

ALMA MATER STUDIORUM ·  
UNIVERSITÀ DI BOLOGNA

---

**DOTTORATO DI RICERCA IN  
COMPUTER SCIENCE AND ENGINEERING**  
Ciclo 35 Settore Concorsuale: 01/B1 - INFORMATICA  
Settore Scientifico Disciplinare: INF/01 - INFORMATICA

**ON MAKING WEB ACCESSIBILITY  
MORE ACCESSIBLE:  
STRATEGY AND TOOLS FOR  
SOCIAL GOOD**

**Supervisore:**  
Fabio Vitali  
**Co-supervisore:**  
Paolo Ciancarini

**Presentata da:**  
Vincenzo Rubano  
**Coordinatore Dottorato:**  
Ilaria Bartolini

**Esame Finale  
Anno 2023**

An accessible version of this PhD thesis is available at  
<https://www.titengodocchio.it/tesi-dottorato>

*To those who have always been there. In the good times, the bad times, and even the worst times. To those who believed in me, and to those who didn't: this thesis is dedicated to you.*



# Abstract

It is estimated that over one billion people in the world experience some form of disability: whether they are permanent, situational or temporary, disabilities always *impose* restrictions to the independent living of a person. Nowadays, though, digital technologies can dramatically improve the life of people with disabilities, giving them opportunities that could not even be imagined a few years back. Digital technologies move very fast, achieving more and more impressive results as time passes.

There is a paradox in this situation, though. While digital technologies have a huge potential to improve life quality for people with disabilities, the majority of web content available on the Internet still exhibits critical accessibility issues. Given the abundance of technical standards, laws and regulations, as well as educational resources and supporting tools to help anyone produce accessible content, how can this be possible?

Additionally, the full potential of digital technologies when it comes to making the *physical* world more accessible has not been fully unleashed yet. Therefore, the research project documented in this PhD thesis is organized in two different, yet strictly related branches.

In the first one, an innovative approach to improve the current situation with web accessibility is proposed; it is argued that, in order to improve the current situation, it is necessary to *rethink from the ground up* the way in which web accessibility is dealt with, producing innovative resources to *make web accessibility more accessible*. This ambitious goal can be achieved with a threefold approach that encompasses:

- 
1. Creating **AX**, a declarative framework of web components, capable of enforcing as much as possible the generation of accessible markup. This is achieved by employing both static analysis (i.e., enforcing the developer to provide information that is necessary to generate such markup), and integrating existing tools for analyzing the generated markup to catch issues that could not be detected otherwise (i.e., color contrast, where external **CSS** stylesheets can play a significant role). The framework is implemented leveraging existing web technologies (such as various **W3C** standards about web components) and can be leveraged along with other existing **JavaScript** frameworks. The Visual Rendering of the components provided by the framework does not carry any design (i.e., the framework does not take any responsibility regarding their visual appearance, unless this is strictly necessary to fulfill a requirement to create a component).
  2. Creating an innovative accessibility testing and evaluation methodology, that leverages an innovative approach to communicate accessibility testing results. The main idea behind this approach is to “map” accessibility issues to concepts (such as mouse interactions and the visual rendering of a page) that developers are already extremely familiar with, so that they can (a) immediately realize the presence of accessibility issues and (b) clearly and directly perceive their impact on people with disabilities. This methodology has been implemented by developing the Sighted Architects Helper for Accessibility Notation (**SAHARIAN**) browser extension available for **Google Chrome**.
  3. The creation of a categorized, structured collection of curated accessibility resources to make it easier for non-accessibility experts to find them, even if they do not know what to look for, how and where. This has been done by developing the **A11A** website.

Both the theoretical foundations behind these tools, and their design, development and evaluation process will be discussed; they will be compared

to the current solutions available on the market to highlight their advantages naturally arising from their innovative characteristics.

