IAAP

International Association
of Accessibility Professionals

IAAP CERTIFIED WAS

IAAP WEB ACCESSIBILITY SPECIALIST (WAS)

Body of Knowledge

October 2020

United in Accessibility

www.accessibilityassociation.org

2020 Co-Editors:

Rosemary Musachio, CPWA & Richard Streitz

Contributors:

- Dr. Paul Bohman, CPWA
- Pina D'Intino, CPACC
- Samantha Evans
- 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

Edited: September 2019 to include WCAG 2.1

First Draft Edit Version v1.10 October 2020

Table of Contents

A. The Purpose of this Document

This Body of Knowledge document outlines the skills expected of candidates seeking to obtain the Web Accessibility Specialist (WAS) designation. The IAAP Web Accessibility Specialist (WAS) exam is a technical knowledge exam.

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 guide does not guarantee successful completion of the exam.

As of the WAS BOK Update from June 2020, the WAS BOK covers concepts from WCAG 1, WCAG 2.0, and WCAG 2.1. Unless specific information is discussed about a certain version, “WCAG” will be used.

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

B. IAAP Exam Preparation Resources

Test-takers can study resources available anywhere in preparation for the exam. IAAP lists a collection of WAS Exam resources for preparation that are both free and for purchase. You can find these resources on the IAAP Prepare for the WAS Exam webpage.

Candidates should review each section of the WAS Content Outline to determine where they have the most background, where they have some knowledge, and 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.

C. 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 programming alone will not provide the background necessary to achieve the WAS credential successfully. Hands-on experience and knowledge of programmatic code elements, WCAG 2.1 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 accessibility activities/projects for persons with disabilities

How to contribute to the creation of an inclusive society

Additional Information

- IAAP homepage
- General information about IAAP certification
- WAS Exam Content Outline
- WAS Frequently Asked Questions
- WAS Preparation Resources

D. The WAS Exam Content at a Glance

Creating Accessible Web Solutions (40% of the exam)

1. Guidelines, principles, and techniques for meeting success criteria (including WCAG, WAI-ARIA, ATAG, basic concepts, limitations of the specific guidelines, principles, and techniques, what is normative vs. non-normative; what is included in the different levels (A, AA, AAA))
2. Basic knowledge of programming (at a conceptual level; principles and concepts related to programming; the impact of specific coding practices on web solutions vs. writing specific code)
3. Accessibility quality assurance (i.e., assuring the quality of accessibility throughout the development life cycle, difference and overlap between user experience and accessibility)
4. Accessibility supported technologies (including user's assistive technologies and accessibility features; a combination of assistive technologies and users agent; design decisions in choosing technologies that support accessibility; e.g., not choosing Flash when something else has better accessibility support, differences in assistive technology supports and behaviors, differences in support for touch when a screen reader is on vs. off)
5. Standard controls vs. custom controls (e.g., using standard controls when possible, if using custom controls build them using WAI-ARIA best practices)
6. Single-page applications (e.g., focus control, delays for AJAX-screen reader compatibility, live announcements)
7. 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)

Identify accessibility issues in web solutions (40% of the exam)

1. Interoperability and compatibility issues (e.g., works with JAWS, Chrome, Safari, etc…)
2. Identifying guidelines and principles regarding issues (including WCAG, 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))
3. Testing with assistive technologies (e.g., navigation of screen reader users, headings and landmarks, screen magnifiers, high contrast, using keyboard vs. mouse)
4. Testing for end-user impact (e.g., low vision, cognitive, mobile/touch)
5. 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)

Remediating issues in web solutions (20% of the exam)

1. Level of severity and prioritization of issues (e.g., cost-benefit, legal risk, user impact, what is the problem, what to focus on first)
2. Recommending strategies and/or techniques for fixing issues (i.e., the best solution, a solution that is most widely useful, feasibility of the solution, fixing vs. redesign, how to fix it)

1. Creating Accessible Web Content

1.1. Understand and interpret accessibility specifications and techniques.

1.1.1.Overview

This competency focuses on designing and creating web content in accordance with the following W3C accessibility specifications:

- Web Content Accessibility Guidelines (WCAG) 2.1
    - WCAG 2.1 (normative)
    - Understanding WCAG 2.1 (non-normative)
    - Techniques for WCAG 2.1 (non-normative)
    - How to Meet WCAG 2.1 (non-normative)
- Accessible Rich Internet Applications (WAI-ARIA) 1.1
    - WAI-ARIA 1.1 (normative)
    - WAI-ARIA 1.1 The Roles Model (non-normative)
    - WAI-ARIA Authoring Practices 1.1 (non-normative)
    - Accessible name and description computation 1.1 (non-normative)
- Authoring Tool Accessibility Guidelines (ATAG) 2.0
    - ATAG 2.0 (normative)
    - Implementing ATAG 2.0 (non-normative)

1.1.2.The W3C

The W3C is an internationally recognized web standards body that identifies its approved technical specification standards as “W3C Recommendations” (such as HTML, CSS, etc.). The consortium has several Accessibility specifications that have achieved W3C Recommendation status, including WCAG, ATAG, and WAI-ARIA. Other accessibility related W3C recommendations, such as the User Agent Accessibility Guidelines (UAAG), are out of the scope of the IAAP WAS certification.

