IAAP
International Association
of Accessibility Professionals
A division of G3ict

IAAP WAS

WEB ACCESSIBILITY SPECIALIST

Body of Knowledge

October 2024

United in Accessibility
www.accessibilityassociation.org

Version 2.3

This page intentionally left blank.

2024 WAS BoK Co-Editors:

- Sheri Byrne-Haber, CPACC
- Detlev Fischer, WAS
- Sonja Weckenmann, WAS

2017 WAS Original BoK Contributors:

- Dr. Paul Bohman, CPWA
- Pina D'Intino, CPACC
- Katie Haritos-Shea, CPWA
- Eric Hind, CPACC
- David McDonald, CPACC
- Rosemary Musachio, CPWA
- Radek Pavlíček, CPWA
- Allison Ravenhall, CPWA
- Paul Rayius
- Damian Sian, CPWA
- Stacy Iannaccone, CPACC

Staff Liaison

Samantha Evans

This page intentionally left blank.

IAAP Web Accessibility Specialist (WAS)Table of Contents

This page intentionally left blank.

IAAP WAS Body of Knowledge Introduction

1. The Purpose of this Document

WAS knowledge includes comprehending web accessibility theory, principles, and fundamental information appropriate to an intermediate level of skills and experience. The IAAP WAS designation requires three to five years of hands-on work experience in or with a web accessibility team.

Specifically, the three main purposes of this document are as follows:

1. List the categories of information covered in the exam
2. Present general information about each category
3. List additional resources to help test takers prepare for the exam

The Body of Knowledge is designed to be a starting point when studying for the WAS exam. It is not intended to be an exhaustive explanation of every concept or question on the exam. Please note that the use of this document does not guarantee success on the IAAP WAS exam.

The WAS BOK covers concepts from WCAG 2.0, WCAG 2.1, WCAG 2.2 as well as EN 301 549.

Unless specific information is discussed about a certain version, “WCAG” will be used.

If you discover any broken links, please contact certification@accessibilityassociation.org.

2. IAAP WAS Exam Preparation Resources

Test-takers can study resources available anywhere in preparation for the exam. The IAAP lists a collection of WAS Exam resources for preparation. You can find these resources on the IAAP Prepare for the WAS Exam webpage. Some of them are free of charge, others must be purchased.

Candidates should review each section of the WAS Content Outline to determine where they have the most background, where they have some knowledge, and to identify sections that are less familiar where they will spend most of their time studying to prepare for the WAS Exam. Candidates may also utilize the WAS BOK for more detailed review and study preparation. All WAS Exam items are written from content contained in this WAS BOK.

3. About the WAS Designation

The IAAP Web Accessibility Specialist credential is intended for accessibility professionals who are expected to evaluate the accessibility of existing content or objects according to published technical standards and guidelines and provide detailed remediation recommendations.

The WAS exam allows individuals to certify their skillset in the specialized professional discipline of web accessibility. Individuals who pass the Certified Professional in Accessibility Core Competencies (CPACC) and the Web Accessibility Specialist (WAS) exams are eligible to carry a higher-level credential called the Certified Professional in Web Accessibility (CPWA).

The WAS credential represents an ability to express technical proficiency for someone with at least an intermediate level of experience designing, developing, implementing, and evaluating accessible web-based content, projects, and services. This exam is not intended for beginners or those without regular hands-on experience in remediating or identifying accessibility issues in code. The WAS exam is not intended to illustrate or assess the ability to write code. Knowledge of HTML authoring alone will not provide the background necessary to achieve the WAS credential successfully. Hands-on experience and knowledge of programmatic code elements, W3C standards, and contextual implications for end users of assistive technology are all required.

Web Accessibility Specialists are expected to know and use the relevant technologies, not merely be aware of them. Relevant domains for the WAS designation include:

- creating accessible web content
- identifying accessibility issues/problems
- remediating (fixing) accessibility issues

Web accessibility refers to the inclusive practice of making the web usable by people of all abilities and disabilities.

Expected results from WAS exam:

The following knowledge will be demonstrated in the exam:

- An understanding of general ICT accessibility principles based on the needs of persons with disabilities and how they use the Internet, the tools they can use, and the creation or support of an accessible ICT ecosystem
- How to create accessible digital content
- General principles on how to implement ICT products and services that are accessible for persons with disabilities

4. The WAS Exam Content at a Glance

Domain One: Creating Accessible Web Content (40% of the exam)

- Accessibility standards and specifications
- Create accessible Interfaces with JavaScript
- Create interactive controls/widgets based on accessibility best practices
- Integrate accessibility into the product life cycle
- Create web content that is compatible with the strategies used by users with disabilities to access web content

Domain Two: Identify Accessibility Issues/Problems (40% of the exam)

- Identify issues regarding to accessibility needs of users with disabilities
- Determine conformance with accessibility standards
- Use accessibility testing tools effectively
- Test with assistive technologies
- Test for end-user impact

Domain Three: Remediate (fix) Accessibility Issues (20% of the exam)

- Communicate and prioritize accessibility issues

5. Additional Resources

- IAAP Homepage
- General Information about IAAP Certification
- WAS Exam Content Outline
- WAS Frequently Asked Questions
- WAS Preparation Resources
- WAS Certification Process

WAS Content Outline

Domain One: Creating Accessible Web Solutions -- (40%).

Domain One A: Guidelines, principles and techniques for meeting success criteria

(including WCAG 2.2, WAI-ARIA, ATAG, EN 301 549, basic concepts, limitations of the specific guidelines, principles, and techniques, what is normative vs. non-normative; what is included in different levels (A, AA, AAA)).

Study Tasks

Understand and interpret WCAG 2.2

- Be familiar with WCAG 2.2 Success Criteria
- Understand the relationship between principles, guidelines, and success criteria.
- Understand the intent, requirements, and impact of each principle, guideline, and success criterion.
- Be familiar with Sufficient Techniques, Failure, and Advisory Techniques for each success criterion.
- Understand the conformance level designations (A, AA, AAA).
- Identify the conformance level of each WCAG 2.2 success criterion.
- Understand the three types of techniques and the W3C vetting process for techniques.
- Understand the difference between normative and non-normative information in WCAG 2.2.

Understand and interpret WAI-ARIA 1.2

- Understand the purpose and impact of WAI-ARIA 1.2.
- Understand the WAI-ARIA 1.2 model of roles and properties.
- Know when and why to use WAI-ARIA 1.2, and when to use standard HTML instead.

Understand and interpret ATAG 2.0

- Understand how ATAG 2.0 applies to web content authoring tools.
- Understand the meaning and intent of the two main sections of ATAG 2.0.

- Understand the intent, requirements, and impact of each principle, guideline, and success criterion.
- Distinguish between good, automated practices in authoring tools and good practices that require author/user input.
- Understand the power and limitations of automated accessibility authoring features.
- Understand the difference between normative and non-normative information in ATAG 2.0.

Understand and interpret EN 301 549

- Know what kind of technologies are addressed by EN 301 549
- Identify the similarities and differences between WCAG and EN 301 549
- Describe how WCAG is represented in EN 301 549
- Identify requirements beyond WCAG that are relevant for compliance with the Web Accessibility Directive 2016/2102.
- Understand the intent and scope of requirements in Annex A Table A1 that go beyond WCAG.

Overview

This competency focuses on designing and creating web content in accordance with the W3C accessibility guidelines and specifications as well as Annex A of the EN 301 549.

Resources

W3C Web Accessibility Initiative (WAI)

- Essential Components of Web accessibility
- WCAG 2 Overview
- The WCAG 2 Documents
- W3C Process for Developing Standards
- What's New in WCAG 2.2
- ATAG at a Glance

- WAI-ARIA Overview

Web Content Accessibility Guidelines (WCAG) 2.2

- WCAG 2.2 (normative)
- How to Meet WCAG 2.2 (non-normative)
- Techniques for WCAG 2.2 (non-normative)

Authoring Tool Accessibility Guidelines (ATAG) 2.0

- ATAG 2.0 (normative)
- Implementing ATAG 2.0 (non-normative)

Accessible Rich Internet Applications (WAI-ARIA) 1.2

- WAI-ARIA 1.2 (normative)
- ARIA Authoring Practices Guide (non-normative)

European Norm 301 549

- EN 301 549 V3.2.1 (2021-03) (PDF)
- Introduction to the EN 301 549 V.1.1.2 (YouTube) by Funka & Microsoft, 2017
- Latest changes to accessibility standard, EU Commission, 2022

1. The W3C Guidelines and Specifications

Accessibility depends on several components working together, such as web content, browsers, assistive technologies, and authoring tools. The World Wide Web Consortium (W3C), an internationally recognised web standards body, identifies its approved technical standards as "W3C Recommendations" related to these components.

W3C WAI Guidelines associated with accessibility include:

- **Web Content Accessibility Guidelines (WCAG):** Addresses web content.
- **User Agent Accessibility Guidelines (UAAG):** Addresses software that people use to access the web, for example, browsers and media players (UAAG is out of the scope of the IAAP WAS certification).
- **Authoring Tools Accessibility Guidelines (ATAG):** Addresses software and services that people use to create web content.

The accessibility guidelines are based on technical specifications, such as **HTML and CSS**, whereas the technical specification **Accessible Rich Internet Applications** (**WAI-ARIA) 1.2** addresses accessibility directly.

Guidelines and specifications provide **"normative"** documents. They define accessibility practices required for conformance and its verification. The normative part of the WAI guidelines identifies the minimum to make a website or software accessible for users with disabilities. In addition, there are many best practices that benefit people and are therefore recommended to be used.

"Non-normative" documents provide guidance and techniques for interpreting and conforming with the normative requirements, but they are not required for conformance. They provide information about the different ways web technologies work together with user agents and assistive technologies to provide an overall accessible experience for end users. Non-normative documents may change more frequently than normative documents, to adapt to changing technologies and current best practices.

1.1 Web Content Accessibility Guidelines (WCAG) 2.2

The international standard, WCAG 2.2, covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content more accessible to a wider range of people with disabilities, including accommodations for blindness and low vision, deafness and hearing loss, limited movement, speech disabilities, photosensitivity, and combinations of these, and some accommodation for learning disabilities and cognitive limitations. However, it will not address every user need for people with these disabilities. WCAG is device-agnostic to address the accessibility of web content on desktops, laptops, tablets, and mobile devices. Following these guidelines will also often make Web content more usable to users in general.

WCAG is developed through the W3C process in cooperation with individuals and organizations around the world, sharing the goal to provide a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally. The WCAG documents explain how to make web content more accessible to people with disabilities.

1.2 WCAG Overview

The WCAG document is organised into Principles, Guidelines, and Success Criteria. The four POUR Principles are designed to guide you in thinking about how people use web content:

- **Perceivable:** Information and user interface components must be presentable to users in ways persons with disabilities can perceive (including blindness, low vision, deafness and hearing loss, limited movement, and cognitive limitations).

- **Operable:** User interface components and navigation must be operable in different ways. For example, someone who cannot use a mouse may use the keyboard or voice recognition instead of a mouse or trackpad.
- **Understandable:** Information and the operation of the user interface must be understandable. For example, someone who is blind needs their screen reader to speak foreign language content with the correct pronunciation, and someone who has a learning disability wants things to work consistently within a website.
- **Robust:** Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

WCAG has an additive approach. This ensures that websites conforming to later versions of WCAG also conform to earlier versions. Websites conforming to WCAG 2.2 or WCAG 2.1 also conform to WCAG 2.0.

Each Guideline has at least one or more Success Criteria. Each Success Criterion is assigned a conformance level: A, AA, or AAA. The Success Criteria determine "conformance" to WCAG. A web page that meets all the Success Criteria at level A is said to “conform to” level A. A web page that meets all the Success Criteria of both level A and level AA conforms to level AA. A web page that meets all the Success Criteria at all levels conforms to level AAA. WCAG states that "It is not recommended that Level AAA conformance be required as a general policy for entire sites because it is not possible to satisfy all Level AAA Success Criteria for some content." (see Conformance requirements of WCAG).

Most laws and policies that reference WCAG focus on conformance to level AA.

1.3 WCAG Supporting Documents

The W3C has published Techniques for meeting WCAG 2.0, WCAG 2.1 and WCAG 2.2. The Techniques can be one of three kinds:

- **Sufficient Techniques:** If the web content meets Sufficient Techniques, it successfully meets the success criterion.
- **Failure Techniques:** If the web content fails any of these Techniques, it does not meet the success criterion.
- **Advisory Techniques:** Optional or conditional Techniques may represent accessibility best practice or possible ways of meeting the Success Criterion.

The Techniques published by the W3C are not normative, but they have been vetted by a group of accessibility professionals. The Techniques may change over time due to evolving technologies, improvements in accessibility support, or the emergence of new accessibility-related specifications.

The process of updating or retiring Techniques is slow. It is worth noting that some Techniques now appear dated or even irrelevant. Also, solutions described in some Sufficient Techniques may now be considered insufficient when better implementations have become available over time.

The Understanding Documents provide detailed explanations to WCAG Guidelines and Success Criteria. The supporting documents help readers better understand the intent of the Success Criteria. Like the Techniques, they are not normative.

1.4 WCAG Versioning

Candidates should understand the history of WCAG releases from WCAG 2.0 in 2008 and WCAG 2.1 in 2018 and WCAG 2.2 in 2023. They should demonstrate an understanding of the changes introduced in WCAG 2.1 and WCAG 2.2.

WCAG 2.1 added Success Criteria for new input modes like touch and speech, and more criteria that focus on individuals with low vision or cognitive disabilities which were not sufficiently addressed in WCAG 2.0.

WCAG 2.2 became a final recommendation in October 2023. It added further requirements, some focusing on cognitive aspects such as authentication, form entries, or the provision of help, others supporting keyboard operation (Focus not Obscured) or pointer operation (Target Size, Dragging Movements).

A new major version of WCAG, WCAG 3.0, is in its early stages of development. WCAG 3.0 is likely to introduce significant changes to the structure of requirements, adding requirements that have not been deemed testable so far, and introduce a new the conformance model. The gestation period of WCAG 3.0 until it becomes a Public Recommendation is likely to still take several years.

1.5 Authoring Tool Accessibility Guidelines (ATAG) 2.0

Authoring tools are software and services that authors use to produce web content, for example WYSIWYG editors, site builders, form generating software, social media sites or learning management systems.

If authoring tools take accessibility into account, more content creators, including people with disabilities, will be able to use these interfaces and produce web content. Content itself will be

more accessible because the tools following the guidelines will provide the resources needed to create accessible web content and avoid accessibility problems.