The second part of the research project documented in this PhD thesis provides a significant contribution towards unleashing the full potential of digital technologies to improve the accessibility of the *physical world*. More specifically, this is achieved with two different, yet related proposals. First, the Scientific Collections Accessibility Making Process (SCAMP) methodology to make scientific, manipulable artifacts accessible both for blind and visually impaired people and the general public has been developed, applying it to the collection of the Brendel models available at university of Bologna to demonstrate its feasibility; this methodology has been carefully developed to *emphasize* natural characteristics of the objects it is applied to, yet enhancing such strengths to make these artifacts more accessible by means of interactive, multimodal and multisensorial experiences. Finally, the prototype of Accessibility for Virtual Tours (A11YVT), a system to support *accessible virtual tours* is presented. What makes A11YVT outstanding from other similar attempts already proposed in the literature is the fact that it provides all the features that blind and visually impaired need in order to explore indoor environments they are not already familiar with, while satisfying all the requirements and expectations of their sighted peers from such a tool in a truly universal design fashion.

All the proposed artifacts have been evaluated by performing various usability tests with their intended target audiences; evidence shows that they are effective towards reaching the goals for which they have been developed. Insights suggest that they have the potential to *change* the current status quo for the better in a very significant way.





# Contents

<b>Abstract</b>	<b>i</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 An overview of accessibility resources</b>	<b>13</b>
2.1 Web accessibility standards and guidelines . . . . .	13
2.1.1 WCAG 2.1 . . . . .	15
2.1.2 ATAG 2.0 . . . . .	19
2.1.3 WAI-ARIA 1.1 . . . . .	21
2.1.4 Assessing conformance to accessibility standards . . . . .	24
2.1.5 Making other content types accessible . . . . .	26
2.1.6 Supporting accessible content . . . . .	28
2.2 Laws and regulations . . . . .	32
2.3 Accessibility resources . . . . .	39
2.3.1 Resources for accessible content production . . . . .	39
2.3.2 Resources for accessibility testing and evaluation . . . . .	41
<b>3 The current status of digital accessibility</b>	<b>47</b>
3.1 The current situation of web accessibility . . . . .	47
3.1.1 E-Government websites accessibility . . . . .	49
3.1.2 Education websites accessibility . . . . .	52
3.1.3 Service providers websites and mobile apps accessibility . . . . .	54
3.2 Reasons behind the inaccessible Web . . . . .	56
3.3 Physical objects and indoor environments' accessibility . . . . .	63

---

3.3.1	Making indoor environments more accessible . . . . .	63
3.3.2	Making physical objects more accessible . . . . .	66
<b>4</b>	<b>The research project</b>	<b>69</b>
4.1	The research project and its purposes . . . . .	69
4.2	Is web accessibility accessible? . . . . .	73
4.2.1	The usability of accessibility tools . . . . .	75
4.2.2	Accessibility is not <i>enforced</i> . . . . .	77
4.2.3	Finding accessibility resources . . . . .	82
4.3	Making web accessibility more accessible . . . . .	85
4.3.1	Introducing Sighted Architects Helper for Accessibility Notation ( <i>SAHARIAN</i> ) . . . . .	86
4.3.2	Introducing the <i>AX</i> framework . . . . .	91
4.3.3	Introducing <i>A11A</i> . . . . .	95
4.4	Introducing <i>MARE</i> and <i>ARIA-Service</i> . . . . .	97
4.4.1	Introducing <i>MARE</i> . . . . .	98
4.4.2	Introducing <i>ARIA-Service</i> . . . . .	100
4.5	Introducing <i>A11YVT</i> and the <i>SCAMP</i> methodology . . . . .	103
4.5.1	Introducing <i>A11YVT</i> . . . . .	103
4.5.2	Introducing the <i>SCAMP</i> methodology . . . . .	108
<b>5</b>	<b>Implementing the research project</b>	<b>115</b>
5.1	Implementing the <i>AX</i> framework . . . . .	116
5.2	Implementing <i>SAHARIAN</i> . . . . .	123
5.3	Implementing <i>A11A</i> . . . . .	129
5.4	Implementing <i>MARE</i> and <i>ARIA-Service</i> . . . . .	138
5.4.1	Implementing <i>MARE</i> . . . . .	139
5.4.2	Implementing <i>ARIA-Service</i> . . . . .	142
5.5	Implementing <i>A11YExamples</i> . . . . .	149
5.6	Implementing <i>A11YVT</i> and the <i>SCAMP</i> methodology . . . . .	156
5.6.1	Implementing <i>A11YVT</i> . . . . .	156
5.6.2	Implementing the <i>SCAMP</i> methodology . . . . .	161