1.1.3.Web Content Accessibility Guidelines (WCAG) 2.1

The international standard, WCAG 2.1, 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; but 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.

Web Content Accessibility Guidelines (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.

1.1.4.W3C overview of the Web Content Accessibility Guidelines

The WCAG document is organized into principles, guidelines, and success criteria. The four POUR principles are:

- **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 (functionality from keyboard).
- **Understandable**: Information and the operation of user interface must be understandable.
- **Robust**: Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

WCAG 2.1 has 13 guidelines within these principles, while WCAG 2.0 has 12. Input Modalities is the new guideline added to WCAG 2.1. This additive approach helps to make it clear sites that conform to 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 level: A, AA, or AAA. A site that meets all the success criteria at level A is said to “conform to” level A. A site that meets all the success criteria of both level A and level AA conforms to level AA. A site that meets all the success criteria at all levels conforms to level AAA.

Most laws and policies that reference either WCAG 2.0 or WCAG 2.1 focus on conformance to level AA (which includes success criteria at both the A and AA level).

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

- **Sufficient technique**s: If the web content meets sufficient techniques, it successfully meets the success criterion.
- **Failure techniques**: If the web content fails any of these, 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, improved ideas, the emergence of new accessibility-related specifications, and so on.

For further information, go to:
W3C's Understanding Techniques for WCAG Success Criteria.

1.1.5.WCAG Versioning

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

WCAG 2.1 added 17 success criteria added to WCAG 2.0. WCAG 2.1 did not change the 63 existing success criteria. The latest WCAG update includes how content for new technologies, such as mobile phones or tablets, should be developed or remediated so persons with disabilities can use them. WCAG 2.1 also includes more criteria that focus on individuals with low vision or with cognitive disabilities, which were not addressed in WCAG 2.0.

For further information, go to: W3C's What's New in WCAG 2.1.

1.1.6.Accessible Rich Internet Applications (WAI-ARIA) 1.1

The Web Accessibility Initiative (WAI) of the W3C created WAI-ARIA to increase the accessibility of content – dynamic content in particular – for assistive technology users, such as screen reader users. WAI-ARIA defines attributes that can be added to standard HTML to define the name, role, and values (properties and states) 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 full accessibility.

ARIA enables developers to mark up dynamic content, including custom controls

created with AJAX, HTML, and JavaScript, to improve their accessibility. The WAI-ARIA Authoring Practices define the best practices for widget structure, keyboard behaviors, and dynamic content. They also promote interoperability.
See some ARIA examples:

- aria-label
    - Invisible
    - Defines a new name for the element, which usually comes from its content
    - Doesn´t allow the text to be clicked
    - Contains a string
- aria-labelledby
    - Relates to the id of another element
    - Can relate multiple ids from different elements
    - The related texts are normally visible in the page
    - Replaces the element name presented to assistive technologies that usually comes from its content
- aria-describedby
    - Relates the id of another element
    - Provides additional information to the element

For further information, go to:W3C's Introduction to WAI-ARIA webpage.

1.1.7.Authoring Tool Accessibility Guidelines (ATAG) 2.0

The ATAG specification requires authoring tools (HTML/web editors, content management systems, social media sites, blog commenting features, discussion forums, user rating features, etc.) to:

1) Have an accessible user interface and

2) Support the production of accessible content.

For further information, go to:
W3C's Authoring Tool Accessibility Guidelines (ATAG) Overview.

1.1.8.Normative versus non-normative documents:

"**Normative**" documents define accessibility practices required for conformance (to a specification).

"**Non-normative**" documents provide guidance and techniques for interpreting and conforming with the normative requirements, but non-normative techniques are not required for conformance.

Non-normative documents provide information about the different way web technologies need to work with authoring tools, user agents, and assistive technologies.

Non-normative documents may change more frequently than normative documents, to adapt to changing technologies and current best practices.

Study topics related to specifications and techniques:

Topics that should be studied and mastered to enable you to pass the WCAG 2.1 components of the Web Accessibility Specialist exam.