ATAG 2.0 specifies requirements primarily for developers of authoring tools. Besides, the guidelines and supporting documents help people to evaluate and choose software that provides an accessible interface and accessible content output.

1.5.1 ATAG Overview

ATAG has two main parts:

A. Providing an accessible user interface and
B. Supporting the production of accessible content.

Like WCAG, ATAG is organised into Principles, Guidelines and Success Criteria. The Success Criteria specify the testable requirements assigned to three conformance levels: A, AA and AAA. Implementing ATAG 2.0 refers to ATAG Guidelines and Success Criteria. It is a supporting document that is intended to provide guidance for understanding ATAG. It is not normative. While ATAG currently refers to WCAG 2.0, it is not difficult to take the new success criteria from WCAG 2.1 and 2.2 and incorporate them into an ATAG analysis.

1.6 Accessible Rich Internet Applications (WAI-ARIA) 1.2

WAI-ARIA (Accessible Rich Internet Applications) is a technical specification of the Web Accessibility Initiative (WAI). WAI-ARIA is intended to increase the accessibility of content for assistive technology users, such as screen reader users. The ARIA specification provides a set of attributes to fill in missing semantics, particularly for dynamic content and custom widgets.

While the technical specification ARIA 1.2 defines an ontology of roles, the non-normative ARIA Authoring Practice Guide describes applying ARIA roles, states, and properties to common design patterns and widgets.

Versions of the ARIA specification can differ in structure and semantics. This difference can significantly impact user agent and screen reader support of web content. Knowing the current status of the ARIA specification is important to identify outdated implementations and the level of accessibility support of new roles and attributes.

ARIA attributes need to be used carefully. To understand the importance of semantics, it is necessary to know how assistive technologies get this information.

2. European Accessibility Standard EN 301 549

EN 301 549 is a so-called harmonised European standard. It is developed by the three official European Standards Organisations CEN, CENELEC, and ETSI on the explicit demand (“Mandate")

of the European Commission, to support European legislation. Harmonised European standards provide voluntary technical specifications for presumed conformity. Using a standard is an efficient way to show compliance, but an organisation is free to choose other technical solutions to fullfil the legal obligations.

EN 301 549 was originally developed to support procurement of accessible products and services. The current version (3.2.1) was published in 2021. Annex A contains tables showing presumed conformance for websites and apps of the Web Accessibility Directive (WAD), (Directive (EU) 2016/2102), covering public sector bodies. The EN 301 549 is currently being updated under Mandate 587, together with several other standards, to support the European Accessibility Act (EAA) that will enter into force on 28th of June 2025.

2.1 EN 301 549 Overview

EN 301 549 defines accessibility requirements for ICT (Information and Communication Technology) products and services. The scope of the standard is broader than web accessibility and includes a variety of ICT solutions, such as hardware, software, and support services, as well as websites, documents, and apps.

EN 301 549 contains a list of functional performance statements, or user needs, for locating, identifying and operating ICT functions, and for accessing information, regardless of physical, cognitive or sensory abilities (Clause 4). The technical requirements of clauses 5 to 13 address "Generic requirements" applying to all ICT products, "ICT with two-way voice communication," "ICT with video capabilities," "Hardware," "Web," "Non-web documents," "Software," "Documentation and support services" and "ICT providing relay or emergency service access."

Annex A presents the relationship between the technical specifications and the Web Accessibility Directive. Annex B contains a table, mapping the functional performance statements (user needs) of clause 4 with the technical specifications of the standard in clauses 5-14. This makes it possible to assess the impact of specific requirements and how they affect user groups. (see also chapter 2.1 of the BOK). Annex C describes the test procedures and evaluation methodology for determining conformance with each accessibility requirement.

2.2 Adoption outside Europe

EN 301 549 has relevance far beyond the European Union: It has been adopted as a standard in Australia as AS EN 301 549 and is also used in other countries such as Canada, Mexico, and Kenya.

The Information Technology Industry Council (ITI) has produced a Voluntary Product Accessibility Template (VPAT 2.4 Rev EU) for EN 301 549.

2.3 WCAG reflected in EN 301 549

When it comes to web accessibility, EN 301 549 intends to harmonise with WCAG in clauses 9, 10 and 11. The current version (3.2.1) references WCAG 2.1 success criteria level A and AA as well as the non-normative W3C Guidance on Applying WCAG 2.0 to Non-Web Information and Communications Technologies (WCAG2ICT). Furthermore, WCAG AAA success criteria are referred in 9.5, and the WCAG conformance requirements in clause 9.6.

2.4 Self-scoping requirements (applicability language)

The European Standard 301 549 provides accessibility requirements that can be applied to any ICT product or service. Therefore, the structure is based on a feature-based approach. That is, each requirement except those in clause 12 (related to documentation and support services) has a wording that queries for its applicability (self-scoping or preconditions). For example: “Where ICT displays video with synchronised audio, it shall have a mode of operation [...]". This language is intended to identify whether each clause applies to any ICT product or service.

If the pre-condition (for example, ICT displays video with synchronised audio) is not given, the requirement is not applicable. If the pre-condition is given, you have to assess, whether the requirement passes or fails the corresponding test provided in Annex C. Although Annex C lists a test for each requirement of the standard, it does not provide a detailed testing methodology.

Domain One B: Basic knowledge of programming

(at a conceptual level; principles and concepts related to programming; impact of specific coding practices on web solutions vs. writing specific code).

Study Tasks

Create Accessible Interfaces with JavaScript

- Understand the concept of progressive enhancement.
- Understand how JavaScript can improve or damage accessibility and user experience depending on its use.
- Describe the impact of device-independent event handlers on accessibility.
- Understand the impact of an `onClick()event` applied to a native `<a>` or `<button>` element in contrast to applying it to a non-semantic element, such as a generic `<div>`.

- Be familiar with the user's expectations regarding focus movement.
- Know when and how to send focus to new content using JavaScript.
- Understand the methods that can be used to notify screen readers that new content has been added dynamically.

Overview

The WAS exam does not cover the details of JavaScript programming syntax. This is because a person should be able to pass the exam without being a professional JavaScript programmer. However, web designers and developers must know how asynchronous JavaScript and interactive content affect accessibility. They must be able to identify the concepts, principles, and strategies of accessible JavaScript interaction design.

1 Support for JavaScript in Accessibility APIs and Assistive Technologies

JavaScript can be processed by modern screen readers and other assistive technologies as long as JavaScript is coded with accessibility in mind. There are no inherent barriers in the technologies that would make JavaScript inaccessible. However, it is generally recommended to develop web content with a progressive enhancement approach in mind. This means that the essential content and functionality are available even when accessibility-supported technologies like JavaScript are unavailable due to old browsers, restrictive settings, bandwidth limitations, or internal company policies.

2 Semantic HTML and Custom Controls

HTML defines sets of elements, attributes, and attribute values. These features have specific semantic meanings that user agents intend to process in particular ways. Keyboard accessibility is another crucial aspect of web accessibility. Many people rely on keyboard support, such as keyboard-only users or assistive technology users. Native interactive HTML elements provide built-in keyboard functionality.

If you use an `onClick()event` on a semantic HTML element, such as a `<button>` or `<a>`, it includes both mouse and keyboard functionality by default. If you use a non-semantic element, like a `<div>` and you add an `onClick() event`, keyboard functionality is not automatically included even if the element is made focusable using `tabindex`, or it is focused using scripting. You must add a `keydown/keyup event` to detect the enter or space key presses.

3 Device Independent Event Handlers

Create device-independent event handlers. The functionality must be available when using different input modes, such as keyboard, mouse, touch, or voice. Device-dependent event handlers rely wholly upon a certain type of input, such as a pointer device.

4 Simplify Events

Buttons and other interactive elements should generally have only one type of event associated with them. Multi-event elements are more difficult to make accessible and more difficult for users to understand. For example, a menu item should act as a link or as a button that expands a submenu. Coding a menu item to expand the menu on hover and activate a link on click is problematic for keyboard users, touch users, and gesture-based mobile screen reader users.

5 Dynamic Content

JavaScript provides the functionality to add HTML and CSS to any page element. That means developers can create dynamic applications depending on user interactions. For example, developers can toggle CSS classes to hide and show content for dynamic HTML, change values of ARIA attributes to reflect the state of an element, or add content to a live region container.

6 Managing Focus

When operating content with the keyboard, it is essential to ensure that the focus is easy to locate and to provide a logical focus sequence. The focus movement must be predictable and meet users' expectations. Focus management also means knowing when to trap focus (for example, within a modal component).

Whenever JavaScript changes the visual focus (e.g. when a dialog is activated), JavaScript should be used to manage keyboard focus to follow the visual focus.

Consider DOM order and likely focus location when adding or updating content dynamically. When content is added or altered on a page, it should generally be added after the current point of focus because screen reader users are much less likely to navigate backward in the DOM than forward, causing them to miss most additions/changes in previous positions in the DOM.

JavaScript widgets made accessible using ARIA require keyboard implementation and focus management according to keyboard interface conventions. It is recommended to follow the APG patterns. There are two strategies for managing focus inside composite widgets depending on the widget type: creating a roving `tabindex` and using the `aria-activedescendant`

property (see details on both strategies in Domain One D3: Keyboard interaction and focus management within components).

7 Managing State

JavaScript is critical to accessibility, as it ensures that assistive technologies can gather information about the status of user interface controls.

When using standard HTML user interface elements according to the specification, the state of the element is conveyed to assistive technologies. Additional effort is needed for custom controls. Information about a component's state must be managed using ARIA state attributes. JavaScript is used to toggle their values. Examples of states of user interface controls include whether an element is selected or whether a collapsible component is expanded or collapsed.

If parts of a page are changed dynamically (with JavaScript) without reloading the entire page, these changes are not obvious to users of assistive technologies since they do not receive focus. Such updates, for example, information on the success or result of an action, have to be conveyed to screen reader users using ARIA live regions. For the content to be reliably announced across platforms by assistive technology, the container marked up as live or alert region must be present in the DOM when loading, and then the text can be added or exchanged dynamically.

Resources

- CSS and JavaScript Accessibility Best Practices, MDN, 2023
- HTMLElement: focus() method, MDN, 2023
- Accessible JavaScript, WebAIM, 2020
- Recommendations for Single Page Applications, Orange, 2021
- What we learned from user testing of accessible client-side routing techniques with Fable Tech Labs by Marcy Sutton, Gatsby 2019
- Engaging users to create accessible client-side routing (YouTube) by Marcy Sutton and Samuel Proulx, #id24 2019

Domain One C: Create Interactive Controls/Widgets Based on Accessibility Best Practices

Study Tasks

- Be familiar with the 5 rules for using ARIA.
- Understand the importance of coding to standards, rather than to the quirks or features of only one set of technologies.
- Know about the semantic structure of the roles. Some roles require parent or child roles, and some roles require attributes.
- Be familiar with techniques for providing accessible names and descriptions.
- Be familiar with the authoring practices for custom widgets, including semantic structure, keyboard behavior, etc.
- Be familiar with the keyboard interaction model for ARIA custom widgets. There are general keyboard patterns and patterns specific to certain types of widgets.
- Understand the importance of testing web designs for accessibility across a variety of platforms, browsers, and assistive technologies. Do not just assume they will work, even if they technically conform to published accessibility specifications.
- Know how to determine when an inaccessible outcome is the result of flaws in the design versus poor support of the technology.
- Know when it may be appropriate to write code that overrides, supplements, or fixes bugs in browsers or assistive technologies (do this only with great care).

Overview

Web designers often like to push the limits of conventional web design by creating interactive controls or widgets, using custom designs that exceed the capabilities of standard HTML. These interactive components do not necessarily constitute accessibility barriers. They can be made accessible when designed with accessibility in mind. Accessibility techniques for custom controls and widgets usually require ARIA attributes and patterns, which should be thoroughly tested in different environments to ensure broad accessibility.

Resources

- Using ARIA, W3C
- ARIA Authoring Practices Guide (APG), WAI W3C
- Understanding Accessibility Support, W3C
- Techniques for WCAG 2.2, WAI W3C
- A11ysupport.io

Domain One D: Using ARIA

Study Tasks

- Understand and interpret WAI-ARIA 1.2
    - Understand the purpose and impact of WAI-ARIA 1.2.
    - Understand the WAI-ARIA 1.2 model of roles and properties.
    - Know when and why to use WAI-ARIA 1.2, and when to use standard HTML instead.

Overview

Websites or applications often come with complex user interface components. It is important to use these with care since they add more complexity to making content overall accessible. ARIA can both degrade and enhance accessibility, depending on its correct use.

Reasons to use ARIA include:

- Ensuring custom widgets are understood by assistive technology users
- Programmatically indicating relationships between elements
- Notifying users about dynamic updates using ARIA live regions
- Hiding irrelevant visible content from assistive technology
- Fixing inaccessible content without full recoding

Remember that ARIA roles and properties do not cause browsers to provide keyboard behavior or styling. It is the author's responsibility to make elements focusable, implement expected keyboard interaction, and update properties.

The document Using ARIA identifies 5 rules you should follow:

1. If you can use a native HTML element or attribute with the semantics and behavior you require already built in, instead of re-purposing an element and adding an ARIA role, state or property to make it accessible, then do so.
2. Do not change native semantics, unless you really have to.
3. All interactive ARIA controls must be usable with the keyboard.
4. Do not use `role="presentation"` or `aria-hidden="true"` on a focusable element.
5. All interactive elements must have an accessible name.

1. Understanding Accessible Names and Descriptions

The accessible name (and accessible description) of an HTML element is exposed in the browser's accessibility tree (see Domain One D4). It is a string associated with an element that indicates the purpose of the element to assistive technologies and distinguishes the element from other elements on the page. For example, the accessible name of a simple link is the link text included between the link's start and end tag. For a text field, it is usually the label (if the latter is correctly programmatically linked to the field).

An accessible description is also an author-provided string. With an accessible description, additional information can be associated, for example, an instruction for an input element.

Assistive technologies announce accessible names differently from accessible descriptions. Whereas the accessible name is presented first, the accessible description is presented last, sometimes after a slight delay.

When developing accessible web experiences, authors must make sure that an element (e.g. button, link or form input) can be identified by providing an accessible name (see WCAG Success Criterion 4.1.2 Name, Role, Value) and, where relevant, an accessible description.

The Accessible Name and Description Computation is an algorithm for user agents to determine the name and description of an element from web content. Sources can be, for example, link text or text within buttons, the value of the alt attribute of an image, programmatically linked label text of form elements, the value of an attribute like aria-label or title, or (possibly multiple) text nodes referenced by the `aria-labelledby` or `aria-describedby` attributes, User agents walk through a list of potential (and possibly divergent) naming (or describing) sources and use the one with the highest priority.

