.3*	AA	Status Messages
<b>Identify controls and their purpose</b>	1.3.5*	AA	Identify Input Purpose
	1.3.6*	AAA	Identify the Purpose
	4.1.2	A	Name, Role, Value
	2.4.4	A	Link Purpose (In Context)
	2.4.9	AAA	Link Purpose (Link Only)
	2.5.3*	A	Label in Name
<b>Separate content from presentation</b>	1.1.1	A	Non-Text Content
	1.4.5	AA	Images of Text
	1.4.9	AAA	Images of Text (No Exceptions)
<b>Use via keyboard</b>	2.1.1	A	Keyboard
	2.1.2	A	No Keyboard Trap
	2.1.3	AAA	Keyboard (No Exceptions)
	2.1.4*	A	Character Key Shortcuts
	2.4.3	A	Focus Order
	2.4.7	AA	Focus Visible
	2.4.11**	AA	Focus Not Obscured (Minimum)
	2.4.12**	AAA	Focus Not Obscured (Enhanced)
<b>Make it easy to navigate</b>	2.4.1	A	Bypass Blocks
	2.4.2	A	Page Titled
	2.4.8	AAA	Location
<b>Language</b>	3.1.1	A	Language of Page
	3.1.2	AA	Language of Parts
<b>Help fill out forms</b>	3.3.1	A	Error Identification
	3.3.2	A	Labels or Instructions
	3.3.3	AA	Error Suggestion
	3.2.4	AA	Consistent Identification
	3.3.8**	AA	Accessible Authentication (Minimum)
	3.3.9**	AAA	Accessible Authentication (Enhanced)

<b>Concept</b>	<b>Criterion</b>	<b>Level</b>	<b>Description</b>
<b>Do not disturb</b>	2.2.4	AAA	Interruptions
	2.2.5	AAA	Re-Authenticating
<b>No time constraints</b>	2.2.1	A	Timing Adjustable
	2.2.3	AAA	No Timing
	2.2.6*	AAA	Timeouts
<b>Design for different sizes</b>	1.3.4*	AA	Orientation
	1.4.4	AA	Resize Text
	1.4.8	AAA	Visual Presentation
	1.4.10*	AA	Reflow
	1.4.12*	AA	Text Spacing



# Glossary

**A/B testing.** A type of testing where two versions of a website or app are compared to determine which one works best. [https://en.wikipedia.org/wiki/A/B\\_testing](https://en.wikipedia.org/wiki/A/B_testing)

**ACR - Accessibility Conformance Report.** It is a document that describes the degree of conformance of an information and communication technology product or service with an agreed set of international accessibility guidelines and standards.

<https://www.section508.gov/sell/how-to-create-acr-with-vpat/>

**Accessibility API** - API that exposes information about objects and events to supporting products. For example, MSAA, IAccessible2, or UI Automation are accessibility APIs for the Windows operating system.

<https://www.smashingmagazine.com/2015/03/web-accessibility-with-accessibility-api/>

**APCA - Accessible Perceptual Contrast Algorithm:** is a new method for calculating and predicting readability contrast.

<https://git.apcacontrast.com/documentation/APCAeasyIntro>

**API - Application Programming Interface.** It is an interface implemented by software that allows it to interact with other software. <https://en.wikipedia.org/wiki/API>

**ARIA - Accessible Rich Internet Applications.** It is a technical specification that provides ways to markup HTML code or other markup languages to improve the accessibility and interoperability of content and applications.

<http://www.w3.org/WAI/PF/aria/>

**ASCII - American Standard Code for Information Interchange.** It is a character encoding scheme based on the order of the English alphabet.