---

<b>6</b>	<b>Evaluating the research project</b>	<b>167</b>
6.1	The evaluation process . . . . .	167
6.2	Evaluating the AX framework, SAHARIAN and A11A . . . . .	175
6.2.1	Evaluating the AX framework . . . . .	176
6.2.2	Evaluating SAHARIAN . . . . .	179
6.2.3	Evaluating the A11A website . . . . .	182
6.3	Evaluating the SCAMP methodology and A11YVT . . . . .	187
6.3.1	Evaluating the SCAMP methodology . . . . .	188
6.3.2	Evaluating A11YVT . . . . .	192
<b>7</b>	<b>Conclusions and future developments</b>	<b>197</b>
	<b>Appendices</b>	<b>205</b>
	<b>Appendix A WebAIM one million 2022 - Main Web Content Accessibility Guidelines (WCAG) violations</b>	<b>207</b>
	<b>Appendix B Features of the SAHARIAN browser extension</b>	<b>209</b>
	<b>Appendix C Elements in the AX framework</b>	<b>213</b>
	<b>Appendix D A class diagram for the AX framework</b>	<b>217</b>
	<b>Appendix E A stylesheet to make different checkbox implemen- tations look the same</b>	<b>221</b>
	<b>Appendix F My publications</b>	<b>225</b>
	<b>Bibliography</b>	<b>227</b>
	<b>Glossary</b>	<b>285</b>
	<b>Ringraziamenti</b>	<b>291</b>



# List of Figures

2.1	Accessibility specifications and their targets . . . . .	14
3.1	Most common WCAG violation types in the top 1,000,000 home pages analyzed by WebAIM . . . . .	48
4.1	Number of elements in the top 1,000,000 home pages analyzed by WebAIM . . . . .	74
4.2	Percentage of pages in the top 1,000,000 home pages analyzed by WebAIM affected by accessibility issues over time . . . . .	75
4.3	Visual rendering for a checkbox group for non-disabled users as can be expected by non-disabled developers . . . . .	88
4.4	The Brendel model of <i>Phaseolus vulgaris</i> Linn, illustrating the early stages of the germination process of a bean plant. Source: Sistema Museale di Ateneo (SMA). Alma Mater Studiorum - Università di Bologna . . . . .	112
5.1	A class diagram showing some components in the AX framework and their relationships . . . . .	118
5.2	The first screen of the interface for the SAHARIAN browser extension . . . . .	123
5.3	Homepage of the New York Police Department's official website as shown in a web browser . . . . .	125
5.4	Visual representation of the Homepage of the New York Police Department's website generated by SAHARIAN . . . . .	126

---

5.5	The visual representation generated by SAHARIAN for a checkbox implemented correctly . . . . .	127
5.6	The visual representation generated by SAHARIAN for a checkbox which does not support keyboard navigation properly as it cannot be toggled . . . . .	128
5.7	A screenshot of the A11A Homepage . . . . .	132
5.8	A screenshot showing the A11A contact form . . . . .	137
5.9	A screenshot showing the same accessible checkbox implemented with various HyperText Markup Language (HTML) elements (on the left) and their code (on the right) . . . . .	153
5.10	A screenshot showing the same checkbox with various accessibility issues implemented with different HTML elements (on the left) and their code (on the right) . . . . .	154
5.11	Accessibility for Virtual Tours (A11YVT) frame of a virtual tour with two points of interest . . . . .	159
5.12	Artifact obtained by applying the model recreation phase to the Brendel model of <i>Phaseolus vulgaris</i> Linn. . . . .	163

# Listings

4.1	Example of a simple, inaccessible web page that renders fine . . .	78
4.2	Five options to implement an accessible text field in HTML . . .	79
4.3	Creating a text input field using the AX framework . . . . .	93
5.1	Ain't Markup Language (YAML) code from which the A11A contact form is generated . . . . .	136
D.1	the PlantUML code for the class diagram showing some components of the AX framework . . . . .	217
E.1	CSS stylesheet applied to make them look the same even if implemented with different HTML elements . . . . .	222





# List of Tables

A.1	WCAG violation types detected in the most popular 1 million Homepages by WebAIM Source: [Web22a] . . . . .	208
-----	---	-----