2. ARIA Authoring Practice Guide

The technical specification **Accessible Rich Internet Application (ARIA 1.2)** defines accessibility semantics to create an accessible web experience. Whereas the ARIA specification is a normative document, the W3C provides further helpful informative resources, such as **"**Using ARIA**"** and **"**ARIA Authoring Practice Guide (APG).**"**

The main objective of the ARIA Authoring Practice Guide is to illustrate the appropriate use of ARIA, as defined in the ARIA specification (ARIA 1.2). It aims to provide a broad understanding of the accessibility concepts illustrated by a wide range of patterns and examples.

The Authoring Practice Guide describes both naming and describing techniques and cardinal rules that an author should follow. That is, it contains information on how to provide an accessible name required in 4.1.2 Name, Role, Value, among other things.

The APG states (see Read Me First and Introduction) that the guide does not serve as a comprehensive design system or production-ready code. Therefore, testing code with different browsers and assistive technology combinations is essential before using it.

The sample code does not address problems caused by gaps in ARIA 1.2 support in browsers and assistive technologies, nor does it indicate compatibility with mobile browsers or touch interfaces.

The APG is organised into two main sections: "Patterns" and "Practices":

- The section **"Pattern"** describes various widget types, keyboard interaction, and ARIA roles, states and properties and gives examples demonstrating implementation.
- The section "**Practices"** explains how to meet accessibility needs, for example, landmark regions, providing accessible names and descriptions or developing a keyboard interface.

3. Keyboard Interaction and Focus Management within Components

ARIA design patterns reference user expectations and keyboard conventions, which derive from the GUI components of common desktop GUIs. The goal is to enable easy learning and efficient operation of keyboard interfaces on the web.

It is common across all platforms that the tab and shift + tab keys move focus from one UI component to another while other keys, primarily the arrow keys, move focus inside complex components with multiple focusable elements. The ARIA specification refers to a UI component that contains multiple focusable elements, as a composite widget.

There are two approaches for managing focus within components:

Managing Focus Using Roving Tabindex

The `tabindex` attribute plays an important role in implementing custom focus handling. By implementing a “roving tabindex” you lead focus to elements within a composite widget.

The element that needs to receive focus (the active element) has `tabindex="0"`. All other elements within the composite have `tabindex="-1"`. They are still programmatically focusable (using the `focus()method)` but removed from the natural tab order. By using a keyboard event listener, you can determine when the user is pressing a navigation key to move focus, such as the arrow key. When this happens, JavaScript toggles the previously focused element’s `tabindex="0"` to `tabindex="-1"`, and the to-be-focused element’s `tabindex="-1"` to `tabindex="-0"`, and sets focus `(element.focus())` on it.

Not all complex widgets lend themselves to the roving tabindex technique. For example, an editable Combobox marked up with `role="combobox"` needs `aria-activedescendant`, because actual DOM focus must remain inside the input element.

Managing Focus Using aria-activedescendant

When managing focus with `aria-activedescendant` property, you do not manipulate the `tabindex` attribute and move DOM focus among focusable elements within the container. Instead, only the container with the role that supports `aria-activedescendant` has DOM focus. When moving within the component using arrow keys, the elements get a kind of pseudo focus to expose to the assistive technology in which the element is currently active. The assistive technology can announce the active element even though the actual DOM focus is on the container element. The value of `aria-activedescendant` applied to the container element has to match the id of the focused element. That means when navigating within the component, the value of `aria-activedescendant` must change dynamically according to the position of the visual focus.

4. The Accessibility Tree and Its Impact on Users of Assistive Technology

People who use a website with assistive technology need information about an element's meaning, function, and state in a way that assistive technology can understand. Assistive technologies gain access to this information through the Accessibility API.

Whenever a browser processes an HTML document, it parses the code and converts it into an object structure called the Document Object Model (DOM). This interface's tree structure

represents all HTML elements and content. Technologies such as CSS and JavaScript use the object structure to target elements.

What you see in the browser is based on the DOM, not the original HTML source code. The browser modifies the DOM tree into a simplified form for assistive technologies (called the Accessibility Tree) and passes the information to the Accessibility API of the operating system. This is what assistive technologies can interact with. Any DOM content can affect what information is available in the Accessibility Tree.

The accessibility API provides information about the objects of a web page. It maps the structure of the page, much like the DOM. However, the information in the Accessibility Tree is reduced to the essential information relevant to assistive technologies. The information provided by ARIA attributes include the following:

- **role:** the type / kind of thing (e.g. “button”, “main content”)
- **state:** its situation (e.g. “disabled”, “expanded”, “checked”)
- **name:** its name or label (e.g. name of the button, label of a form control)
- **description:** its description (if you want to provide more description beyond the name)
- **relationship:** indication of the relationship between elements, if needed

The implicit ARIA semantics for the features of HTML are defined by the HTML Accessibility API Mappings specification.

5. WAI-ARIA Roles, States, and Properties

WAI-ARIA specifies attributes that can be added to standard HTML to define the role, state and property values of elements, especially for custom widgets. One of the goals of WAI-ARIA is to make web applications behave more like software components. WAI-ARIA widgets interact with the accessibility API of the operating system, providing assistive technologies with the semantics and live updates necessary for accessibility. WAI-ARIA divides the semantics into roles (the type defining a user interface element), states, and properties.

- **Roles:** ARIA roles provide the meaning of a non-semantic element in a document. Authors need to associate non-semantic elements to a WAI-ARIA role whereas native HTML elements already provide implicit roles.

  The specification categorises ARIA roles into:

- **Abstract roles** (used only for categorization, authors must not use them in content)
- **Widget roles** (for interactive controls, simple or composite – e.g. `button`, `menu`, `slider`, `progressbar`, `combobox`)
- **Document structure roles** (define presented contents/structure – e.g. `heading`, `table`, `cell`)
- **Landmark roles** (identify important parts of a document/page – e.g. `banner`, `navigation`, `main`)
- **Live region roles** (for dynamically updated content that should be announced by screen readers – e.g. `alert`, `status`)
- **Window roles** (for content that functions as a separate window – `alertdialog`, `dialog`)

The browser does not handle the ARIA role attribute the same as it would with an implicit role using a native HTML element. ARIA attributes do not affect the browser's behavior and therefore do not change the way elements behave.

- **States and Properties:** State and property attributes both provide specific information about an element.
  - **States** are attributes that reflect the state of an element (such as `aria-expanded` for a show/hide widget or `aria-checked` for a checkbox). The value may change frequently due to user interaction.
  - The values of **properties** (such as `aria-label`, `aria-labelledby`) are often less likely to change due to interaction. But frequency of change difference is not a strict rule, there are also properties, such as `aria-valuetext` that are expected to change often.

The specification categorises ARIA states and properties into widget attributes (e.g. `aria-selected`), live region attributes (e.g. `aria-live`), drag-and-drop-attributes (e.g. `aria-grabbed`) and relationship attributes (e.g. `aria-activedescendants`).

The information provided via ARIA attributes (role, state, name, relationship) is mapped by the browser to the operating system's accessibility API and is exposed to assistive technologies.

ARIA enables developers to markup dynamic content, including custom controls created with HTML and (asynchronous) JavaScript, to improve its accessibility. The ARIA Authoring Practices define the best practices for widget structure, keyboard behaviors, and dynamic content and promote interoperability.

Resources

- Accessible Rich Internet Applications (WAI-ARIA) 1.2
- Using ARIA
- ARIA Authoring Practice Guide (APG)
- Understanding Accessibility Support, W3C
- Techniques for WCAG 2.2, WAI W3C
- A11ysupport.io

Domain One E: Accessibility-supported technologies

(including user's assistive technologies and accessibility features; combination of assistive technologies and users agent; design decisions in choosing technologies that support accessibility; e.g., differences in assistive technology supports and behaviors, differences in support for touch when screen reader is on vs. off).

Study Tasks

- Study how assistive technologies interact with user agents(browsers, apps).
- Understand differences in AT behavior across platforms and devices.
- Learn how touch interactions change with screen readers (e.g., gestures on mobile with VoiceOver).
- Understand the role of ARIA in enhancing AT support for dynamic content.

Overview

Most modern screen readers have comparable support for the most important accessibility techniques and features. Still, screen reader behavior and support for newer features—including some types of custom ARIA widgets—can sometimes be incomplete, inconsistent, or buggy. It is generally preferable to use a native HTML element or attribute with the appropriate semantics and functions already built in instead of repurposing an element and adding an ARIA role, state, or property.

WCAG requires only to use accessibility supported technologies to meet the success criteria. Any information or functionality that is provided in a way that is not accessibility-supported must also be available in a way that is accessibility-supported (Conformance Requirement 5.2.4). Using a technology in a way that is accessibility-supported means that the technologies will work with user agents and assistive technologies. When new technologies are introduced, they may not immediately be accessibility supported since the user agents and assistive technologies may need to be redesigned or modified to be able to actually work with these new technologies.

In some cases, it can be appropriate to implement workarounds, compromises, or polyfills, to deal with lack of screen reader support or bugs but recognise that screen reader behaviors can change at any time. Polyfills should be used with caution as they have been the source of security concerns.

1. Screen Readers

While both laptop-based and mobile screen readers serve the same fundamental purpose, there are key differences in how they operate and user interaction. Laptop-based screen readers, such as JAWS and NVDA, typically offer more robust features, allowing for advanced keyboard shortcuts and extensive customization, which can be particularly useful for complex tasks like coding or document editing. In contrast, mobile screen readers, like VoiceOver on iOS and TalkBack on Android, rely heavily on touch gestures, making them optimized for navigation and interaction on smaller screens. Mobile screen readers are designed to integrate smoothly with apps and on-the-go tasks, making them more intuitive for quick actions but less feature-rich than their desktop counterparts.

Screen readers function by converting the text and elements on a screen into audible speech or Braille, enabling people with vision loss to interact with digital content. They work by interpreting the structure of websites, applications, and documents, allowing users to navigate through various sections like headings, links, buttons, and forms. Users can move sequentially through content or jump directly to specific elements, such as headings or links, using keyboard commands or gestures. Screen readers also announce critical information such as formatting, images (if alt text is provided), and system alerts. They help users complete forms, read tables, and interact with dynamic content like pop-up windows or drop-down menus.

Resources

- Using ARIA (Aria-live)
- Definition of User Agent
- VoiceOver for Mac
- VoiceOver gestures on iOS
- Deque University: Talkback Gestures on Android
- NVDA Quick Start Guide
- JAWS Quick Start and Keystrokes

Domain One F: Standard controls vs. custom controls

(e.g., using standard controls when possible, if using custom controls build them using WAI-ARIA best practices).

Study Tasks

Become familiar with the keyboard interaction model for ARIA custom widgets. There are general keyboard patterns, plus keyboard patterns specific to types of widgets.

- In many widgets, the keyboard interaction model is to tab to the widget as a whole, or the active/selected element within the widget, then use the arrow keys to navigate within the widget. The tab key is generally not used for navigation within the widget. Other keystrokes may be recommended for certain widgets.
- Native HTML widgets should be used instead of custom WAI-ARIA widgets whenever possible, because of the built-in accessibility features of native HTML widgets. Implementing custom widgets requires greater attention to detail in the coding techniques and patterns, and support for custom widgets may vary, especially for less common widgets.
- When creating ARIA widgets, pay attention to the semantic structure of the roles. Some roles have required parent or child roles, and some roles have required attributes.
- When a custom role is assigned to an element, the custom role completely overrides the native role. For example, `<li role="button">` will be treated as a button by the accessibility API, not like a list item.
- Roles to describe the type of widget presented, such as “menu”, “treeitem”, “slider”, and “progressbar”. • Roles to describe the structure of the Web page that include headings, regions, and tables.
- Properties to describe the state widgets are in, such as “checked” for a checkbox, or “expanded” for a menu.
- Properties to define live regions of a page that are likely to get updates (such as stock quotes), as well as an interruption policy for those updates—for example, critical updates may be indicated in an alert dialog box, and incidental updates occur within the page.
- In the case of complex web applications or dynamic contents, ARIA roles, states, and properties may be used that communicate to the screen reader with what is occurring in the interface.

- The application role should be used sparingly, if at all, because it overrides many assistive technology keystrokes, such as the keystrokes that allow screen reader users to navigate by semantic elements such as headings, landmarks, tables, etc.

Overview

When it comes to creating accessible web interfaces, the decision between using standard controls and custom controls is crucial. This choice can significantly impact the user experience, especially for people with disabilities.

1. Standard Controls

Standard controls are designed with accessibility in mind. They come with built-in support for assistive technologies (AT) like screen readers, voice recognition software, and keyboard navigation. This means that when you use standard controls, you're leveraging a foundation that already meets many WCAG success criteria.

Consider the case of an e-commerce website. If you're building a form for users to enter their shipping information, using standard HTML form elements is usually the best choice. These controls are familiar to users and work consistently across different platforms and browsers. The accessibility support embedded in these controls ensures that all users, including those relying on assistive technologies, can interact with the form quickly and efficiently.

Additionally, standard controls save time and money. They reduce the need for extensive testing and validation because they have been pre-built and thoroughly tested by the platform providers. These controls are also regularly updated to align with the latest accessibility standards and practices, easing maintenance burden and ensuring long-term compliance.

2.Custom Controls

However, there are situations where custom controls become necessary. Imagine you're designing a sophisticated online music player with unique interaction features that standard controls can't support. In such cases, custom controls can offer the flexibility to create highly specialised user interfaces.

Custom controls may also be necessary when branding and design requirements are specific and stringent. If your project demands a particular design language or branding that standard controls cannot accommodate, custom controls allow you to create a UI to meet these needs.

While custom controls can enhance user experience with advanced interactions, ensuring they are accessible is paramount. This means implementing proper ARIA roles, states, and properties

and conducting thorough accessibility testing with real users, including those who rely on assistive technologies.

Whether you choose standard or custom controls, thorough testing is essential. Always test with real users, including those with disabilities, to ensure accessibility. Document custom controls meticulously to guide other developers in implementing and using them accessibly.

In conclusion, the decision between standard and custom controls hinges on specific project requirements and the need to ensure accessibility. Standard controls offer a reliable, accessible, and cost-effective solution for many use cases. On the other hand, custom controls provide the flexibility to meet specialised requirements, provided they are designed and tested with accessibility in mind. By carefully considering these factors, you can create an inclusive user experience.

Resources

- Deque University's Code Library of Accessibility Examples: ARIA Widgets
- W3C's WAI-ARIA Authoring Practices 1.1

Domain One G: Single page applications

(e.g., focus control, delays for AJAX-Screen Reader compatibility, live announcements).