- Understand and interpret WCAG 2.1
    - Understand the relationship between principles, guidelines, and success criteria.
    - Understand the intent, requirement, and impact of each principle, guideline, and success criterion.
    - Be familiar with sufficient, failure, and advisory techniques for each success criterion.
    - Understand the conformance level designations (A, AA, AAA).
    - Identify the conformance level of each WCAG 2.1 success criterion.
    - Understand the three types of techniques and the W3C vetting process for techniques.

For further information, go to:W3C's Introduction to understanding WCAG 2.1 webpage.

- Understand and interpret WAI-ARIA 1.1
    - Understand the purpose and impact of WAI-ARIA 1.1.
    - Understand the WAI-ARIA 1.1 model of names, roles, and values.
    - Know when and why to use WAI-ARIA 1.1, and when to use standard HTML instead.
    - Be familiar with the authoring practices for custom widgets, including semantic structure, keyboard behavior, etc.

For further information, go to:CanAdapt's Demystifying WAI-ARIA.

- Understand and interpret ATAG 2
    - Understand how ATAG 2 applies to web content authoring tools.

- Understand the meaning and intent of the two main sections of ATAG 2.
- Understand the intent, requirement, 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

For further information, go to: W3C's Guide to Understanding and Implementing Authoring Tool Accessibility Guidelines 2.0.

- Understand the difference between normative and non-normative documents, information, and be able to identify which documents are normative.

1.2. Create accessible JavaScript, AJAX, and interactive content

The WAS exam does not cover the details of JavaScript programming syntax (a person can pass the exam without being a professional JavaScript programmer). However, web designers and developers must be aware of how JavaScript, AJAX, and interactive content affect accessibility. They must be able to identify the concepts, principles, and strategies of accessible JavaScript interaction design.

1.2.1.Support for JavaScript in accessibility APIs and assistive technologies

Modern screen readers and other assistive technologies can process the results of JavaScript processes, as long as the JavaScript is coded with accessibility in mind. There are no inherent barriers in the technologies themselves to making JavaScript inaccessible.

1.2.2.Study topics related to JavaScript, AJAX, and Interactive Content

Some of the highlights of accessible interaction design are shown in the list below.

- **Manage focus** - When JavaScript changes the visual focus (e.g., when a dialog is activated), JavaScript should be used to manage the keyboard focus so that it follows the visual focus.
- **Use semantic HTML** - HTML defines sets of elements, attributes, and attribute values. These features have specific semantic meanings that user agents intend to process in particular ways.

- **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 in the DOM, causing them to miss most additions/changes in previous positions in the DOM

- **Create device-independent event handler**s - JavaScript event handlers must be device-independent. The functionality must be available with the keyboard, mouse, touch, voice, etc.

- **Create accessible JavaScript widgets** - design patterns and examples of common widgets.

- **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, either 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.

For further information, go to:
Mozilla’s Developer CSS and JavaScript accessibility best practices.

1.3. Integrate accessibility into the quality assurance process.

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 need to be part of all these stages to ensure the quality and usability of the product.

Accessibility needs to be integrated into the entire product life cycle. 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:

Plan and design phases include:

- research
- requirements
- design of information architecture (IA) and user experience (UX)

Create content & components phase includes:

- creating front-end markup & programming
- creating text content
- creating multimedia

Test content and components phase includes:

- testing markup & programming
- testing text content
- testing multimedia

1.3.1.Study topics related to accessibility quality assurance

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

For further information, go to:
integrating accessibility across the agile and waterfall development lifecycle by Irfan Ali.

1.4. Choose well-supported accessibility techniques

Most modern screen readers have comparable support for the most important accessibility techniques and features. Still, screen reader behaviors and support for newer features – including some types of custom ARIA widgets – can sometimes be incomplete, inconsistent, or faulty.

It is important to design to the accessibility standards, rather than cater to specific screen reader differences or bugs. In some cases, it can be appropriate to implement workarounds, compromises, or polyfills, but recognize that screen reader behaviors can change at any time. In contrast, guidelines and recommendations are more stable over time and usually represent the best approach.

1.4.1.Study topics related to accessibility support

- Understand the importance of coding to standards, rather than to the quirks or features of only one set of technologies.
- Understand the concept of progressive enhancement, in which the baseline interaction is possible using legacy technologies, and the more modern web site features are available for technologies that are more modern.
- Understand the importance of testing web designs for accessibility across a variety of platforms, browsers, and assistive technologies, and not just assuming they will work, even if they technically conform to published accessibility specifications.
- Know which combinations of assistive technologies work best with which browsers for testing purposes.
- Know how to determine when an inaccessible outcome is the result of poor design versus poor support of the technology.
- Know how to tell the difference between inaccessible content and incomplete or faulty accessibility testing techniques.
- Avoid design techniques or technologies that work only on certain platforms.
- 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 caution!).