<https://en.wikipedia.org/wiki/ASCII>

**Assistive technology or technical aids.** Any technology (*hardware or software*) specially manufactured to prevent, compensate, control, mitigate or neutralize impairments, limitations on activity and restrictions on participation by users with disabilities. [https://en.wikipedia.org/wiki/Assistive\\_technology](https://en.wikipedia.org/wiki/Assistive_technology)

**CAPTCHA - Completely Automated Public Turing test to tell Computers and Humans Apart.** It is a test to determine whether the user using the system is a user or a machine. <http://es.wikipedia.org/wiki/CAPTCHA>

**COGA - Working Group on Accessibility to Cognitive and Learning Disabilities.** <https://www.w3.org/WAI/GL/task-forces/coga/>

**CRO - Conversion rate optimization.** It is the process of increasing the percentage of users or website visitors to take a desired action.

[https://en.wikipedia.org/wiki/Conversion\\_rate\\_optimization](https://en.wikipedia.org/wiki/Conversion_rate_optimization)

**CSS - Cascading Style Sheets.** It is a design language for defining the presentation of HTML pages or other markup languages independently of their structure.

<https://www.w3.org/Style/CSS/>

**CSS pixels.** They are the canonical unit of measurement for CSS measurements. This unit is independent of screen density and different from the physical pixels of the screens. <https://www.w3.org/TR/css3-values/#lengths>

**Decibel.** It is a measure of sound pressure. <https://en.wikipedia.org/wiki/Decibel>

**DOM - Document Object Model.** It is a cross-platform and language-independent interface that treats an HTML or XML document as a tree structure wherein each node is an object representing a part of the document.

[https://en.wikipedia.org/wiki/Document\\_Object\\_Model](https://en.wikipedia.org/wiki/Document_Object_Model)

**ETSI - European Telecommunications Standards Institute.** It is a non-profit organization made up of more than 800 global organizations that helps members develop, test, and ratify standards. <http://www.etsi.org/about>

**Flexbox.** It is a property of CSS to adapt the size and position of elements to different screen sizes and different devices.

[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)

**HTML - Hypertext Markup Language.** It is the predominant markup language for web pages. <https://www.w3.org/TR/html/>

**HTTP - Hypertext Transfer Protocol.** It is a network protocol for distributed, collaborative, and hypermedia information systems.

<https://en.wikipedia.org/wiki/HTTP>

**ISO - International Organization for Standardization.** It is an international organization that publishes standards for all types of products and services.

<http://www.iso.org>

**ITI - Information Technology Industry Council.** It is an association that represents a group of important technology companies. <https://www.itic.org/about/>

**JavaScript.** It is an object-oriented client-side programming language.

<https://en.wikipedia.org/wiki/JavaScript>

**Keyboard interface.** It is an interface a program uses to obtain keystrokes even when the native technology does not contain a keyboard such as a touchscreen or voice recognition applications. <http://www.w3.org/TR/2008/REC-WCAG20-20081211/#keybrd-interfacedef>

**Leet.** It is a type of writing composed of alphanumeric characters. For example, the word "Hacker" is spelled "H4x0r". <https://en.wikipedia.org/wiki/Leet>

**Microformats.** They are small HTML patterns used to semantically represent people's contacts, calendar events, places, etc. <http://microformats.org>

**OCR - Optical character recognition.** It is a process of optical recognition of characters in documents, to transform them into text.

[https://en.wikipedia.org/wiki/Optical\\_character\\_recognition](https://en.wikipedia.org/wiki/Optical_character_recognition)

**PDF - Portable Document Format.** It is an open standard for the exchange of documents in a certain format. <https://en.wikipedia.org/wiki/PDF>

**Programmatically determined:** When the semantic structure of a website is understandable through a page's markup, it can be *programmatically determined*.  
<https://www.boia.org/blog/what-programmatically-determined-means-for-accessibility>

**Section 508.** It is the U.S. standard for making technology and information accessible to people with disabilities. <http://www.section508.gov>

**SEM – Search engine marketing.** It is a form of Internet marketing that involves the promotion of websites by increasing their visibility in search engine pages primarily through paid advertising. [https://en.wikipedia.org/wiki/Search\\_engine\\_marketing](https://en.wikipedia.org/wiki/Search_engine_marketing)

**SEO – Search Engine Optimization.** It is the process of improving the quality and quantity of website traffic to a website or a web page from search engines.  
[https://en.wikipedia.org/wiki/Search\\_engine\\_optimization](https://en.wikipedia.org/wiki/Search_engine_optimization)

**SIDAR - Ibero-American Seminar on Disability and Accessibility on the Internet.** It is a permanent and voluntary working group, made up of experts in innovative technologies and their accessibility. <http://www.sidar.org/>

**SMIL - Synchronized Multimedia Integration Language.** It is a W3C standard for using XML markup to describe multimedia presentations.