Study Tasks

- If the AJAX content is loaded as the direct result of a user action (e.g., activating a button), the screen reader user should be notified that new content has loaded. Methods that can be used to notify screen readers that new content has loaded include:
    - Sending the focus to the new content
    - Using aria-live to announce content without moving the focus
- If the AJAX content is loaded passively (i.e., not as the direct result of a user action like activating a button), users may not need to be notified that the new content has loaded. This action may be dependent on the importance and urgency of the new content, and whether the content has been inserted above the user's current position or not.

Overview

Single-page applications that present new content – often through asynchronous JavaScript-based processes, without reloading the page present a unique set of accessibility challenges.

Screen readers typically react to a normal page load event by reading the page title and (in the case of some screen readers) reading a summary of available semantic elements (e.g., “page has 7 headings, 3 landmarks, and 27 links”). Single-page applications do not fire normal page load events. Single-page applications use AJAX to pull in content to the current page/URL, rather than load a new page/URL, usually to improve performance and streamline the web application's user experience. Screen readers typically do not announce anything when content is loaded via AJAX, so users may be unaware that new content has been loaded.

If the AJAX content is loaded as the direct result of a user action (e.g., activating a button), the screen reader user should be notified that new content has loaded. Recommendations for Single Page Applications include:

- Update the page title via JavaScript (document.title)
- Send the focus to the new content
- Using aria-live to announce content without moving the focus (e.g., providing a status message, such as “New Page: [Name of new page])

If the AJAX content is loaded passively (i.e., not as the direct result of a user action like activating a button), users may not need to be notified that the new content has loaded. This action may depend on the importance and urgency of the new content and whether the content has been inserted above or below the user’s current position.

Resources

- Deque’s Dynamic Updates, AJAX, and Single-Page Apps

Domain One H: Strategies of persons with disabilities in using web solutions

(e.g., navigation of screen reader users, headings and landmarks, coping strategies, user preferred methods vs. website specific methods, using keyboard vs. mouse).

Study Tasks

- Understand the accessible design principles and how they relate to user needs.
- Define the terms “assistive technology” and “adaptive strategies”.
- List different types of assistive technologies and adaptive strategies.
- List types of disabilities and related accessibility needs.
- Understand the impact of design decisions on accessibility for people.
- Be familiar with different types of assistive technologies and how they work.
- Identify most common screen readers and recommended combinations to ensure better compatibility.
- Be familiar with navigation strategies using a screen reader.
- Understand the modes of interaction across screen readers and their impact on browsing with the assistive technology.
- Know the consequences of using contrast enhancement modes such as Windows High Contrast Mode.

Overview

People with disabilities often use different methods to access web content than the general population. Some people rely on **adaptive strategies**, which mean configuring standard software and hardware according to a person’s needs, such as increasing text size, changing colour and background colour, reducing mouse speed, and turning on captions.

Others may use **assistive technologies,** such as screen readers, magnification software, etc., or just alternative input devices such as keyboard or voice input. Many kinds of assistive technologies are designed for many different kinds of disabilities. For example, most people who are blind use a screen reader that allows them to listen to web content and navigate through it according to the semantic markup (e.g., landmarks, headings, tables, etc.).

Standard computer hardware, mobile devices, operating systems, and web browsers have built-in features for customizing the computer, accessibility features, and solutions.

Creating accessible content benefits from knowing about different strategies; and how different concepts, designs, and implementations affect those strategies. The following is a summary of some highlights of the strategies and technologies used by people with various disabilities (this list is not all-inclusive).

Resources

- How People with Disabilities Use the Web, WAI W3C
- Diverse Abilities and Barriers, WAI W3C
- Tools and Techniques, WAI W3C
- Web Accessibility Perspectives Videos, WAI W3C
- Browsing with assistive technology, TetraLogical, 2021
- Dyslexia friendly style guide, The British Dyslexia Association, 2023
- Cognitive Accessibility at W3C, WAI W3C
- All Supplemental Guidance, WAI W3C
- Making Content Usable for People with Cognitive and Learning Disabilities, W3C Working Group Note
- Keyboard Accessibility, WebAIM, 2022
- Screen Reader User Survey #9 Results, WebAIM 2021
- Designing and Coding for Voice by Brian DeConinck, 2023

1. Users without Vision

Blind users need a screen reader to explore and interact with digital content. A screen reader is a software that processes content on the desktop and in web browsers and converts it to other forms, such as text-to-speech and braille, on a refreshable Braille display. A refreshable Braille display is an external device that is made up of rows of pins, each of which can raise or lower to form a Braille character. Since the user cannot see the interface, the screen reader needs to provide the user with mechanisms to understand the structure of the interface and the meaning of the controls.

1.1. Screen Reader Software

When content is coded correctly, screen readers will read everything on a web page, including the semantic information. Using keyboard commands, screen reader users can explore, navigate, and interact with content.

Among screen reader users, knowledge and strategies vary considerably. A common strategy is to use headings or landmark regions to scan a page and then use other keyboard commands to explore relevant content in more detail. Some users only use a few keystrokes, such as the arrow keys to navigate from item to item down the page and the tab key to go to focusable items.

A common approach using a screen reader is to use special commands to call up lists of headings, links, form controls, etc. This way, users can quickly move between such elements on the page. For this to work, it is important to use correct semantic markup. Standard HTML elements are well supported. Most screen readers also support significant parts of WAI-ARIA, but you need to know if the markup you plan to use is sufficiently supported by screen readers and browsers.

In contrast to how most people visually use an interface, screen readers present content linearly to users. The reading order follows the order of the elements in the DOM. In addition to reading the page content in a linear order, screen readers also offer modes for users to interact with specific types of webpage content. These vary across screen readers (read/browse mode, forms mode, application mode, VoiceOver rotor and TalkBack menu).

The interaction model for blind touchscreen users differs from that for sighted touchscreen users. Sighted users explore content and activate items based on their position on the screen by swiping or touching the items directly. Blind users, on the other hand, have a completely different set of gestures available to them when a screen reader is activated on a touch device. The screen reader overrides the visual interaction method and replaces it with a gesture-based system. There are screen reader gestures for going forward through content (swipe right), going backward (swipe left), activating the current button or link (double-tap the screen anywhere), and so on. The specific gestures vary from one brand of screen reader to the next.

1.2 Some Concepts That Are Important to Understand

- Unlike how most people visually use an interface, screen readers present content linearly to users. The reading order follows the order of the elements in the DOM.
- By default, the focus order is set by the order of the elements in the DOM.

- Screen readers can use the Accessibility API to retrieve information from the Accessibility Tree. Semantic elements in the DOM affect information available in the Accessibility Tree.
- Screen reader navigation depends on providing semantic elements that convey the page's structure and controls that ensure keyboard support.
- Landmarks help screen reader users orient themselves to a page and navigate easily to a page's main sections.
- Screen reader users may also use voice input tools to navigate as many have or develop physical weaknesses associated with repetitive physical functions such as carpal tunnel and other limitations. Voice input is often quicker for them, especially compared to using virtual keyboard input.

Note: The results of screen reader output may vary due to the various types of browsers and screen readers (and different versions of these).

For further information, see the following resources:

- JAWS Documentation
- NVDA User Guide
- Mac OS VoiceOver User Guide
- ChromeVox User Guide
- Android Talkback User Guide
- iOS VoiceOver User Guide

2. Users with Low Vision

Users with low vision may also use a screen reader to get text read aloud, but they often use the vision they have to find their way around the interface. There are several strategies for users with low vision:

- Using text enlargement and zoom in the browser
- Changing colours in the browser or operating system
- Using magnifying tools

2.1 Some Concepts That Are Important to Understand

- It is quite common for magnification users to use a mouse to navigate, but they also use tabbing or keyboard commands to speed up the interaction.
- Users may alter colours, contrasts, fonts, focus indicators, and cursors.
- Users often combine zooming and screen reading.
- Users engage the screen reader temporarily to read text that runs beyond the magnified area.
- Users engage the magnification software's screen reader or the operating system's voice output to read longer text that would require a lot of scrolling in the magnified area.
- There is a significant difference between users who only need the built-in moderate zoom in the browser and those who use high zoom levels screen readers as a complementary technique. Their strategies vary accordingly.

2.2 Magnification Software

Magnification software presents the information in a way suitable for people with low vision. It enlarges everything visible on the screen, including the operating system, applications, and content.

Magnification software can enlarge the information displayed up to 60 times the original size (6000%) and provide special colour and contrast settings (for example, inverting colours or customizing the mouse pointer or insertion mark). Users of screen magnification use a mouse, touchpad, keyboard or other pointer device.

When navigating with magnification software, orientation on the page can be challenging for users, especially when the level of magnification is high. A common way of dealing with that is to zoom out and in again, depending on whether users want to get an overview of the page and identify their location or explore content in-depth. Moreover, focus tracking helps with orientation. This means that when the pointer or keyboard focus moves, the visible portion of the content moves with it. If additional content becomes visible on hover or keyboard focus (e.g., custom tooltips or submenus), this content should be dismissible since obscures content underneath. It must be possible to move the mouse pointer over the additionally displayed content without it disappearing so that it can be read and interacted with.

Usually, the magnification software can be combined with a screen reader, which is often already integrated. In some cases, a Braille display may be connected as a third output medium.

The combination of different output media relieves the eyes and ensures the availability of information that is otherwise not visible in the enlarged screen display. People with cognitive or learning disabilities may also use screen magnification to help them read or focus on their tasks.

3.3 Common Magnification Software

Screen magnification is built-in on all popular platforms, such as Windows, macOS, iOS and Android.

- Magnifier (Windows)
- Vision Accessibility features (Mac / iPhone)
- Magnification (Android)
- Font size and display size (Android)

Magnification and Zoom increase everything on the screen. iOS and Android also support pinch zoom and the ability to magnify parts of the screen.

If magnification functions are provided as an accessibility feature of Operating Systems, the level of magnification and the feature set are limited. Therefore, people with low vision may use a magnification tool independent of the operating system.

Windows:

- SuperNova
- ZoomText
- Fusion (combination of the JAWS screen reader and screen magnification)

3. Users with Limited Reading Capacities

There is a wide spectrum of people having reading difficulties. These may include, but are not limited to, people with ADHD, dyslexia, Irlen syndrome, or memory loss. This group may use assistive technology such as “Kurzweil” (educational software designed to provide literacy support) or text-to-speech provided by the platform, or they may not use any assistive technology.

Some users may instead seek information that is not text. They may prefer alternative formats, such as audio, video, or slide-based presentations with visual content, wherever these are available.

To read text, some users make changes to the presentation in the browser. You should know how this affects the interface and what challenges might arise from making such changes. Users may choose to change:

- Text size
- Text colour and background colour
- Font
- Spacing
- Line width

Users might even define a customised style sheet to be used.

There are some assistive technologies available to this group of users. They are based on either speech output or on altering the visual presentation of text.

3.1 Some Concepts That Are Important to Understand

- It is crucial that text is offered as real text and not as images of text. Otherwise, it becomes difficult or impossible for assistive technology and browser settings to alter the text.
- There are recommendations for fonts that offer good readability and best practices regarding minimal font size and letter and word spacing widths.
- Structuring text with section headings helps users navigate through content.
- The semantic markup of text is important to support assistive technology users.

3.2 Text-To-Speech Software

Text-to-speech software (TTS) is a commonly used assistive technology for reading. When using TTS, you can see and hear text read aloud simultaneously. Words clicked on or highlighted are read by a computer-generated voice. This helps people focus their energy on comprehending the text instead of decoding the words.

The software is focused on text content and does not offer additional semantic information or navigation hints (as screen readers used by blind persons do) since most individuals in this user group do not need that level of support.

3.3 Common Assistive Technology for Reading

Desktop and laptop computers have built-in text-to-speech software. Browsers also provide extensions for read aloud functionality.

There is software that adds reading functions beyond built-in browser or operating system functionality, for example:

- read&write by Texthelp
- BrowseAloud by Texthelp
- NaturalReader
- Kurzweil

Mobile devices (like tablets and smartphones) also have built-in assistive technology, such as text-to-speech and dictation technology. Various reading tools (apps) are also available for mobile devices.

4. Users with Cognitive Disabilities

This user group is also very diverse. The group includes users that have concentration problems, users with neuropsychiatric or neurological disorders as well as behavioral and mental health disorders. Autistic people, attention deficit (hyperactivity) disorder (AD(H)D), dyslexia, and people with other diagnoses may identify as neurodivergent. They may experience specific barriers when using the web but don't necessarily identify as having a disability.

Because cognitive and neurological disabilities may affect any part of the nervous system, they can affect a person's ability to hear, move, see, speak, understand, and remember information.

Some users with intellectual disabilities cannot use the web content even with assistive technology. However, most users can use the interface even if it might require some alterations or assistive technology.

A clean, simple layout and presentation are important to aid this group in understanding content. Provide control of as many aspects of the website as possible. Using semantic markup and CSS for presentation allows for different types of adaptation, for example, via browser settings or plugins, style sheets, or other assistive technologies.

4.1 Some Concepts That Are Important to Understand

- Plain simple language is critical for this group.
- People differ in their ability to process information in different forms. Providing content alternatives (text, image, audio) helps to reach more people.
- Using images and multimedia to supplement text helps users who prefer text alternatives.
- Appropriate and clear graphics can facilitate understanding of content. However, the overuse of graphics and animated graphics can be distracting and increase cognitive load.
- Users with cognitive disabilities often use assistive technology to read and write.
- Predictable and consistent web design, appearance, operation, and navigation help users with cognitive disabilities successfully complete their tasks and stay focused.
- Unexpected and novel designs or navigation structures can be confusing for people with visual, cognitive, or motor disabilities.

4.2 Assistive Technology for People with Cognitive Disabilities

People with limited cognition often use the same assistive technologies as users with reading difficulties (e.g., adapting the presentation of content according to their individual needs and preferences, using text-to-speech software). They might also use ad blockers, screen masking, and other such tools to help them focus on the present task.

There are also assistive technologies to help users with cognitive impairments to add and edit data on webpages, such as filling out form fields.

5. Users with Motor Disabilities

Users with motor disabilities are a broad group. Some are entirely or partly paralyzed; others have fine motoric dysfunctions, for example, due to pain, tremors, or rheumatism. Motor disabilities also include limitations of sensation, joint disorders (for example, arthritis), and limb differences.

Depending on their needs and preferences, people with motor disabilities use specialised hardware and software (mouse / mouse-like devices, keyboard / keyboard-like devices, or voice control).