# Chapter 1

## Introduction

The history of the Web is pretty recent. It can be very impressive to reflect back on what the World Wide Web (WWW) has become in thirty years and its accomplishments. As Paul reflects in her book *100 things we've lost to the Internet* [Pau21], the Internet led to a complete revolution in many aspects of society having an impact both on “big” phenomenons (e.g. the creation of whole new business opportunities, innovative services, or brand-new approaches to the way in which people learn and study), as well as the little details of our everyday life: think for instance to the endless debates within a group of friends to understand who is right in a discussion, perhaps giving phone calls to many friends or involving unknown people just to have another opinion; while in the past this was pretty standard, anyone nowadays can search for the truth just in a few seconds. The debate on what the WWW gave us and what is taking back from our lives is going stronger than ever, involving all aspects of our society and interesting almost any web technology, and the mentioned book by Paul is only one of such interesting analysis on how the Internet has changed our life.

But the history of web accessibility is even more recent than that! The first web accessibility-related act was enacted only in 1998 in the United States (U.S.) (Section 508, an amendment added to the Rehabilitation Act of 1973 [U.S15]), when the World Wide Web was already going strong. It

was pretty evident, though, that it was not as inclusive as even its creators imagined it to be. While the World Wide Web Consortium (W3C) was already providing different standards to regulate various aspects and web technologies which laid the foundation for the WWW (think of HTML and Cascading StyleSheets (CSS), for instance), web accessibility was not considered in the process; the first official recognition of its importance, as well as the necessity of putting additional efforts towards the goal of making the WWW accessible, came when Tim Berners-Lee (the inventor of the WWW itself) announced the creation of a task-force at W3C specifically in charge of creating the necessary standards to ensure an accessible web. In a very popular quote of 1998, he stated that:

The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.

This statement clearly and undoubtedly recognizes the importance of web accessibility and how it is essential to the existence of the WWW itself. The first official recommendation from the W3C about web accessibility has been *WCAG 1.0* [CVJ99] (nowadays superseded by *WCAG 2.1*), published only in 1999.

But at the end of the day, apart from technical standards and regulations, one may wonder: what is accessibility? How does it impact web pages? While providing exhaustive answers to these apparently simple questions would perhaps require a PhD thesis on its own (and these topics have been indeed widely discussed in the literature), some fundamental aspects can be briefly covered in order to provide some context about this research project.

In a broader sense, accessibility is the design of products, devices, services, vehicles, or environments ensuring that they are usable by people with disabilities. Two main elements characterize the definition of accessibility: accessible products should guarantee both “direct access” to such products and “indirect access” to them (i.e. meaning that they are compatible with assistive technologies). More specifically, when it comes to the WWW, the concept of *web accessibility* can be introduced: essentially, web accessibility

is the inclusive practice of ensuring there are no barriers that prevent interaction with, or access to, websites on the World Wide Web by people with physical disabilities, situational disabilities, and socio-economic restrictions on Internet connections' bandwidth and speed. Web accessibility aims to accommodate the different needs of people with various disability types:

- people with visual impairments (including blindness, low vision and poor eyesight, various types of color blindness, and so on);
- people with motor impairments (such as difficulties or inability to use hands including tremors, muscle slowness, loss of fine muscle control, and so on);
- people with hearing impairments, including deafness and people who are hard of hearing;
- people with Seizures (such as photo epileptic seizures caused by visual strobe or flashing effects);
- cognitive and intellectual disabilities, including developmental disabilities, learning difficulties (such as dyslexia or dyscalculia), and cognitive disabilities of various origins affecting memory, attention, developmental “maturity”, problem-solving and logic skills, and any other cognitive or intellectual “ability”;

But web accessibility does not only benefit people with these disability types; indeed, it also benefits people who may be experiencing permanent, temporary and situational disabilities. People with permanent disabilities can be seen as all those people who experience one or more of the disability types illustrated before; except for very rare circumstances, these disabilities are *permanent* as they last for all life of the people experiencing them. In contrast, temporary disabilities are all disabilities which are *temporary* by their nature: a person may be experiencing them for a period of time, but except for very rare circumstance the date in which the person will stop

experiencing such limitations is known and falls in a short-term period; these disabilities include (but are not limited to) broken limbs, hand injuries, or short term disabilities following surgery or medical treatments. Finally, just like permanent disabilities, situational disabilities prevent people from doing what they need to do and being who they want to be; people in this situation do not need to be “disabled” permanently or in the traditional sense, yet due to the context (the “situation”) in which they perform one or more tasks they experience the same limitations that a person with a comparable disability would. Examples include all people who may be experiencing a boundary based on the current “situation” in which they are performing the task (e.g. partial sight due to sun lighting, being one-handed due to carrying a baby in a stroller, pushing a shopping cart or pulling a trolley).