For further information, go to: W3C's Techniques for WCAG 2.1.

1.5. Create interactive controls/widgets (standard or custom) based on accessibility best practices

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. In broad strokes, these interactive components do not have to become 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 can sometimes complicate achieving success.

1.5.1.Study topics related to interactive controls/widgets

Some highlights of key concepts to consider with interactive controls/widgets include the following:

- 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 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.

For further information, go to:

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

1.6. Create accessible single-page applications (SPAs).

Single-page designs (designs that bring in new content – often through AJAX 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 with the intent to improve performance and streamline the user experience of the web application. Screen readers typically do not announce anything when content is loaded via AJAX, so users may be completely unaware that new content has been loaded.

1.6.1.Study topics related to single-page applications

- 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.

For further information, go to:
a11yportal's Dynamic Updates, AJAX, and Single-Page Apps

1.7. Create web content that is compatible with the strategies used by persons with disabilities to access web content.

People with disabilities often use different methods to access web content, compared to the general population. They may use only the keyboard, voice input, or alternative input devices. Many people with disabilities use assistive technologies. There are many kinds of assistive technologies, designed for many different kinds of disabilities. Most people who are blind, for example, use screen reader software to listen to web content and to allow them to navigate through web content according to the semantic markup (e.g., landmarks, headings, tables, etc.). Creating accessible content requires knowing about different strategies; and how different concepts, designs, and implementations affect those strategies.

The following is a summary of some of the highlights of the strategies and technologies used by people with various kinds of disabilities (this list is not all-inclusive).

1.7.1.Vision

Depending on if a user has some vision or no vision, the strategies are quite different. Users with limited vision often use the vision they do have to help them navigate and find their way around an interface, while blind users need other strategies to navigate. It is also a difference between using interactive interfaces via a touch screen or not.

1.7.2.Blind users

Blind users need a screen reader to translate what is shown visually on the screen to speech or (less commonly) to braille. Since the user can´t see the interface, the screen reader needs to provide the user with mechanisms to understand the structure of the interface and to navigate.

A common approach in screen readers to help users is to create lists of headings, links, form controls, etc. In this way, users can quickly move between such elements on the page. For this to work, it is important to use the correct markup. The support for standard HTML elements is often quite good, and most screen readers also support significant parts of WAI-ARIA, but you need to know how the support is for the markup you are planning to use in common screen readers and browsers.

Among screen reader users, knowledge and strategies vary considerably. 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. In case of text messages, for example: in the questionnaire, a person who is blind that browses with a screen reader uses the tab key to surf from field to field. If the instruction messages are in a <p>, <span>, or <div>, these will not receive the default focus, and the user will probably not find them. The best way to resolve this is aria-describedby for screen readers to read measures when users navigate to form fields.

1.7.3.Blind users with a touch screen

The interaction model for sighted touchscreen users is quite different from the interaction model for blind touchscreen users. Sighted users swipe through 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 method for interacting 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 a screen reader to the next.

You should be familiar with the most common screen readers, including how they work and what parts of HTML5 and WAI-ARIA they support. Some examples of common screen readers include:

- JAWS for Windows by Freedom Scientific
- NVDA by NV Access
- Narrator for Windows by Microsoft
- VoiceOver for MacOS by Apple
- ChromeVox (on Chromebooks)
- TalkBack (Android)

1.7.4.Screen reader and browser combinations

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

1.7.5.Recommended combinations to guarantee better compatibility:

- Windows:
  - JAWS and Google Chrome (and in some cases IE11)
  - NVDA and Firefox
  - Narrator and Edge
- macOS
  - VoiceOver and Safari
- iOS
  - VoiceOver and Safari
- Android
  - TalkBack and Chrome

1.7.6.Screen reader considerations

- The difference between visual order of content and the structural order of content presented to the screen reader
- How the role attribute in WAI-ARIA and semantic elements in HTML work to create a page structure
- How to use screen reader lists of different objects (e.g., tables, lists, images, etc.) to find important information on the screen
- How to navigate with a keyboard using a screen reader
- How to enter information in a form

While browsing with keyboard and screen reader, the tester must verify several points:

- Elements must receive focus (links, data input or buttons)
- The location of focus is visualized on the webpage for sighted keyboard users
- The screen reader announces adequate tags or alternative texts
- The functions of the components, such as menu and submenu, accordion, tabpanel, carousel, etc., can be browsed or performed correctly. In such components, the roles and state notification should be verified.