People who depend on the keyboard instead of the mouse often navigate sequentially through the focusable page elements, such as links, buttons, and form elements. To be able to do this predictably, the order in which interactive items receive keyboard focus must be logical and intuitive. This generally means that it follows the visual order on the page. By default, the focus order is set by the order of the elements in the DOM. This means that the order in which HTML elements appear in the DOM affects the focus order. Focus order problems often occur when interactive elements are inserted into the DOM, for example, appended at the end of the page. Furthermore, it is essential to provide a clearly visible focus indicator for sighted keyboard users.

Standard keystrokes are used to navigate between native HTML elements, such as Tab to move to the next focusable element and Shift + Tab to move to the previous focusable element. Enter activates links, and the Enter or Space keys activate buttons.

Unlike using a screen reader, keyboard-only users cannot jump back and forth between headings, lists, links, or form fields. Navigating through a page or complex navigation structure with many links can be very tedious, so it is best practice to provide a “skip to main content” link on the page.

Complex widgets derive keyboard conventions from the GUI components of standard desktop GUIs. Often, the arrow keys are used to navigate within the component, for example, when traversing a tab panel (See Domain One D3 Keyboard interaction and focus management within components).

5.1 Some Concepts That Are Important to Understand

Much like blind users, users with motor disabilities often use the keyboard or depend on keyboard functionality, but with some important differences:

- Users must be able to see where the focus is; what link, form control, or button has focus when the user navigates down the page with a keyboard since they do not use a screen reader that announces the element focused.
- Because they don’t have a screen reader, users cannot call up lists of headings, links, and other items, they cannot navigate a page as quickly.
- For various user groups, full keyboard navigation and operation (generally with tab, arrow keys, enter, and space bar) and an intuitive focus order are critical. This includes people who cannot use a mouse and people with motor disabilities who depend on assistive technologies (like switches) that simulate sequential movement with tab keys.

- To have large, clearly visible, and easy to spot clickable areas are of key importance for many users.
- When avoiding duplication of link names, elements can be addressed unambiguously by speech input
- Prediction and autofill are very helpful as it limits the number of keystrokes required when typing.
- Users have different input devices so it must be possible to use web content with different input devices and methods.

5.2 Pointer-Based Software and Keyboard-like Device

There are a variety of input methods for people with motor disabilities. The most common include:

- Ergonomic or specially designed keyboard or mouse
- Head pointer, mouth stick
- Sip-and-puff mouse operated with the mouth
- Voice-controlled mouse grid to home in on elements on the screen
- On-screen keyboard with trackball, joysticks, or other pointing devices
- Switches operated by foot, shoulder, sip-and-puff, or other movements
- Voice recognition, eye tracking, and other approaches for hands-free interaction

5.3 Common Point and Click-Based Software

- Ordinary mouse
- SteadyMouse (tremor mouse accessibility software)
- EyeGaze (eye-tracking)
- Tobii (eye-tracking)
- Switch Control on iPhone, iPad and iPod Touch (point scanning with a switch control)
- Switch Control (Android)
- Mac OS Headpointer Feature (Head pointer)

5.4 Speech Recognition Software

People with physical disabilities who cannot use the keyboard or mouse as their primary input method use speech recognition. People with cognitive disabilities also benefit from this technology because they can speak rather than type.

Speech recognition software transcribes human speech into text or actionable commands. You use spoken commands to navigate and interact with web content, such as navigating and activating controls, dictating text, and filling out form fields. You can interact with the browser or the entire operating system.

The commands used to navigate with speech recognition are often based on keyboard operation. For example, instead of selecting TAB on a keyboard, you say “Tab” to move the focus to the next control and "Shift-Tab" to move to the previous item. To activate a link or button, you say “Click” along with the text used in the link or button, for example, “Click home”.

If there are multiple links on the same page with the same link text, a voice command can display a number next to all links in the page, so that the right one can be activated by saying, for example, "click seven".

If controls don’t have labels, users can activate a grid overlay by saying "MouseGrid". They can then repeatedly focus on numbered boxes of the grid until they have homed in on the element they want to activate.

5.5 Common Speech Recognition Software

Most computers and mobile devices today have built-in speech recognition functionality.

- Windows Speech Recognition for Windows
- MacOS: Voice Control
- iOS: Voice Control
- Android: Voice Access

While computers and mobile devices offer built-in speech recognition, standalone software offers better support. The most popular software available is Dragon:

- Desktop: Dragon
- iOS and Android: Dragon Anywhere

6. Users with Auditory Disabilities

People with auditory disabilities do not use special assistive technology. Nevertheless, they have needs that should be considered. People with auditory disabilities may prefer different forms of communication:

An auditory disability acquired **before the completion of language acquisition** is referred to as **early hearing loss or deafness.** Deaf people often prefer to use sign language. There are several national sign languages and "International Sign". For people who use sign language, written language (and thus also texts on websites or in apps) can be more difficult to understand, similar to a foreign language. For this reason, sign language videos (a WCAG 2.1 AAA Success Criterion) are helpful for them.

There are some attempts to create assistive technology that translates written text to sign language. Although this is not yet a common practice, it is preferable to use an easy-to-read language to help this type of assistive technology and readers without assistive technology. Icons, illustrations, and images should also be used to enhance the information.

An auditory disability acquired **after the completion of language acquisition** is called **late hearing loss or deafness.** These people prefer to use written language information even if they have already developed hearing loss. Alternatives for auditory information, such as transcripts for audio files and accurate captioning video content is very important. The use of only automated tools and those relying only on artificial intelligence creates many opportunities for an incorrect translation of spoken content.

6.1 Some Concepts That Are Important to Understand

- For pre-recorded audio-only, persons with auditory disabilities need an alternative that presents equivalent information for such as a transcript. Transcripts can be available on the same screen where the audio is, e.g., by using a disclosure (show/hide) functionality.
- For pre-recorded (or live) video, persons with auditory disabilities need captions. Closed captions (that can be hidden or shown by users) are preferable to open captions as they are customizable.
- For pre-recorded video where the sound doesn't always convey what is going on visually, persons with visual disabilities need an audio described version.

Domain Two: Identify accessibility issues in web solutions -- (40%).

Domain Two A: Interoperability and compatibility issues

(e.g., works with JAWS, Chrome, Safari, etc…)

Study Tasks

- Understand the needs of users of different types of disabilities
- Identify issues for the methods, technologies or strategies used by people with disabilities

Overview

When evaluating web content for accessibility flaws, it's important to identify any aspects of the design that can cause problems for the methods, technologies, or strategies used by people with disabilities (see Domain One H). Be aware that some requirements affect multiple user groups. For example, keyboard support is a requirement for different types of assistive technologies. And well-structured content helps all users, even if not highlighted for each of the following user groups.

In clause 4 Functional Performance and in Annex B, the EN 301 459 lists the following distinct user requirement categories, here called "Functional Performance statements":

- Usage without vision
- Usage with limited vision
- Usage without perception of colour
- Usage without hearing
- Usage with limited hearing
- Usage without vocal capability
- Usage with limited manipulation or strength
- Usage with limited reach
- Minimise photosensitive seizure triggers

- Usage with limited cognition
- Privacy

These categories are not meant to portray distinct user groups, but specific functional requirements. Some users will require several of these at the same time. A sighted user with limited vision using a screen reader in addition will simultaneously require zoom and good contrast and the programmatic access encapsulated in "Usage without vision". Other users will simultaneously require usage with limited hearing and with limited cognition.

The mapping of accessibility needs of different user groups to specific accessibility requirements in Domain Two C1 to 6 below has been informed by EN 301 549, Annex B, but does not follow the exact same structure.

Not all combinations of screen readers, browsers, and operating systems work equally well in terms of accessibility support. It is recommended to perform tests with at least two screen readers. Among the most used are:

- JAWS
- NVDA
- VoiceOver

Recommended Combinations to Guarantee Better Compatibility

- Windows:
  - JAWS and Google Chrome
  - NVDA and Firefox or NVDA and Chrome
  - Narrator and Edge
- MacOS
  - VoiceOver and Safari
- iOS:
  - VoiceOver and Safari

- Android:
    - TalkBack and Chrome

Resources

- Diverse Abilities and Barriers, WAI W3C
- Tools and Techniques, WAI W3C
- EN 301 539, Annex B (informative): Relationship between requirements and functional performance statements (PDF)
- Windows:
    - JAWS for Windows by Freedom Scientific
    - NVDA for Windows by NV
    - Narrator for Windows by Microsoft
- MacOS
    - VoiceOver for MacOS by Apple
- Chromebooks:
    - ChromeVox for ChromeOS
- iOS:
    - VoiceOver
- Android:
    - TalkBack

Domain Two B: Identifying guidelines and principles regarding issues

(including WCAG 2.2, WAI-ARIA, ATAG, basic concepts, limitations of the specific guidelines, principles, and techniques, what is normative vs. non-normative; what is included in different levels (A, AA, AAA)).

Study Tasks

- Become familiar with the specifications (WCAG, WAI-ARIA, and ATAG), and know which Success Criteria apply to which conformance level.
- Understand the WCAG conformance requirements.
- Understand the evaluation process of the WCAG-EM methodology.
- Be able to distinguish between failures of accessibility criteria versus other bad accessibility practices that are not covered in the specifications.

Overview

In order to determine conformance, it is first of all important to specify the guidelines or standards applied and to understand the Success Criteria thoroughly. When assessing the website, distinguish between true accessibility failures with respect to the normative standard, and poor design decisions that do not fail the applicable standard requirements (even if they may fail a different standard or a different level within the standard, or are considered bad practice).

1. Accessibility Target Level

Local or national laws, customers, internal organization policies or overall accessibility goals will define which guidelines to follow, and what conformance level to meet. If your project is not subject to specific regional, national or supra-national legislation or policies, the recommendation is to follow the latest version of WCAG. In some jurisdictions, such as in Member States of the European Union, standards like EN 301 549 that contain requirements beyond WCAG may be applicable.

WCAG follows a research-focused, user-centered design methodology to produce the most effective and flexible outcome, including the roles of content authoring, user agent support, and authoring tool support.

WCAG 2.1 has 12 more Success Criteria than WCAG 2.0 at Level A and Level AA. They relate to cognitive disabilities, touch interface devices, and screen orientation. WCAG 2.2 added 9 new Success Criteria. 2 at level A, 4 at level AA, and 3 at level AAA.

EN 301 549, while referring directly to the WCAG Success Criteria in its clauses 9 (Web), 10 (Documents) and 11 (Software), contains 38 additional requirements applicable to web content (see Annex A, Table A.1), which together provide the normative requirements to guarantee conformance with the Web Accessibility Directive 2016/2102 and other directives, like the European Accessibility Act.

2.Conformance Requirements

Conformance to a standard means satisfying all requirements specified by the standard. In order to test content against the WCAG requirements (Success Criteria), one can use a combination of automated testing and human evaluation. WCAG provides in its normative document five general conformance requirements that content must meet to achieve conformance to WCAG.

It is important to understand that WCAG conformance is only achieved when content meets all the Success Criteria. The conformance model is based on a pass / fail approach: there must not be any content that violates any one of the Success Criteria. In practice, a conformance test typically includes only a representative sample of pages. Not all pages are tested. A strict conformance claim is tied to individual pages that have been shown to conform, not to the overall site.

The following gives a brief summary of the Conformance Requirements. For more details, refer to Understanding Conformance.

1. Conformance level: A web page is considered conformant to a targeted conformance level if all Success Criteria of this conformance level are satisfied or a conforming alternate version is provided. For example, to conform to level AA the page must meet all Success Criteria on level A and AA. If only one single requirement fails, the page can no longer be declared conformant.
   If you want to achieve conformance providing a conforming alternative version, you must ensure the following. (see conformance alternate version in detail):

   - The conforming alternate version meets all requirements at the designated level.
   - The conforming alternate version is equivalent to the non-accessible version in terms of content and functions, and is up-to-date.
   - The conforming alternate version can be accessed in an accessible way.

2. Full pages: Conformance refers to full web pages only. It cannot be achieved if parts of page are excluded.
3. Complete processes: All web pages in a process (e.g. a series of steps to complete a task) have to conform at the specified level.
4. Only accessibility-supported ways of using technologies: Only Techniques are used that is supported by assistive technologies. If any information or functionality is provided in a way that is not accessibility supported, it must also be available in a way that is accessibility supported.
5. Non-interference: If technologies are used in a way that is not accessibility supported, or if they are used in a non-conforming way, then these technologies must not block users from accessing the rest of the page. For this, you have to meet the following Success Criteria for the whole page including the non-conforming content:

    - 1.4.2 Audio Control
    - 2.1.2 No Keyboard Trap
    - 2.3.1 Three Flashes or Below Threshold, and
    - 2.2.2 Pause, Stop, Hide

Website Accessibility Conformance Evaluation Methodology (WCAG-EM) describes the process of selecting a representative page sample in order to be able to draw conclusions regarding the accessibility of the website, but still conformance can only be declared for each individual page.

3.WCAG-EM

Conformance tests measure the accessibility of websites or applications against a specified accessibility standard and on a given date. W3C provides an Accessibility Conformance Evaluation Methodology (WCAG-EM) for determining conformance to WCAG. The methodology defines a procedure for auditing websites which focuses on establishing the scope of evaluation (e.g. what can or cannot be excluded) and the process of creating a representative sample. It does not provide any details on the actual evaluation of content against the Success Criteria. To ensure reliable test results, auditors need a profound knowledge of how to interpret WCAG

Success Criteria and assistive technologies, and how people with different disabilities use the Web.

The conformance evaluation procedure is detailed under 5 main steps:

- **Define the scope of the evaluation** - what is included in the evaluation; the goal of the evaluation; and the WCAG conformance level (A, AA, AAA).
- **Explore the website** - identifying key web pages; key functionality; types of web content, designs, functionality, etc.; required web technologies.
- **Select a representative sample** - guidance on structured and randomly selected web pages when it is not feasible to evaluate every web page on a website.
- **Evaluate the selected sample** - determining successes and failures in meeting WCAG; accessibility support for website features; and recording evaluation steps.
- **Report the evaluation findings** - aggregating and reporting evaluation findings; making evaluation statements; and calculating overall scores.

You can generate a structured report from your input using the WCAG-EM Report Tool. Besides creating a report it assists you on following the steps of the evaluation process.

4.Assigning Success Criteria and Failures

Often the main goal of an accessibility evaluation is to determine if the website or web-app conforms to a particular standard or not. These audits and assessments may also have a legal dimension. When conducting a conformance test, you should follow the W3C's guidance and respect normative requirements.