Additionally, different studies and researches available in the literature have proven that an accessible website or application is a website or application that is more *usable* for all people [SSS16, SSS18, SSS17, AHV16]. Indeed, at the moment there is not a clear, very well established border in between accessibility and usability, as often both intersect and contribute to each other: unsurprisingly, many research projects have tried to investigate the problem, leading to different ideas, approaches and even philosophies applied to the product design and evaluation processes (just mentioning a few of them, these include the proposal by Sauer et al. [SSS20], *universal design* [Mac97], and *inclusive design* [CCKL13]).

In light of these considerations, it is clear that there are a lot of people who may benefit from having an accessible web in many ways. Therefore, it is reasonable to wonder about the efforts to achieve such goal and the current situation with it. Over the years, many standards and regulations have been adopted at various levels to provide the opportunity of creating (and consuming) accessible web content. Such standards had to be produced for each layer that actually make the web possible in order to ensure that they *cooperate* towards enabling the accessible web. To mention a few of such layers and standards, these include:

- Application Programming Interface (API)s within operating systems, designed and implemented to let Assistive Technologies (AT) get the information they need to convey when they need it, as well as enabling other user agents (such as web browsers) to expose the *semantics* about the rendered elements in a structured way suitable for AT consumption;
- technical standards (such as *User Agents Accessibility Guidelines (UAAG) 2.0* [ALPS15], *core accessibility API mappings* [DSS+17, DSC22] and *graphics accessibility API mappings* [BRDC+18a]), aimed at *regulating* how user agents should communicate information to operating system APIs and expose the semantics of HTML elements in web pages;
- specifications on how AT are expected to consume data through accessibility APIs provided by operating systems and convey information to users, such as the *accessible name and description computation 1.1* [DGC18] specification;
- various specifications (including *WCAG 2.1* [CCKC18], *Active Rich Internet Applications (WAI-ARIA) 1.1* [DMC+17], *Authoring Tools Accessibility Guidelines (ATAG) 2.0* [CCKC15], to mention only a few) indicating how web content should be structured, so that user agents can expose all the necessary information to accessibility APIs provided by the operating system, and AT can consume it properly, making web accessibility actually possible.

Among all these specifications, *WCAG 2.1* [CCKC18] lie the foundation for accessible web content as it provides the *characteristics* that web content should satisfy in order to be considered accessible. As clearly explained in the document *understanding WCAG 2.1* [Acc22c], given the wide variety of web technologies available nowadays and the complexity arising from the fact that there can be multiple implementation techniques to achieve the same goal using them, WCAG has been structured in three abstraction layers:

- fundamental principles (perceivable, operable, understandable, robust), which inspire the whole standard;

- guidelines, a set of which is provided to address each fundamental principle; they help content authors, developers, designers and any stakeholder in general understand the characteristics that web content should have in order to be accessible;
- success criteria, which are *practically testable* statements that allow to check whether web content satisfy each guideline or not.

Depending on which and how many success criteria are satisfied, web content conformance to WCAG 2.1 can be classified in three different levels (A, AA, and AAA). The problem, though, is that this standard is not (and cannot ever be) complete on its own as a standalone resource: given the rapid evolution of the web and the very high number of technologies that make it actually possible, the WCAG 2.1 specification has been written to be *abstract enough* to be applied to various present and future web technologies, yet *practical enough* to be both implementable in practice leveraging present and future technologies and testable to assess web content conformance to it. These complexities lead to the proliferation of a very high number of additional support resources (guides and tutorials, code examples, videos, courses, ETC.) explaining how to produce accessible content, how to achieve the desired conformance level to the WCAG standard, how to actually assess web content conformance to it, and in some cases even providing clarifications on how the standard itself is intended to be used. Such (very useful) resources are provided by many sources, including the W3C itself. To further complicate the situation, additional standards are provided to illustrate how to achieve conformance with WCAG 2.1 with specific digital technologies whose application domain contains specific accessibility-related challenges: for instance, these include the *WAI-ARIA 1.1* [DMC<sup>+</sup>17] specification for *rich* web applications, the *PDF/UA* [ISO14] standard for Portable Document Format (PDF) files, the *the Digital Accessible Information System (DAISY) 3.0* and the *EPUB accessibility guidelines* [GKL<sup>+</sup>22a] standards for books, or the *Mathematical Markup Language (MathML) 3.0* language for Math content.