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

For further information, see the following resources:

JAWS 2020 Documentation

NVDA User Guide

Mac OS VoiceOver User Guide

ChromeVox User Guide

Android Talkback User Guide

iOS VoiceOver User Guide

1.7.7.Low vision users

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 different strategies for users with low vision:

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

You need to have a basic understanding of common magnifying tools like:

- ZoomText by Freedom Scientific
- SuperNova by Dolphin

You also need to understand how to support text enlargement in smartphones and how the zoom functions work:

- Vision Accessibility features in iPhone
- Magnification in Android
- Font size and display size in Android

Some concepts that are important to understand include:

- It is quite common to use a mouse to navigate but also to use keyboard commands to speed up the interaction
- Users need clear borders to be able to see where different areas start and end
- Users might alter colors, contrasts, and font
- Users often combine zooming and screen reading

- A big gap exists between users that only need the built-in zoom in the browser and users that zoom the interface 32 times and use screen readers as a complementary technique. Their strategies vary accordingly.
- 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

1.7.8.Reading

Users that have reading difficulties may use assistive technology such as Kurzweil or text-to-speech provided by the platform, or may not have any assistive technology. Some may instead search for information that is not text. Instead of searching on Google, a user might use YouTube. If there is an alternative to text, an alternate format may be preferable.

When reading text, it is possible to 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. Some of the changes that can be done are:

- Text size
- Text color and background color
- Font

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

There are some assistive technologies available to this group of users. They are based on either screen reading or altering the presentation.

Screen readers for users with reading difficulties read the text that the user selects. The screen reader does not offer any navigation help since most individuals in this user group do not need that level of support. Examples of readers for people with reading difficulties include:

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

For assistive technology that alters the presentation, it is of key importance that text is offered as real text and not images of text. Otherwise, it becomes difficult or impossible for the assistive technology to do any alteration of the text.

Some concepts that are important to understand include:

- Differences between screen readers for this group and screen readers for blind users
- The importance of using real text instead of images of text
- Typeface, color, spacing, and more: Readable fonts include typeface suggestions, as well as font size, and letter and word spacing widths.

1.7.9.Cognition

This user group is also very diverse. The group includes users that have concentration problems, users with neuropsychological disabilities, and users with intellectual disabilities, among others. Some users with intellectual disabilities can´t use the interface even with assistive technology. However, most users can use the interface even if it might require some alterations or assistive technology.

When using assistive technology, it is often mainly the same assistive technology as users with reading difficulties. Additionally, they might also use ad blockers, screen masking, and other such tools to help keep focusing 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.

Some concepts that are important to understand include:

- The importance of a clean, simple layout and presentation to aid this group: Provide control of as many aspects of the website as possible. CSS can be used to provide control of how information is presented.
- Plain simple language is key for this group.
- Content alternatives: Use sound also to present information. People differ in their ability to process information in different forms (text, image, audio). So, providing more than one form reaches more people.
- Images and multimedia should be used to supplement text wherever possible. The use of appropriate and clear graphics can help to enhance understanding of materials, but do not overuse graphics and avoid animated graphics, as they can be distracting and increase cognitive load.
- Users with cognitive disabilities often use assistive technology to read and write

For further information, go to:

- Cognitive Accessibility at W3C
- W3C’s Making content usable for people with cognitive and learning disabilities

1.7.10. Motor

Users that have motoric disabilities are a broad group. Some of them are completely or partly paralyzed, others have trouble with fine motoric actions, perhaps due to tremor or rheumatism.

Another example is users that need to control the interface with a keyboard. These users often use the tab key to jump between interactive objects in the interface, like buttons, form fields, and links.

Some users use a point and click-based solution; this might be:

- Ordinary mouse
- Tremor filtering mousepad
- SteadyMouse
- Eye-tracking
    - EyeGaze
- Tobii
- Point scanning with a switch control
    - Switch Control on iPhone, iPad and iPod Touch
    - Switch Access for Android
- Mac OS Headpointer Feature

Some users use the keyboard, much like blind users, 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 can tell them this.
- Since they don’t use a screen reader, they can’t use lists of headings, links, and other items to move down a page quickly.
- When a site is navigable through the keyboard (generally with Tab, Enter, and arrow keys) in an orderly and intuitive order, important groups of persons are benefited: persons who cannot use a mouse, persons with motor disabilities. Furthermore, structuring the site to be operable through the keyboard facilitates navigation with assistive technologies (like switches), which simulate movement with Tab or Enter keys.