It is important to have a profound understanding of the Success Criteria. Auditors should be able to map each accessibility issue to one of the Success Criteria within that standard and interpret WCAG correctly and accurately. Knowing the boundaries and demarcations of the criteria is not always easy, though WCAG is built on the idea that Success Criteria can be evaluated clearly, unambiguously and consistently. Requirements that are misunderstood or misinterpreted lead to wrong, or at least inconsistent, error reporting.

For example, 2.4.6 mandates if a page uses headings and labels, they must be descriptive. 1.3.1 Info and Relationships mandates that headings and labels be correctly marked-up, and if labels are not provided it's a problem for 3.3.2 labels or instructions.

5.Accessibility Guideline Limitations

While WCAG 2.2 offers extensive guidance to improve web accessibility, there are still areas where additional success criteria could enhance inclusivity further. Here are some of the gaps:

1. **Pseudomotion**: The existing WCAG guidelines primarily focus on reducing motion, but they do not account for pseudomotion-visual effects that mimic motion. This would include graphics like optical illusion. These effects can be particularly damaging to users with vestibular disorders, migraines, or epilepsy.
2. **Colour Contrast for Colour Blind Users:** While the WCAG guidelines cover colour contrast, they do not specifically address to the needs of users with colour blindness. Additional criteria are necessary to ensure that content is distinguishable without relying solely on colour, by using patterns or textures to differentiate elements.
3. **Colour Contrast and Font Weight/Size**: WCAG addresses colour contrast but does not address the impact font weight has on the contrast. Font size is addressed broadly and could improve with a more granular analysis.
4. **Ornate Fonts**: Overly ornate fonts can be challenging for users with dyslexia or visual impairments. Criteria could recommend the use of simple, readable fonts and provide options to switch to more accessible typography,
5. **Cognitive Load:** While there are some guidelines for cognitive accessibility, more comprehensive criteria could help reduce cognitive load by simplifying navigation, avoiding complex language, and limiting information density on pages,
6. **Personalization/Customization/Theming**: WCAG could benefit from criteria supporting personalised and customizable user interfaces, allowing users to adjust settings like font size, colour schemes, and layout to suit their individual needs without relying on browser extensions or assistive technologies.

    A. Personalization ensures that content is presented in the most accessible way for each user, enhancing their ability to navigate and interact with web content.

B. Customization goes hand-in-hand with personalization. While personalization typically refers to system-driven adjustments based on user preferences, customization allows users to modify the interface to meet their specific requirements manually.
C. Theming would provide default behavior associated with different types of disabilities. Some themes could include a colour blindness theme, which eliminates the use of red and green; a dyslexic which eliminates the use of justification and italics; or an autistic theme, which eliminates the use of neon colours and stops all motion. Users would be able to further customise their settings on top of the themes.

7. **Consistent Layout**: More specific criteria could enforce consistent layout and design patterns across different pages and sections of a website, aiding users with cognitive disabilities by reducing the learning curve and improving predictability,
8. **Voice Interaction**: As voice interaction becomes more prevalent, criteria could ensure voice commands are accessible, with clear instructions and alternatives for users with speech impairments or non-standard speech patterns.

Addressing these best practices would significantly enhance WCAG's ability to create a more inclusive and user-friendly web environment.

Domain Two C: Testing with assistive technologies

(e.g., navigation of screen reader users, headings and landmarks, screen magnifiers, high contrast, using keyboard vs. mouse).

1. Identify Issues for Screen reader users

Some highlights of the types of things to look for include:

- Alternatives exist for images and video
- Semantic markup conveys meaning commonly conveyed via visual presentation (heading, lists, tables, paragraphs etc.)
- The order of content presented to assistive technology is “meaningful” (and usually follows the visual order)
- No instructions are based on sensory characteristics alone (e.g., shape, colour, placement etc.)

- Automatically playing audio can be turned off
- Scrolling or moving content is avoided or can be stopped by the user
- Page titles describe topic or purpose
- There is a way to bypass repeated content (e.g., Skip links, landmarks, section headings)
- All functionality is operable via keyboard, there is no keyboard trap
- The focus order is logical, intuitive and predictable
- Link text is descriptive or can be determined together with its context
- The correct language of the page is set so that screen readers pronounce it properly
- Error messages can be programmatically identified, and they are described to the user in text
- The role, name and value of each interactive component, such as a button, an accordion, a tab panel, or a carousel, are available and updated correctly
- Dynamic status messages and updates are conveyed to screen readers using live regions

2.Identify Issues for Users who use a keyboard or alternative input device

Some highlights of the types of things to look for include:

- All functionality is operable via the keyboard, there is no keyboard trap
- There is a way to bypass repeated content (e.g., Skip links)
- The focus order is logical, intuitive and predictable
- The focus indicator is easily visible
- Functionality that uses multipoint or path-based gestures can be operated with a single pointer (e.g., by activating a button)
- Only up-events execute the function, or users can reverse or undo that function
- For functionality operated by device motion or user motion, there is an alternative user interface control (e.g., clicking a button).

- Motion activation can be disabled to prevent accidental activation due to tremors or spasms.
- Interactive elements are implemented using native controls or proper ARIA mark-up (so they can be numbered via speech command for activation).
- Character key shortcuts can be turned off, remapped or are only active on focus (to reduce accidental activation when using voice control.
- No audio, or audio can be turned off (voice control).
- The programmatic label matches the visual label (voice control).

3.Identify Issues for Users with Auditory Disabilities

Some highlights of the types of things to look for include:

- Videos have captions
- Captions are accurate (automatic captioning errors are corrected)
- Transcripts are provided for audio-only content

Testing for end-user impact (e.g., low vision, cognitive, mobile/touch).

4. Identify Issues for Users with Low Vision

Some highlights of the types of things to look for include:

- No restriction to a single display orientation (portrait or landscape)
- No use of colour alone to convey information
- Sufficient contrast of text
- Sufficient contrast of non-text elements (for example icons or borders of input fields)
- Text can be resized to 200%
- No loss of functionality when zooming up to 200%
- Use of real text instead of images of text
- No two-dimensional scrolling necessary, content adapts to browser or screen width
- No loss of content or functionality when users change text spacing

- Content appearing on hover and focus can be dismissed without changing focus, does not auto-close, and can be hovered over
- Scrolling or moving content is avoided or can be stopped by the user
- The focus order is logical, intuitive and predictable
- The focus indicator is easily visible
- Descriptive labels or instructions for user input elements are available

5.Identify Issues for Users with Cognitive or Learning Disabilities

Some highlights of the types of things to look for include:

- Input fields collecting information about the user are marked up with appropriate autocomplete attributes to help users fill in data
- Automatically playing audio can be turned off
- Session timeouts can be turned off or adjusted
- Scrolling or moving content is avoided or can be stopped by the user
- Page titles describe topic or purpose
- Link text is descriptive or can be determined together with its context
- More than one way to locate a page is available (e.g., navigation, search, sitemap etc.)
- Headings and labels are descriptive
- No user interface component causes an unexpected change of context when it receives focus (e.g. automatically submitting a form)
- No user interface component causes an unexpected change of context when the user changes its setting (e.g. when a checkbox is checked)
- Navigation mechanisms are positioned consistently within the website
- Components that have the same functionality are identified consistently within the website
- Error messages can be programmatically identified, the error is described to the user in text

- Descriptive labels or instructions for user input elements are available
- Error suggestions are available
- When handling legal and financial data, a mechanism for error prevention is available
- Use of a combination of icons and text

6. Identify Issues for Touch Users

Not all touch device owners use touch interaction. Other methods include voice commands or the use of coupled Bluetooth keyboards or switch devices.

Some highlights of the types of things to look for include:

- Large touch target size
- Adequate spacing between touch targets
- No restriction to a single display orientation, such as portrait or landscape
- Functionality that uses multipoint or path-based gestures can be operated with a single pointer (e.g., by activating a button)
- For a functionality operated by device motion or user motion, there is an alternative user interface control (e.g., clicking a button).
- The user must be able to disable the motion feature to prevent accidental activation due to tremors or spasms.
- All content or functionality available at desktop viewport widths or also available on small touch screens (possibly via a menu)
- Text can be resized to 200%
- No loss of functionality when zooming up to 200%

Domain Two D. Testing tools for the web

(both automated and manual tools, i.e., what they are and what are their limitations; e.g., unit testing, browser-based tools, spider tools, bookmarklet, automated tools used to monitor site vs. external tools).

Study Tasks

- Understand the purpose of ACT Rules.
- Know the strengths and limitations of automated testing tools.
- Differentiate between the kinds of accessibility issues that can be found with automated tools and issues that require manual testing.
- Understand how accessibility software tools can be used at various stages in the web development process (e.g., design/develop/test).
- Be familiar with the types of software tools available (site-wide scanning, server-based analysis, unit testing, integration testing, browser developer tools, browser add-ons, simulators, guided manual testing, etc.).
- Know how to use the keyboard and screen readers to navigate ARIA custom widgets.
- Be familiar with standard keystrokes for interaction with interactive elements in native HTML.
- Know how to use the browser's built in Development Tools to inspect source code.

Overview

There are different ways to test a website or web app for accessibility. One approach is to thoroughly assess it against an accessibility standard. You cannot do conformance testing with automated tools only, it will be a combination of both automated and manual testing. Auditors should have profound expertise across a variety of areas including:

- Understanding web technologies
- Understanding the Web Content Accessibility Guidelines and associated Techniques
- Understanding evaluation tools
- Combining automatic and manual approaches when evaluating web accessibility

- Interpreting the results of automatic testing, identifying false positives and omissions
- Being aware of disability barriers, assistive technologies, and adaptive strategies
- Involving of people with disabilities in evaluation

Resources

- ACT Rules Community, WAI W3C
- ACT Rules, WAI W3C

1.ACT Rules

The Accessibility Conformance Testing (ACT) Rules Community Group is a group of accessibility tool vendors, test procedure authors, and accessibility test experts. It aims to harmonise the interpretation of W3C accessibility standards, such as WCAG and WAI-ARIA for testing purposes and more consistent results and provides exact test cases. The ACT Task Force is a subset of Accessibility Guidelines Working Group (AGWG) participants and authors ACT Rules for publication.

The rules are being implemented by accessibility test tools and manual testing methodologies. That means an ACT Rules are primarily intended for people who design tests and methodologies and not for content creators, developers, or people using testing tools. The Community defines an ACT Rules Format describing the structure of ACT Rules.

Be aware that ACT Rules are not required for determining conformance. They are non-normative W3C resources. Conformance with WCAG must be determined based on the normative text of WCAG's Success Criteria.

ACT Rules check for failures. Content that contains failures does not meet the corresponding requirement. However, passing an ACT Rule does not necessarily mean that all aspects of the corresponding accessibility requirements are met. It only means that no error addressed by a particular rule has been identified. ACT Rules only describe how to test one particular aspect, whereas the WCAG Success Criteria usually cover multiple aspects. For example, "Form Field has non-empty accessible name" is an atomic rule to test 4.1.2 Name Role, Value. Verifying that content meets all aspects of the WCAG Success Criteria usually requires further verification by human testers.

2.Automated Testing Tools

No accessibility software tool can find all the accessibility issues on a website. Neither can automated testing tools determine conformance. However, software tools can help identify accessibility issues faster or assist you during the evaluation process, for example, by revealing semantic errors in complex code that are difficult to identify in manual inspection. Automatic tools support running tests repeatedly within the product lifecycle. They give quick results and can provide a measure of improvements over time. Be aware that such tools can include false positives, that is, an issue may be reported that on closer inspection is not a WCAG violation. Automatic tools analyze code but may not take into account the context of the page. Limits also exist when the page changes dynamically through user interaction.

The overall accuracy of automated testing should be increased by skilled manual evaluation of the same content. For example, an automatic tool may scan all the alternative texts of images on a page (to verify if all have alternative texts). Still, only a human can determine if the alternative texts are adequate for the images.

There are many tools available (some of which are listed below). When deciding if and when the use of automated tool is beneficial, include some considerations, for example:

- In which phase of the software development life cycle do you test your product?
- Should the tests be performed repeatedly?
- Who is conducting the test (designer, developer, content author, quality assurance, individual, team, company)?
- What is the level of expertise of the auditor? Automated findings need human interpretation to derive further actions.

Note: The following list of products is not an endorsement by the IAAP of any of the products or companies that produce these products. These are merely examples of various kinds of software for accessibility testing. For more options, refer to the W3C's Web Accessibility Tools List (the IAAP is not responsible for the content or accuracy of that list).

- Site-wide scanning and reporting
    - AMP (by Level Access)
    - Tenon.io (by Level Access)
    - WAVE API (by WebAIM)
    - WorldSpace Comply (by Deque Systems)

    - Sort site (by PowerMapper)
- Server-based page analysis
    - Accessibility IMpact (AIM) Report (by WebAIM)
    - SiteImprove Accessibility (by SiteImprove)
- Browser-based developer/QA tools (one page at a time)
    - AInspector for Firefox (by University of Illinois at Urbana-Champaign)
    - ANDI – Accessible Name and Description Inspector (by the Accessible Solutions
    - ARC Platform (by TPGi)
    - axe browser add-on (by Deque)
    - Branch of the United States Social Security Administration)
    - Google accessibility developer tools
    - Google Chrome Lighthouse
    - JavaScript bookmarklets written to test specific accessibility criteria or expose certain types of markup (by various companies and individuals)
    - MS Accessibility Insights
    - WAVE browser add-on (by WebAIM)
- Unit testing during development
    - axe API (with integration into Selenium, etc., by Deque)
    - Tenon API (by Level Access)
- Integration testing prior to deployment
    - axe API (with integration into Selenium, etc., by Deque)
    - Tenon API (by Level Access)
- Automated scanning tools used for EU Monitoring

- Access Monitor (open source monitoring tool used by the Portuguese government)
- BOSA Accessibility Checker (open source monitoring tool used by the Belgian government)
- Observatorio de Accesibilidad Web (open source monitoring tool used by the Spanish government)

Resources

- Accessibility Conformance Evaluation Methodology (WCAG-EM) 1.0, Working Group Note, W3C
- WCAG Conformance
- Understanding Conformance

Domain Two E: Accessibility quality assurance

(i.e., assuring the quality of accessibility throughout the development life cycle, difference and overlap between user experience and accessibility).

Study Tasks

- Characterise and differentiate between the disciplines of Agile and Waterfall project management methodologies and compare the approaches each methodology would have for accessibility quality assurance.
- Demonstrate an understanding of the benefits of designing digital content with accessibility in mind as opposed to remediation.
- Characterise and differentiate between the disciplines of accessibility and user experience design and compare assumptions of each discipline.
- Understand how accessibility needs to be integrated into the entire product life cycle, including concept, requirements, design, prototyping, development, quality assurance (QA), user testing, support, and regression testing.
- Identify ways in which each person's role in the product life cycle can include some aspect of accessibility.