[https://en.wikipedia.org/wiki/Synchronized\\_Multimedia\\_Integration\\_Language](https://en.wikipedia.org/wiki/Synchronized_Multimedia_Integration_Language)

**SVG - Scalable Vector Graphics.** It is a format of two-dimensional vector graphics, both static and animated, in XML format. <https://en.wikipedia.org/wiki/SVG>

**Unicode.** It is a computer industry standard for encoding, rendering, and handling text consistently across various systems. <http://unicode.org>

**VPAT - Voluntary Product Accessibility Template.** It is a template created by the ITI (Information Technology Industry) to prepare an ACR (Accessibility Conformance Report). It is available for all major accessibility standards: Section 508 (United States), EN 301 549 (European Union), and W3C/WAI WCAG (versions 2.0, 2.1, and 2.2). <https://www.itic.org/policy/accessibility/vpat>

**W3C - World Wide Web Consortium.** It is an international non-profit organization of public interest, where member organizations, a full-time staff, and the public work together to develop web standards. <http://www.w3.org>

**WAI - Web Accessibility Initiative.** It is a W3C group that develops strategies, guidelines, and resources to help make the internet accessible to people with disabilities. <http://www.w3.org/WAI/>

**WCAG - Web Content Accessibility Guidelines.** They are a series of recommendations from the W3C to make web content accessible to people with disabilities.  
<http://www.w3.org/WAI/intro/wcag>

**XHTML - eXtensible Hypertext Markup Language.** It is an HTML document expressed as valid XML, stricter at a technical level.

<http://es.wikipedia.org/wiki/XHTML>  
<https://en.wikipedia.org/wiki/XHTML>

**XML - Extensible Markup Language.** It is a simple, yet flexible text formatting language designed to exchange a wide variety of data.

<https://en.wikipedia.org/wiki/XML>



# Global Index

<b>Introduction to Web Accessibility</b>	1
What is Web Accessibility?	3
Accessible interactions	6
What professionals should know about Web Accessibility	11
What you need to know according to your profession	13
Evaluation in Web Accessibility	19
<b>Implementing Web Accessibility policies</b>	21
Why do websites have accessibility problems?	23
Arguments for investing in Web Accessibility	25
Accessibility Maturity Model	28
Steps to integrate Web Accessibility into an organization	30
Examples of "Personas" for an online university	36
<b>The WCAG Guidelines</b>	39
What are the WCAG Guidelines?	41
What was new in WCAG 2.1	43
What was new in WCAG 2.2	43
WCAG Guidelines 2 Documents	44
How WCAG 2.2 are organized	45
The Four Principles and Their Guidelines	46
The success criteria	48
Techniques and mistakes	49
What technologies do WCAG apply to?	50
Why following WCAG 2 and upgrading to 2.2	51
A sneak peek into the future: WCAG 3	52
<b>How to make an accessible website</b>	55
The 5 requirements of Web Accessibility	57
Web page compliance levels	58
Declarations of conformance	59