Some users use speech control. You should have a basic knowledge of how this works. Some examples include:

- Voice Control for iOS
- Voice Control for Mac OS
- Voice Access for Android
- Speech Recognition for Windows

Some concepts that are important to understand include:

- Users have different input devices so our web solution must be possible to use with different input methods like:
  - Mouse
  - Keyboard
  - Voice
  - Touch/gesture
- To have large, clearly visible, and easy to spot clickable areas are of key importance for many users
- It is important to help users see what object is currently in focus
- Prediction and autofill are very helpful as it limits the number of keystrokes required when typing.
- Note: When creating web pages that appear and operate in predictable ways, the design and appearance of web pages whose appearance, operation, and navigation are predictable and consistent, avoid confusion to the user. For this reason, unexpected content should not be created. This can be confusing to persons with visual, cognitive, or motor disabilities.

For further information, go to:

- Motor Disabilities by Deque University
- Video by Level Access: Digital Accessibility User Impact: Motor Disabilities

1.7.11.Hearing

Users that are deaf from birth may have sign language as their first language. For them, this means that text information on websites they visit often is their second or third language.

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.

Accurate captioning is very important for video content. The use of only automated tools and those relying only on artificial intelligence creates many opportunities for an incorrect translation of spoken content.

Some concepts that are important to understand include:

- Provide alternatives, such as captions or transcript, for time-based media
- For prerecorded audio-only, an alternative for time-based media is provided that presents equivalent information for prerecorded audio-only content, such as a transcript
- Offering transcription benefits persons with hearing disabilities. This transcription can be available on the same screen where the audio is or through a link.

For further information, go to: Video Captions by W3C.

2. Identify accessibility issues/ problems

2.1. Identify interoperability and compatibility issues.

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.

2.1.1. Study topics related to strategies used by people with disabilities

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

2.1.2. Keyboard accessibility

- Actionable elements (links, buttons, controls, etc.) are focusable with the keyboard
- All focusable elements must have a visible focus indicator
- Logical tab order
- No keyboard traps

While browsing with keyboard and screen reader, must verify several points:

- Elements must receive focus (links, data input or buttons)
- The location of focus is visualized
- The screen reader announces adequate tags or alternative texts
- The functions of the components, such as menu and submenu, accordion, tabpanel, carousel, etc., can be browsed or performed correctly.

2.1.3. Touch device accessibility

- Sufficiently large touch target size
- Designing and building for any screen orientation

- Custom and complex (e.g. multi-finger) gestures must have an alternative method for activation (e.g., by activating a button)

- Motion-activated events (e.g., shaking the device) must have an alternative method for activation (e.g., clicking a button). The user must be able to disable the motion feature to prevent accidental activation due to tremors or spasms.

For further information, go to: CSS, HTML, ARIA, browsers, assistive technology and interoperability by the Paciello Group

2.2. Determine conformance to accessibility specifications based on accessibility issues found.

It is important first to understand the standard being applied (e.g., WCAG 2.0 or WCAG 2.1 level AA), and to distinguish between true accessibility failures with respect to that standard, versus poor design decisions that don't fail the standard (even if they fail a different standard or a different level within the standard).

2.2.1.Identify the success criterion for every accessibility failure.

Often the main goal of an accessibility evaluation is to determine if the design conforms to a particular specification or not. The target in many settings is WCAG 2.0 or WCAG 2.1 level AA. If this is the task, it is important to be able to map each accessibility issue to one of the success criteria within that specification. An image missing alternative text, for example, would map to success criterion 1.1.1.

2.2.2.Differentiate between WCAG 2.0, WCAG 2.1

As mentioned earlier, WCAG 2.1 has 13 guidelines and 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. While executives need to know WCAG 2.0 as policy standards, developers aim to implement WCAG 2.1 success criteria techniques.

2.2.3.Distinguish between failures (of success criteria) from optional best practices.

If the goal is conformance, don't include every possible accessibility best practice in an accessibility audit. Doing so will make it harder to meet the conformance goal quickly. Any issues not directly related to the target conformance level should be documented separately or flagged somehow as not being required for conformance (even if they are good ideas in terms of their accessibility impact).

2.2.4.Study topics related to determining the level of conformance to accessibility specifications

- Become familiar with the specifications (WCAG, WAI-ARIA, and ATAG), and know which success criteria apply to which conformance level.
- Be able to distinguish between failures of accessibility criteria versus other bad accessibility practices that are not referenced in the specifications.

For further information, go to: W3C's What's New in WCAG 2.1

2.3. Test with assistive technologies.

Testing with assistive technologies (AT) allows the tester to check for issues from the perspective of individuals with disabilities who use AT to access webpages, web applications, and software applications. Types of AT include screen readers, screen magnifiers, voice recognition, and even keyboards. It is important to note that a tester simulating the use of AT is vastly different from someone who uses AT all the time.