Overview

Plan - Create -Test. (P.C.T). These are the three tasks to cycle through during the process of developing a new site/design, new feature, or remediation of a site. Accessibility experts and

people with disabilities should be part of all these stages to ensure the quality and usability of the product.

Resources

- Planning and Managing Web Accessibility, W3C
- Integrating accessibility across the agile and waterfall development lifecycle by Irfan Ali, 2018

1. Planning, Managing, and Implementing Accessibility

1.1 Responsibilities and Organizational Approaches

Everyone involved in a project, from top management to project managers, designers, UX specialists, developers, and quality assurance, has some role in delivering an accessible product. Management should set the conditions for a sustainable accessibility implementation. It has to provide the necessary resources and budgets, possibly organise awareness events, ensure know-how through training, include accessibility in job descriptions, and define goals and policies. Having an accessibility expert on the team (or an external consultant) can be helpful, but the requirements need to be shared across the team. Every member of the team has a role to play.

1.2 Web Development Process

An agile approach to accessibility requires a fundamental understanding of all team members and the acceptance of accessibility as an integral part of the workflow. They must understand what accessibility is, who benefits, why it is important and how to do it.

Accessibility should be integrated into every step of the product lifecycle, starting with planning, followed by the actual creation of the product, and last but not least, testing. This cycle is repeated until the task is considered complete. Even after a project has been completed, care should be taken to ensure that accessibility is maintained.

Details will vary from one web development team to another, but the product life cycle can include concept, requirements, design, prototyping, development, quality assurance, user testing, support, and regression testing. Each person’s role in the product life cycle should include some aspect of accessibility.

Three tasks to cycle through during the process of developing a new site/design include:

- **Plan:**
  - Consider accessibility from the beginning.
  - Include people with disabilities in research.
  - Document your goals. Define conformance to accessibility standards in the "Definition of done".
  - Assign responsibilities to individuals and incorporate them into their job descriptions.
  - Determine budget and resources.
  - Ensure team members and content creators have access to accessibility expertise and training.
  - Review environment (e.g., existing guidelines, coding libraries, accessibility checks being included in quality assurance)
- **Create:**
  - Implement accessibility in the information architecture (IA) and user experience (UX).
  - Address all user needs in design decisions, including people with disabilities.
  - Developers must ensure that the code meets accessibility requirements.
  - Content creators are responsible for producing accessible content, such as structured content, alternatives for images and multimedia, and descriptive headings, links, and labels.
- **Test:**
  - Quality assurance and accessibility auditors need to test markup and content.
  - Use automated and manual testing. Be aware of the limitations of automated testing.
  - The accessibility testing process should follow the intended standard throughout the iterative development.

- Use a standard report structure for evaluation findings.
- Ensure that accessibility bugs are fixed and reviewed again.
- Track and communicate progress

In contrast to the agile approach, the waterfall approach is a linear process breaking down activities into sequential phases. Each phase depends on the deliverables of the previous phase. Instead of testing early and often, testing will be performed at the end of the development process. Since projects tend to be time-pressured and over budget, at the end, tests might then be not thorough, omitted, or ignored. Furthermore, if problems are discovered late in a process, it involves more work and is more expensive to fix problems than doing things right from the start.

Domain Two F: Testing with assistive technologies

(e.g., navigation of screen reader users, headings and landmarks, screen magnifiers, high contrast, using keyboard vs. mouse)

Study Tasks

- Know which combinations of assistive technologies work best with which browsers.
- Know how to use screen readers to navigate content via landmarks and headings, use forms, and read content, including data tables. Be aware that mechanisms often differ between desktop (mostly keyboard-based: tab and arrow keys) and mobile (often gesture-based).
- Be aware that users of AT often use their tools in very different ways. For example, while screen readers offer various navigation modes for expert users, many actual users do not know or use these modes.
- Understand the different screen reader modes for interacting with different types of page content (Windows screen reader: read/browse mode, forms mode, application mode, VoiceOver rotor, TalkBack menu).
- Be familiar with standard keystrokes for interaction with interactive elements in native HTML
- Be familiar with conventions for keyboard interaction within ARIA widgets.
- Know the limitations of your assistive technology knowledge. If you are not experienced, do not assume something is an error. It may be that you do not know

how to use the assistive technology correctly. Output anomalies or gaps could indicate a partial lack of screen reader support or bugs.

Overview

Testing with assistive technologies allows the tester to check for issues from the perspective of individuals with disabilities who use assistive technology to access webpages, web applications, and software applications. Types of assistive technology include screen readers, screen magnifiers, and speech recognition/voice control. It is important to note that results obtained from a tester simulating the use of assistive technology or using it infrequently can be very different from results from someone who uses assistive technology all the time. There are different views on ethical aspects of "simulating" a disability in the community. To understand whether an interface or functionality really works for specific user groups, you would have to carry out user testing (see Domain Two G). But that is something different from an expert using assistive technology to test for compliance.

When using assistive technologies for accessibility testing, the screen reader has a dominant position because it can verify that content generates appropriate programmatic output. For example, the screen reader will reveal if elements have an accessible name or the correct role, and that state changes are reflected also programmatically.

Other assistive technologies are less well-positioned to detect issues because they often compensate for deficiencies and thereby effectively mask issues that testing without them might identify. For example, settings in magnification software can improve the focus visibility or text contrast but may also make it harder to identify a lack of focus visibility or insufficient contrast.

1.Screen Reader Testing

For screen reader testing, be familiar with how people with disabilities use the screen readers. Often, screen reader tests are conducted in parallel or after WCAG conformance tests to verify issues. Screen reader testing in the context of conformance tests does not strive to emulate the typical user experience. This would be the aim of task-based usability tests with native screen reader users. Instead, it focuses on particular types of content to check if they are programmatically available. I.e., they reveal their name, role, and value when focused and allow interaction equivalent to their use without a screen reader. Screen reader testing will also show if a page is navigable by its headings, lists, or landmarks and whether table header cells in data tables are spoken when traversing a table. It can also verify that user interface controls can also be reached and activated when the screen reader is turned on.

Goals of screen reader testing can be:

- to identify issues that are not readily apparent in automatic checks or code inspection
- to analyze ARIA constructs and components
- to verify findings of code analyses
- to assess the level of accessibility support across several environments (browser / screen reader combinations)

While native web content is well-supported, advanced user interface components (created with HTML ARIA and JavaScript) often run into accessibility challenges. They are frequently built incorrectly and not in line with recommended implementation patterns. Gaps may exist regarding screen reader/browser support.

State changes are one accessibility subject that is not always obvious to non-native screen reader users. These are changes that occur in areas of a webpage or mobile app that do not currently have focus. An example might be a toast message, a shopping cart subtotal updating when an item is added, or a delivery fee being updated when a shipping address is entered. The most relevant criteria that involve announcing modifications out of focus for screen reader users are 4.1.3 Status Messages, 4.1.2 Name, Role, Value, and 3.2.2 On Input.

CSS can also affect accessibility for assistive technology users. For example, in some cases, content that should be hidden visually and programmatically, like dropdown menus, is still rendered by the screen reader, which can be very confusing. In other cases, the intention may be to hide content only visually and make non-visual hints or explanations available to screen reader users. In this case, using `display:none` or `visibility:hidden` would hide this content also for screen reader users, and thus not serve the authors' intention. Traversing and interacting content with a screen reader often reveals these common issues.

Assistive technologies must be tested with compatible browsers. The recommended screen reader browser combinations are JAWS with Chrome, NVDA with Firefox or Chrome, and VoiceOver with Safari. Even if assistive technology works with a combination of browsers, some issues may appear in one browser only. For example, some (ARIA) techniques might be supported by one screen reader but not another, depending on the browser used. Do not test on combinations that are not recommended or that do not fully support accessibility. Only use accessibility-supported techniques that work across commonly used screen readers and browsers. Using different browsers is therefore recommended: (Edge, Chrome, Firefox, Safari) and their compatible screen readers (See “Screen Readers and Browser Considerations”, Domain One H).

As pointed out in Domain Two D, automatic testing tools cannot find all the accessibility issues. They analyze code but may not consider the page's context.

Limits also exist when the page changes dynamically through user interaction. Therefore, automatic testing tools alone cannot determine conformance. Additional manual testing is needed, and that involves human judgment.

Automated testing tools can detect formal issues but cannot identify other non-formal, content-related, or context-related aspects of conformity. For example, when checking the alternative text of images, an automated test may reveal that an alt attribute is missing on an image element. Still, when it is present, humans need to check whether the alternative text makes sense: Does it describe the link target? Does it describe an information graphic meaningfully?

Many of the WCAG success criteria require checking formal and content-related aspects. You can automatically assess the formal properties of buttons, text fields, links, and HTML page titles: Is the element named? Are ARIA attributes valid here? Do they use allowed values? Are elements nested correctly? However, it is not possible to determine automatically whether such elements have meaningful text content or whether attribute values are set correctly.

The following list includes examples of what to look for when testing content manually. However, what to test manually depends on the type of content and its likely faults. The test list may be longer than the areas outlined.

Testing with assistive technology and manual testing can often complement each other.

2.Keyboard

The following are key requirements to consider during a manual evaluation process:

- All functionality is operable via keyboard (links, data input, or buttons)
- There is no keyboard trap
- There is a way to bypass repeated content (e. g Skip Links or landmarks)
- The focus order is logical, intuitive, and predictable
- The focus movement within components/widgets is as expected
- The focus indicator is visible
- The visual label matches the programmatic label (or is fully included in it)

3.Content

The following are essential requirements to consider during a manual evaluation process:

Text:

- Headings, lists, and other structural elements are marked up correctly
- Titles, headings, and form labels are descriptive
- Links are descriptive (in context)
- Changes to the language of parts are marked up accordingly
- No loss of functionality occurs when zooming up to 200%
- Text can be resized to 200%
- No loss of content or functionality when users change text spacing parameters

Images, Animation, Multimedia:

- Text alternatives for images are accurate and meaningful
- Alternatives for video (captions, audio description) and audio or video-only (transcript) are provided and accurate
- The user can stop animation, or it stops automatically after 5 seconds
- There are no content flashes

Colour:

- There is sufficient contrast (text on top of a gradient or images cannot be checked automatically)
- Colour is not used as the sole means to convey information

Consistency:

- Navigation mechanisms occur in the same order across a set of pages
- Components with the same functionality are identified consistently across the website

4. Usability testing

Study Tasks

- Demonstrate an understanding of user testing and compare it to accessibility testing.
- Understand the value of user testing by users with various types of disabilities.
- Consider the consequences of certain types of accessibility flaws. Some flaws are more critical than others.
- Demonstrate an understanding of the design's usability versus the accessibility and the conformance to accessibility specifications.

Overview

Some web designs may be complex or unintuitive for people with disabilities to use, even if the design technically passes accessibility guidelines. Consider accessibility requirements to be just the beginning of the process. They are minimum requirements to which further objectives should be added to maximise accessibility and usability.

Resources

- Accessibility, Usability, and Inclusion, WAI W3C
- Supplemental Guidance, WAI W3C
- Involving Users in Web Accessibility, WAI W3C
- Involving Users in Evaluating Web Accessibility, WAI W3C
- Browsing with assistive technology, TetraLogical, 2021
- A11ycasts with Rob Dodson, Google Chrome Developers, 2018 - 2019
- Testing with Screen Readers – Questions and Answers, WebAIM, 2019

Domain Two G: Testing for end-user impact (e.g., low vision, cognitive, mobile/touch)

Study tasks:

- Understand the value of user testing by users with a variety of types of disabilities.
- Think through the consequences of certain types of accessibility flaws. Some flaws are more damaging than others are.
- Consider the usability of the design, not just the accessibility or conformance to the specifications. For further information, go to: W3C's Involving Users in Evaluating Web Accessibility.

Overview

The WCAG Success Criteria are intended to support different user groups. They are written as testable statements to allow for the verification of conformance against the standard. This is particularly important regarding compliance with legislation based on standards like WCAG and EN 301 549.

Accessibility requirements focus primarily on technical accessibility but also include some usability aspects. While their primary aim is improving digital accessibility for people with disabilities, they also benefit other users, such as people who experience poor lighting conditions, use mobile devices or have temporary limitations or impairments, for example, due to an illness. The Success Criteria in WCAG and other accessibility standards usually address issues that disproportionately affect users with disabilities, while usability issues affect all users.

While standards help measure the level of accessibility and identify and then fix problems in a structured way, the extant techniques for accessibility documented in WCAG informational material and elsewhere do not cover all potential solutions that would meet a normative requirement. The constant development of technologies, new techniques, and best practices means that good accessibility techniques can emerge which are not yet formally documented and approved as WCAG Technique. Reasons include:

- Standards development and the authoring of techniques moves slowly, and may not have caught up with technological advances
- The technology used may not be mature enough for a technique to be considered accessibility-supported and therefore appropriate as a WCAG Technique
- It may be not easy to objectively test the conformance of content based on a new technique

Even if a webpage can be accessed with assistive technologies and meets accessibility standards, it may not be usable for end-users with disabilities. For instance, content like site navigation may be fully keyboard-operable but contain dozens of tab stops. Technically, the author meets WCAG requirements, but keyboard users' usability and user experience are poor. Another example would be a web page with images whose alternative texts are too verbose. The page may still be accessible, but the experience is disrupted because users cannot explore alternative texts in the same way as standard text. Users must decide whether to listen to the entire text or cancel the output.

Some best practices that benefit people with cognitive disabilities are not included in the official guidelines. Accessibility for persons with cognitive disabilities means making content as clear and concise as possible. Content optimised for cognitive aspects will have a simple and consistent page layout, common design elements, good content structure, and simple language. These techniques do not only facilitate accessibility for individuals with cognitive disabilities; they also promote usability for all users. WCAG provides supplemental guidance on addressing many cognitive accessibility issues. These recommendations are not required to meet WCAG, but authors are encouraged to follow this guidance to meet more user needs.

While following accessibility standards provides a solid foundation for usability, it alone does not guarantee that a digital product can be used effectively, efficiently, and satisfactorily by people with disabilities. Usability testing with persons with disabilities is therefore recommended. Beyond usability tests, people with disabilities and older users should ideally be involved already in the requirements and early development stages of a design: for example, in task analyses, design walkthroughs, or tests of prototype design patterns or navigation structures.