It may be tempting to think that, given the abundance of accessibility-related resources available nowadays, even if the process of producing accessible web content is complex overall the current situation with web accessibility is in pretty good shape. Unfortunately this is not the case: evidence shows that the majority of web content available on the Internet still exhibits critical accessibility issues. Many websites and applications cannot be used at all (entirely or in part) by people with disabilities, as they do not respect such standards and therefore people with disabilities cannot interact with them leveraging the assistive technologies they need (e.g. screen readers for blind people, or screen magnifiers for the visually impaired). The reasons leading to this situation have been investigated multiple times over the years, yet explaining precisely why there is so much inaccessible content out there remains not possible. Other than a general and very well known lack of interest in the topic among stakeholders, factors like the effectiveness of standards, guidelines, laws and regulations, how stakeholders (especially web developers) perceive them, and the appropriateness of accessibility testing and evaluation tools may play a very significant role [RV21].

But the situation is even worse. Recognizing the importance of web accessibility and that inaccessible web content constitutes a discrimination towards people with disabilities, many countries and international organizations have enacted different acts ([Ita04, U.S15, Gov10, Gov19, Gov18, Eur19, Eur16] and many others for different countries and regions) to promote, and in some cases even mandate, the creation of accessible websites and applications if they *match* some specific criteria (for instance, being developed by public bodies or on behalf of them, or offering public services of general interest). As discussed in depth in Section 3.1 even such laws and regulations may not be effective; in the best scenarios, they produce minimal effects only in the long term. Instead, there is evidence that web accessibility advances as technology makes it easier to produce accessible web content (and not due to an increased interest in it by stakeholders) [HR13]. This is the reason why it has been decided to tackle the problem of *making web accessibility more*

*accessible*.

More specifically, the research project documented in this PhD thesis can be broken down in two different, parallel branches. The first branch arises from the simple consideration that, given the current situation with web accessibility, it may be necessary to *rethink from the ground up* the whole approach to it. In fact, currently web technologies do not enforce at all the creation of accessible content; accessibility issues can easily be unnoticed, as it is possible to create different equivalent implementations of the same widget which can look the same, yet have different accessibility degrees. User agents do not encourage stakeholders to produce accessible web content; additionally, it is argued that the cognitive efforts to create accessible web content are too much to handle for current web development workflows. Therefore, the first part of the research project documented in this PhD thesis proposes an innovative approach to *make web accessibility more accessible* by means of three different components to handle three different, yet related aspects:

- the **AX** framework, which in contrast to current web technologies allows the generation of accessible markup *by default*; whenever information that is necessary to generate such markup is not provided, the corresponding elements are rendered in a way that makes it obvious to understand that something is wrong, and what the problem is;
- an innovative approach to manual accessibility testing, which maps (sometimes complex) concepts related to web accessibility on top of concepts (such as mouse operability and the visual appearance) that web developers are already familiar with; this methodology is then implemented in the **SAHARIAN** browser extension available for **Google Chrome**;
- **A11A**, a structured, categorized repository of accessibility-related resources to make it easier for stakeholders to find and effectively leverage them to produce accessible web content.

The second branch of the research project documented in this PhD thesis

aims to propose two innovative approaches to make indoor environments and scientific, manipulable artifacts more accessible by leveraging digital technologies while providing features to make these products attractive for the general public as well, in a truly universal design fashion. It is argued that indoor environments could be made more accessible for blind and visually impaired people by means of *accessible virtual tours*; therefore, the prototype of a system to let users enjoy such tours has been created, keeping into account both the specific requirements of blind and visually impaired people in order to explore indoor environments they are not familiar with, and the ones of their sighted peers to find such a system actually useful. The other problem tackled in the second branch of this research project, about making scientific manipulable artifacts with a high historical value more accessible, arises from the consideration that there are different objects with these characteristics lying around; unfortunately, while being beautiful and useful artifacts, they often get discarded or made inaccessible (both to the general public and blind and visually impaired people) to preserve them, even if they were initially designed to be touched and interacted with. This is the reason why the Scientific Collections Accessibility Making Process (SCAMP) methodology is proposed, a generic methodology to make such objects *accessible again* by means of digital technologies but *preserving* their physical nature; more specifically, it consists in a theoretical and practical *modelling* process involving the usage of 3D-printing and multimedia technologies to *enhance* the specificities of modelled objects by means of the creation of interactive multimodal experiences.