Testing with a screen reader or voice recognition, for example, allows you to check if a page is navigable by its headings, lists, or landmarks. You can see if you can access form fields or navigate data tables. You can test if user interface controls can be reached and activated by using AT shortcuts. When you test with a screen magnifier such as ZoomText, you can see if content can be viewed in different contrast settings. Screen magnifiers also should be tested to see if shortcuts are functional when navigating webpages.

Although you may not consider a keyboard as assistive technology, it is essential for screen reader users and persons who cannot manipulate a mouse. When testing a webpage using a keyboard, navigate it with the Tab key. Accessing such controls as radio buttons and trees require using arrow keys.

Assistive technologies must be tested with compatible browsers. As of June 2020, JAWS works best with IE 11 or Chrome, NVDA works best with Firefox, and VoiceOver works best with Safari. Even if AT works with a combination of browsers, issues may appear in a browser only. For instance, JAWS doesn't read headings in certain versions of Internet Explorer even when the webpage does have a heading structure.

Using different browsers is recommended: (Internet Explorer, Chrome, Firefox, Safari, Opera) and their compatible screen readers (See "Screen Readers and Browser Considerations").

2.3.1.Study topics related to testing with assistive technologies

- Know how to use screen readers to navigate elements such as landmarks, headings, tables, forms, etc.
- Know how to go forward and backward through focusable content in screen readers (e.g., using the tab key or shift+tab in desktop browsers)
- Know how to go forward and backward through all content in screen readers (e.g., using the down or up arrow keys in most screen readers).
- Know the consequences of using contrast enhancement modes such as Windows High Contrast Mode
- Know how to use the keyboard only to navigate forms.
- Know how to invoke keyboard combinations to navigate various types of elements with a screen reader.
- Know how to use the keyboard and screen readers to navigate ARIA custom widgets.
- Know your limitations in your assistive technology knowledge. If you are not experienced, do not assume something is an error if it may be that you do not know how to use the assistive technology correctly.
- Know why it is important to get the opinions and feedback of users with disabilities using their respective assistive technologies.
- Know which combinations of assistive technologies work best with which browsers. Do not test on combinations that are not recommended or that do not fully support accessibility.

2.4. Test for end-user impact

Some web designs may be difficult or unintuitive for people with disabilities to use, even if the design technically passes accessibility guidelines.

The guidelines are not all-inclusive, meaning that there are some good accessibility techniques not represented by the guidelines. Reasons that some good techniques may not be in the list of guidelines include:

- It is difficult to objectively verify compliance with the technique
- The writers of the guidelines did not recognize the need for the technique when writing the guidelines.
- The technique was not necessary (or at least not anticipated) at the time that the guidelines were written because the technologies or circumstances that require the technique are newer than the guidelines.

Even if a webpage can be accessed with AT and meets accessibility standards, it still may not be usable for end-users with disabilities. For instance, although the chat link is accessible, it is the last link on a webpage. Another instance could be if the webpage is overloaded with images that have alt attributes, the page still may be accessible but not very easy to follow.

Accessibility for persons with cognitive impairments means making a website or program as clear and concise as possible. This type of accessibility implies consistent page layout throughout a website, noticeable links in the same position, and short paragraphs. These techniques do not only facilitate accessibility for individuals with cognitive impairments; they also promote usability for all users. If the contents of a website cannot be followed easily, then nobody can use it. However, some of these techniques are not official guidelines, such as the length of a short paragraph.

With technologies advancing daily, guidelines become outdated quickly. For example, when WCAG 1.0 was written, no success criteria or techniques were developed for ARIA or Silverlight, let alone mobile devices. Since then, WCAG 2.1 and Mobile Web Best Practices 1.0 have been created.

2.4.1.Study topics related to testing the end-user impact

- 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

2.5. Use accessibility testing tools effectively.

No accessibility software tool can find all the accessibility issues on a web site. However, software tools can speed up finding accessibility issues and increase the overall accuracy when supplemented by a skilled manual evaluation of the same content. For example, an automatic tool can filter all the alternative texts of the page images (to verify if all have alternative texts). Still, only a human can determine if the texts are adequate for the images.

Some highlights of key concepts to consider during a manual evaluation process:

- Test environment: evaluate the single page (home page).
- Verify correct navigation and functionality using the mouse and the keyboard:
    - Interactive elements must receive focus (links, data input or buttons)
    - The location of focus is visualized
    - The screen reader announces adequate tags or alternative texts
    - The functions of the components, such as menu and submenu, accordion, tabpanel, carousel, etc., can be browsed or performed correctly. In such components, the roles and states notification should be verified.
- A tester should identify two important points in the results:
    - Accessibility problems
    - Optimal accessible results

There are many tools and brands available, some of which are listed below.