Remember that people with disabilities are just as diverse as any other person. There is variation even within one category, such as people with limited vision. That means input from one person with a disability cannot simply be applied to all people with disabilities. The length of time the person has experienced a disability and their level of assistive technology expertise can also factor into the user experience.

Involving users with disabilities and older users is important for a more effective and efficient way of implementing accessibility. However, this alone cannot cover the diversity of disabilities, adaptation strategies, and assistive technologies. It is best to combine user involvement and standards.

Resources

- W3C’s Involving Users in Evaluating Web Accessibility

Domain Two H: Testing tools for the web

(both automated and manual tools, i.e., what they are and what are their limitations, e.g., unit testing, browser-based tools, spider tools, bookmarklet, automated tools used to monitor site vs. external tools)

Study Tasks

- Understand the purpose of ACT Rules.
- Know the strengths and limitations of automated testing tools.
- Differentiate between the kinds of accessibility issues that can be found with automated tools and issues that require manual testing.
- Understand how accessibility software tools can be used at various stages in the web development process (e.g., design/develop/test).
- Be familiar with the types of software tools available (site-wide scanning, server-based analysis, unit testing, integration testing, browser developer tools, browser add-ons, simulators, guided manual testing, etc.).
- Know how to use the keyboard and screen readers to navigate ARIA custom widgets.
- Be familiar with standard keystrokes for interaction with interactive elements in native HTML.
- Know how to use the browser's built in Development Tools to inspect source code

Overview and Resources

- Guided manual testing based on heuristics
    - Accessibility Insights for Web (by Microsoft)
    - ARC Platform (by TPGi)
    - Axe Auditor (by Deque)
    - JAWS Inspect (by Freedom Scientific)
- Browser developer tools and add-ons
    - The inspector feature in developer tools of browsers

    - Web Developer Toolbar (by Chris Pederick)
- Accessibility API viewers
    - Accessibility Viewer (for Windows, by Paciello Group)
    - XCode Accessibility Inspector (MacOS, by Apple)
- Simulators
    - colour Oracle (for colourblindness simulation, by Bernard Jenny)
    - No Coffee vision simulator (low vision and other conditions, by Aaron Leventhal)
- Single-purpose tools
    - Headings Map (by Jorge Rumoroso)
    - PEAT – Photosensitive Epilepsy Analysis Tool (by Trace R&D Center)
    - Contrast Checker (by WebAIM)
    - colour Contrast Analyser (by the Paciello Group)

Domain Three: Remediating issues in web solutions - (20%).

Domain Three A. Level of severity and prioritization of issues

(e.g., cost benefit; legal risk, user impact, what is the problem, what to focus on first)

Study Tasks

- Characterise and differentiate between the ideal/best solution and the “good enough” solution, respecting the particular project, its environment, intended target groups, and resources.
- Understand the important role of end users in assessing and validating remediation approaches.
- Demonstrate that you understand the difference between fixing the particular issue and completely redesigning the web page.
- Demonstrate the ability to distinguish the feasibility of a particular solution in different contexts.
- Demonstrate the knowledge of practical and simple hints, leading to better web accessibility.
- Be able to communicate the purpose, approach, and strategy to remediate.
- Understand the importance of making the right stakeholders aware, educated, and included in implementation recommendations.
- Know the possibility of using maturity models and tools to illustrate progress and sustainability.

Overview

It is best to integrate accessibility into the whole web production process, from identifying user requirements to design and implementation and, finally, user acceptance testing. As content changes and expands, repeated testing can maintain a good level of accessibility. Testing should occur early and regularly to find and fix problems as early as possible.

However, it may be necessary to prioritise findings and narrow the scope of remediation. One reason for this is the wish to use limited resources most effectively. In other cases, especially when the website or app is already operational, remediation will often prioritise critical issues where the risk is highest that users find themselves excluded.

1.Prioritizing Accessibility Issues

Accessibility audits often result in a list of problems that need to be fixed. Even if the goal is to fix everything, you probably cannot fix every issue in a single update. Prioritizing the more critical issues may be necessary, especially when the website or app has already been released.

- **Impact:** Determine if issues have a high, medium, or low impact on whether people with disabilities can use the website. High-impact issues may prevent users from completing an essential task or process. A CAPTCHA without an alternative, a control that cannot be activated with the keyboard, or important content that is completely hidden from programmatic output may all completely block users from using the website. In contrast, medium impact issues will make use more difficult but will not entirely prevent use. Annex B of the EN 301 549 is a great tool to determine the impact on different user groups.

- **Low effort repairs:** Prioritise issues that are easy to repair. Many issues can be addressed quickly and bring immediate benefits and improvements. Differentiate between style, markup, and functionality changes.

- **Repeated issues:** Problems will occur repeatedly if they are part of the web page template, component library, or design system or because the code is reused in multiple locations (for example, when implementing forms or dialogs). Your remediation effort will benefit greatly if recurring issues are prioritised.

- **Key content and tasks:** First, remediate issues that occur on the pages with the highest traffic. It may be that 20% of your pages get 90% of traffic. You may also identify key tasks, such as registration, search, or checkout processes, or focus on new content or features to avoid introducing new barriers.

- **Cost-benefit:** Prioritization can benefit from assessing the cost-benefit provided by the remediation. This can help identify the most critical issues in determining whether users can complete a task.

- **Impact on interface:** Some issues might have far-reaching effects on the interface as it exists and require substantial changes. Where appropriate, things that require extensive redesign could be subordinated to things that have little to no impact on the interface.

- **Legal risk:** Determine whether the issue identified is a legal risk.

2.Recommend Strategies And/or Techniques for Fixing Accessibility Issues

Recommending remediation strategies requires knowledge of how to create accessible content (competency 1), the ability to identify accessibility issues (competency 2), and the wisdom to choose an appropriate remediation technique for the circumstances, considering all the practical limitations of real-world situations. When communicating issues and recommendations, it is important to inform the developer precisely about where the problems are, what the problems are and how to fix them.

The task of making a website accessible can be simple or complex. Accessibility depends on many factors, such as the type of content, the size and complexity of the site, and the development tools and environment. Recommending the best strategies and techniques that reflect particular conditions (time, money, the current state of the development, chosen CMS, etc.) can help both users and developers in finding the most efficient way to make the website accessible.

Domain Three B: Recommending strategies and/or techniques for fixing issues

(i.e., best solution, solution that is most widely useful, feasibility of solution, fixing vs. redesign, how to fix it).

Study Tasks

- Characterize and differentiate between the ideal/best solution and the “good enough” solution, respecting the particular project, its environment, intended target groups, and resources.
- Understand the important role of end users in assessing and validating remediation approaches
- Demonstrate that you understand the difference between the fixing of the particular issue and the complete redesign of the web page.
- Demonstrate the ability to distinguish the feasibility of a particular solution in different contexts.
- Demonstrate the knowledge of practical and simple hints, leading to better web accessibility.
- Be able to communicate the purpose, approach, and strategy to remediate.
- Understand that it is important that the right stakeholders are made aware, educated, and included in implementation recommendations.

- Know about the possibility of using maturity models and tools to illustrate progress and sustainability.

Overview

Various techniques can be used to meet a Success Criterion when evaluating web content. The normative text of WCAG defines the requirement but does not prescribe a particular coding solution. WCAG provides techniques to meet Success Criteria (the “Sufficient Techniques”), which are only informative. Often, there are different ways to satisfy a particular Success Criterion.

1.Distinguishing between Failures and Optional Best Practices

It is important for an accessibility tester or auditor to understand—and communicate—the difference between conformance and best practices. A conformance test should flag a conformance failure only when deficiencies clearly map on documented WCAG Failures or in other ways clearly fail to satisfy the normative success criterion. The underlying deficiency should be clearly described.

A website that minimally conforms to a standard requirement can still be difficult to use. Beyond normative failures, a good auditor should be able to recognise deficiencies that are not strictly failures and recommend a better practice to achieve a more usable end result.

While they do not provide examples for all scenarios, supporting WCAG documents such as the Understanding texts, the Techniques, and the ARIA Authoring Practices Guidelines can be referred to and consulted to determine whether an issue counts as a normative failure or should be considered not best practice.

2.Fixing vs. Redesign

Deciding whether to remediate or rewrite accessibility defects involves carefully evaluating multiple factors, including the severity and extent of the defects, code complexity, budget, and potential user impact. Here are important considerations for making this decision:

1. **Severity and Scope of Defects**: If the accessibility issues are minor and localised, remediation is often the most efficient approach. However, a complete rewrite might be necessary if the defects are widespread and affect the website's core functionality. This is especially true if the defects are in templates or design systems.

2. **Code Complexity and Maintainability**: Older, complex, poorly structured code can make remediation difficult and costly. Rewriting the code to adhere to modern accessibility standards and best practices can provide a cleaner, more maintainable solution.
3. **User Impact**: The decision should prioritise the user experience. If the defects significantly hinder user accessibility, a more comprehensive approach might be warranted, potentially involving a rewrite to ensure a fully accessible and user-friendly interface.
4. **Resource Availability**: Consider available resources, including time, budget, and expertise. Remediation can be quicker and less costly, but if resources allow, a rewrite can result in a more robust and future-proof solution.
5. **Future-Proofing**: A rewrite can incorporate current and emerging best practices, making the website more adaptable to future accessibility requirements and technological advancements.

In practice, a hybrid approach is often used. Immediate remediation might address critical issues while a longer-term plan is developed for a comprehensive rewrite to ensure sustainable accessibility.

Ultimately, the decision to remediate or rewrite should be driven by a thorough analysis of these factors, aiming to provide the best possible outcome for all users.

Resources

- Planning and Managing Accessibility, WAI W3C
- What do you fix first by OPTASY, Medium, 2020

Domain Three C: Integrate Accessibility into the Procurement Process

Study Tasks

- Understand how accessibility can be incorporated into the procurement process.
- Demonstrate an understanding of the different types of Accessibility Conformance Reports and how they should be interpreted.
- Identify how to mitigate accessibility issues raised during the procurement process.

Overview

Like security and privacy, accessibility is a key consideration during the procurement process. It is at this point that organisations have the best chance to influence how accessible their ICT products and services will be to end users.

1.Role of accessibility experts

Accessibility experts can provide purchasers with:

- Guidance on good governance practices
- Advice on ICT procurement standards, such as EN 301 549
- A list of suitable questions for inclusion in tender documents and Requests for Proposal (RFP)
- Evaluations of tender responses, Accessibility Conformance Reports (ACRs) and Voluntary Product Accessibility Templates (VPATS)
- Testing of products
- Suggested contract wording regarding accessibility
- Vendor management, including defect remediation and accessibility roadmaps

Accessibility experts can also assist vendors to:

- Audit existing systems
- Remedy defects
- Create Accessibility Conformance Reports
- Respond to tender questions regarding accessibility

2.Accessibility Conformance Reports

An Accessibility Conformance Report (ACR), is a reporting format that assists buyers and vendors to identify accessibility features in ICT products and services. ACRs are based on the Voluntary Product Accessibility Template (VPAT) produced by the Information Technology Industry (ITI).

When a vendor offers a VPAT document, what they are technically providing is the ACR; the two terms are often used interchangeably. There are several VPAT formats:

- VPAT 2.4 Rev 508 - the U.S. Section 508 accessibility standard
- VPAT 2.4 Rev EU - the EN 301 549 version
- VPAT 2.4 Rev WCAG - the WCAG 2.1 version
- VPAT 2.4 INT - Incorporating Section 508, EN 301 549, WCAG 2.1

Accessibility Conformance Reports are made up of a set of criteria (taken from either Section 508, EN 301 549, or WCAG), their conformance level, and relevant remarks. The conformance levels are:

- Supports: The product's functionality has at least one method that meets the criterion without known defects or with equivalent facilitation.
- Partially Supports: Some functionality of the product does not meet the criterion.
- Does Not Support: The majority of product functionality does not meet the criterion.
- Not Applicable: The criterion is not relevant to the product.

It is preferable that vendors use an independent accessibility professional to create their ACR, rather than producing it in-house.

Ideally, every criterion will be “supported” in the Reporting Table, but in practice, very few products will fully support every single WCAG or other criteria. If a criterion is partially supported or not supported, this provides you with helpful information about what areas to pay attention to during product reviews.

3.Mitigating Accessibility Defects

When an ICT product or service has known defects, risks can be mitigated by:

- Obtaining a roadmap for addressing known accessibility issues

- Including binding accessibility commitments in contracts for products and services
- Putting in place an alternative access plan for users
- Monitoring vendor performance in relation to defect management
- Considering accessibility performance during contract negotiations and product renewals.

4.Using an Accessible Procurement Maturity Model

Maturity models are powerful tools for improving accessibility compliance, especially within procurement processes. They provide organizations with a structured framework to assess their current accessibility practices and measure progress over time. By identifying key stages of accessibility maturity, from initial awareness to full integration, these models help organizations understand where they stand and what steps are necessary to advance. In procurement, applying a maturity model ensures that accessibility is systematically considered in purchasing decisions, reducing the risk of acquiring inaccessible products or services, and fostering long-term, sustainable accessibility improvements across the supply chain.

There are two publicly available free maturity models that specifically address procurement:

1. Policy Driven Adoption for Accessibility (PDAA) is an initiative by the National Association of Chief Information Officers (NASCIO) that helps procurement organizations choose vendors who can meet accessibility requirements.
2. The W3C maturity model procurement dimension includes assessments for sourcing, negotiation, and selection of goods and services by reviewing use of accessibility processes, criteria, contract language, and decision-making to procure and maintain accessible products and services throughout the procurement life cycles.

Resources

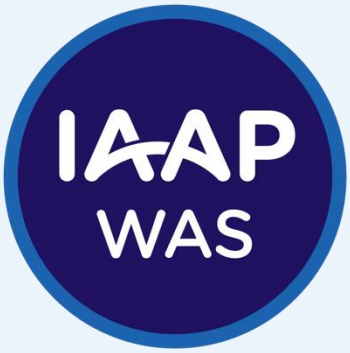
- ITI ACR/VPAT templates
- ITI ACR/VPAT authoring training
- W3C Accessibility Maturity Model, Procurement Dimension

Conclusion

The IAAP Web Accessibility Specialist (WAS) Body of Knowledge (BOK) is provided as general information regarding the current state and future of web accessibility best practices. This document aims to identify intermediate levels of knowledge for those working in web accessibility as one portion of the larger accessibility profession.

The IAAP Certification Team can be reached by email at certification@accessibilityassociation.org.

IAAP
International Association
of Accessibility Professionals
A division of G3ict



WEB ACCESSIBILITY SPECIALIST (WAS)

United in Accessibility

www.accessibilityassociation.org