<b>Assessing Web Accessibility with WCAG-EM</b>	<b>71</b>
The WCAG-EM methodology	73
Review Process	74
Step 1. Define the scope of the assessment	75
Step 2. Explore Website	76
Step 3. Select a representative sample of pages	77
Step 4. Audit the selected sample	78
Step 5. Record the results and produce a report	79
Declaring conformance after following the WCAG-EM.	80
WCAG-EM Report Tool	81
<b>Principle 1. Perceivable</b>	<b>83</b>
Guideline 1.1 Offer <b>text</b> alternatives	85
Guideline 1.2 <b>Time-based media</b>	89
Guideline 1.3 <b>Adaptable</b> , show content in different presentations.	101
Guideline 1.4 <b>Distinguishable</b> , separate background from foreground	113
<b>Principle 2. Operable</b>	<b>135</b>
Guideline 2.1 Accessible with a <b>keyboard</b> , not everybody uses a mouse	137
Guideline 2.2 Give <b>enough time</b> , don't rush	143
Guideline 2.3 <b>Flashes and motion animations</b> under control	153
Guideline 2.4 <b>Easily navigable</b> information architecture	157
Guideline 2.5 <b>Input modalities</b> , multiple ways to enter information	175
<b>Principle 3. Understandable</b>	<b>187</b>
Guideline 3.1 <b>Readable</b> , clear for everybody	189
Guideline 3.2 Make your page <b>predictable</b> , don't reinvent browsing standards	197
Guideline 3.3 <b>Help</b> users to input data. To err is human.	205
<b>Principle 4. Robust</b>	<b>217</b>
Guideline 4.1 Respect and care for <b>the code</b>	219
<b>Accessible PDF documents</b>	<b>225</b>
Accessibility in PDF documents	227
PDF-specific techniques	228
Implement PDF techniques	229
Validate PDF accessibility	246
Export to PDF	248

<b>ARIA, the ally of accessible HTML</b>	<b>251</b>
What is ARIA?	253
Roles	258
States and properties	260
Best practices for ARIA	272
How to Review ARIA	275
The specific techniques of ARIA	282
<b>Accessible Single Page Applications</b>	<b>285</b>
What is a SPA	287
Status messages	288
Dynamic page title	289
Keyboard Focus Control	290
Final recommendations	293
<b>Accessible Design Systems</b>	<b>295</b>
What is a Design System	297
How to Design and Develop an Accessible Design System	298
Documenting the Design System	301
<b>Caring for the words</b>	<b>309</b>
The importance of words	311
Clear communication	312
Writing in plain language	315
Example of communication improvement	317
Text easy to read and understand	320
Inclusive language	328
<b>Validation tools</b>	<b>333</b>
WCAG 2.2 Audit Tool	335
Automatic global validation tools	336
Manual validation tools	341
Criteria validation tools	347
Automated accessibility testing	357
<b>Work tools</b>	<b>359</b>
Screen readers	361
Accessible audio and video players	364

<i>Simulating Visually Impaired Access</i>	365
<i>Design-to-development handover documentation</i>	366
<b>Summaries and Diagrams</b>	<b>369</b>
<i>WCAG 2.2 Concept Diagrams</i>	371
<i>Principles, Guidelines, and Conformity Criteria</i>	376
<i>Tiered Conformity Criteria</i>	381
<i>Criteria for content creators</i>	384
<i>Criteria for UX/UI Designers</i>	386
<i>Criteria for developers</i>	388
<b>Glossary</b>	<b>391</b>
<b>Global Index</b>	<b>395</b>



**Olga Revilla Muñoz**  
**Itákora**

For more than two decades, I have worked as an interaction designer in all kind of projects in established corporations, emerging startups, and everything in between.

I have also founded my own studio, Itákora, where I invest in technology, innovation and design to improve society in all its aspects.

I love consulting, teaching and spreading the word about Web Accessibility and User Experience.

I have also published several books and scientific papers on human factors.

I hold a degree in Journalism, a PhD in Industrial Engineering and a Master in Business Administration.

[itakora.com](http://itakora.com)



**Olga Carreras Montoto**  
**Usable y Accesible**

I am an independent digital accessibility consultant with 20 years of experience and the author of the blog "Usable y accesible".

I conduct accessibility audits for both the public and private sectors. I provide training in companies, public institutions, and various university postgraduate programs.

I have published other guides, such as "Accessible Mobile Applications", "Accessible PDFs with Adobe Acrobat Professional", and "Accessible PowerPoint Documents".

I am a Web Accessibility Specialist (WAS) certified by the International Association of Accessibility Professionals (IAAP).

[olgacarreras.blogspot.com](http://olgacarreras.blogspot.com)