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 Evaluation Tools List (the IAAP is not responsible for the content or accuracy of that list).

2.5.1.Automated Testing

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

- **Server-based page analysis**
  - wave.webaim.org (by WebAIM)
  - Cynthia Says (by Cryptzone)
  - SiteImprove Accessibility (by SiteImprove)
- **Browser-based developer/QA tools (one page at a time)**
  - aXe browser add-on (by Deque)
  - Google accessibility developer tools
  - WAVE browser add-on (by WebAIM)
  - JavaScript bookmarklets written to test specific accessibility criteria or expose certain types of markup (by various companies and individuals)
  - AInspector for Firefox (by University of Illinois at Urbana-Champaign)
  - ANDI
  - ARC Toolkit by TPG
  - MS Accessibility Insights
  - Google Chrome Lighthouse
- **Unit testing during development**
  - aXe API (with integration into Selenium, etc., by Deque)
  - Tenon API
  - WorldSpace Attest (by Deque)
- **Integration testing prior to deployment**
  - aXe API (with integration into Selenium, etc., by Deque)
  - Tenon API

2.5.2.Manual Testing Tools

- **Guided manual testing based on heuristics**
  - WorldSpace Assure (by Deque)
  - JAWS Access

- **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**
  - Color Oracle (for colorblindness simulation, by Bernard Jenny)
  - No Coffee vision simulator (low vision and other conditions, by Aaron Leventhal)
- **Multi-purpose accessibility tools**
  - Web Accessibility Toolbar (by Paciello Group)
- **Single-purpose tools**
  - Headings Map (by Jorge Rumoroso)
  - PEAT – Photosensitive Epilepsy Analysis Tool (by Trace R&D Center)
  - Contrast Checker (by WebAim)
  - Color Contrast Analyser (by the Paciello Group)

2.5.3.Study topics related to accessibility testing tools:

- 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 generate and translate results into usable analysis, prioritization, and categorization, along with how to differentiate technical reports from business reports.

3. Remediate (fix) accessibility issues

3.1. Prioritize accessibility issues based on the level of severity.

When managing accessibility projects, address the most important accessibility issues first. Start with the core functions of the web site. If there were a shopping cart, that would be considered a core function. Every step of the shopping experience needs to be accessible: e.g., searching for products, reading about the products, adding the products to the cart, reviewing the cart, entering account information, entering payment information, and receiving confirmation of the purchase. Non-core functions of the site can wait until the accessibility of the core functions has been addressed.

3.1.1.Study topics related to prioritizing accessibility issues based on the level of severity

- Identify the Issue: Identify the accessibility issue in either the style, markup, or functionality of the content.
- Identify the User Impact: Associate the issue with the impact on the affected accessibility user. Differentiate between minor issues that can be worked around via advanced accessibility methods and complete blocker issues.
- Identify the Legal Risk and Cost-Benefit: Determine whether the issue identified is a legal risk or a potential usability improvement. Consider factors such as visibility and frequency. Determine the cost-benefit provided by the remediation, due to accessibility users' ability (or current inability) to complete the flow in question.
- Determine the Level of Effort associated with Issue Remediation: Differentiate between style, markup, and functionality changes.
- Prioritize: Use the user impact, legal risk, cost-benefit, and level of effort to determine severity, identify low-hanging fruit, and prioritize all issues.

For further information, go to:
MSU's Prioritizing Web Content for Accessibility Review and Remediation

3.2. Recommend strategies and/or techniques for fixing accessibility issues.

Recommending remediation strategies requires knowledge of how to create accessible content (competency I), the ability to identify accessibility issues (competency II), and the wisdom to choose an appropriate remediation technique for the circumstances, taking into account all the practical limitations of real-world situations.

The task of making a website accessible can be simple or complex. Making it accessible 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.

3.2.1.Study topics related to recommending strategies and/or techniques for fixing accessibility issues

- Characterize and differentiate between the ideal/best solution and the “good enough” solution, respecting the particular project, its environment, intended target groups, and resources.
- Demonstrate the understanding 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.
- Communicate the purpose, approach, and strategy to remediate.
- Ensure that the right stakeholders are aware, educated, and included in implementation recommendations.
- Consider the use of maturity models and tools to illustrate progress and sustainability.

For further information, go to: W3C’s Planning and Managing Web Accessibility

- 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 hopes 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 WEB ACCESSIBILITY SPECIALIST (WAS)

International Association
of Accessibility Professionals

United in Accessibility

www.accessibilityassociation.org