The remaining of this PhD thesis is organized as follows. In Chapter 2 the main resources that are related to web accessibility will be described, analyzing the reasons why their creation has been deemed necessary and the reasons why they are important. More specifically, Section 2.1 discusses the various standards that make web accessibility actually possible and how they cooperate to achieve the desired result across different abstraction layers. In Section 2.2 various laws and regulations to enforce the creation of

accessible websites and applications in different scenarios enacted by different national and international institutions will be examined, illustrating their most significant characteristics. To complete the analysis of existing accessibility resources, in Section 2.3 an overview of accessibility-related resources to support stakeholders in producing web content that is accessible, as well as testing and evaluating its accessibility, will be provided.

In Chapter 3 the current situation of digital accessibility and the reasons leading to it will be discussed. More specifically, in Section 3.1 the current status of digital accessibility will be discussed, illustrating of the majority of web content available on the Internet still exhibits critical accessibility issues. After that, in Section 3.2 the reasons leading to the current situation will be investigated: even if there are no absolute certainties in this area, there are various relevant evidences that are worth considering. Finally, in Section 3.3 the current situation about the usage of digital technologies to make indoor environments and cultural heritage more accessible will be discussed, highlighting the most significant experiences in these areas.

After that, in Chapter 4 an overview of the research project documented in this PhD thesis will be provided. In Section 4.1 a high level overview of the research project, the produced artifacts, and the expected outcomes will be given. As a large part of this research project is based upon the premise that currently *web accessibility is not accessible*, in Section 4.2 the reasons that lead to this consideration will be explained. Then, in Section 4.3 the proposed approach to *make web accessibility more accessible* by rethinking from the ground up the way in which it is dealt with nowadays will be illustrated. In Section 4.4 to particular artifacts thought to make web accessibility more accessible as well will be introduced: the **ARIA-Service JavaScript** library (to help developers implementing the required keyboard navigation support for WAI-ARIA roles), and the Missing WAI-ARIA Role Explorer (MARE) (an interactive tool to make information about the WAI-ARIA specification easier to find and understand). Finally, in Section 4.5 the SCAMP methodology and the A11YVT will be described. Manipulable artifacts more

In Chapter 5 some technical details about the implementation of all produced artifacts will be discussed, including the most significant challenges faced in the process and the solutions adopted to overcome them. More specifically, in Section 5.1 the implementation process for the **AX** framework will be described; the same is done in Section 5.2 for the proposed accessibility testing and evaluation methodology implemented by means of the **SAHARIAN** browser extension, in Section 5.3 for the **A11A** website, and in Section 5.4 for the **ARIA-Service JavaScript** library and **MARE**. In Section 5.5 the creation of *A11YExamples* (a repository containing various accessibility examples with different characteristics) will be discussed, illustrating the reasons why it has been deemed necessary. Finally, in Section 5.6 the prototype implementations for **A11YVT** and the **SCAMP** methodology will be illustrated.

Next, in Chapter 6 the evaluation process that has been performed to assess the produced artifacts will be described, discussing the obtained reasons. In particular, in Section 6.1 the performed evaluation process will be described, illustrating how usability tests have been structured and the reasons that lead to the decision. In Section 6.2 the main artifacts produced to implement the proposed approach to *make web accessibility more accessible* will be discussed: in particular, the **AX** framework and the **SAHARIAN** browser extension have been evaluated. Finally, in Section 6.3 the evaluation process for the **SCAMP** methodology and the **glsa11yvt** prototype implementations will be described, discussing the obtained results.

Finally, in Chapter 7 some final considerations are drawn, highlighting various opportunities for future developments and further research questions opened by the research project documented in this PhD thesis.



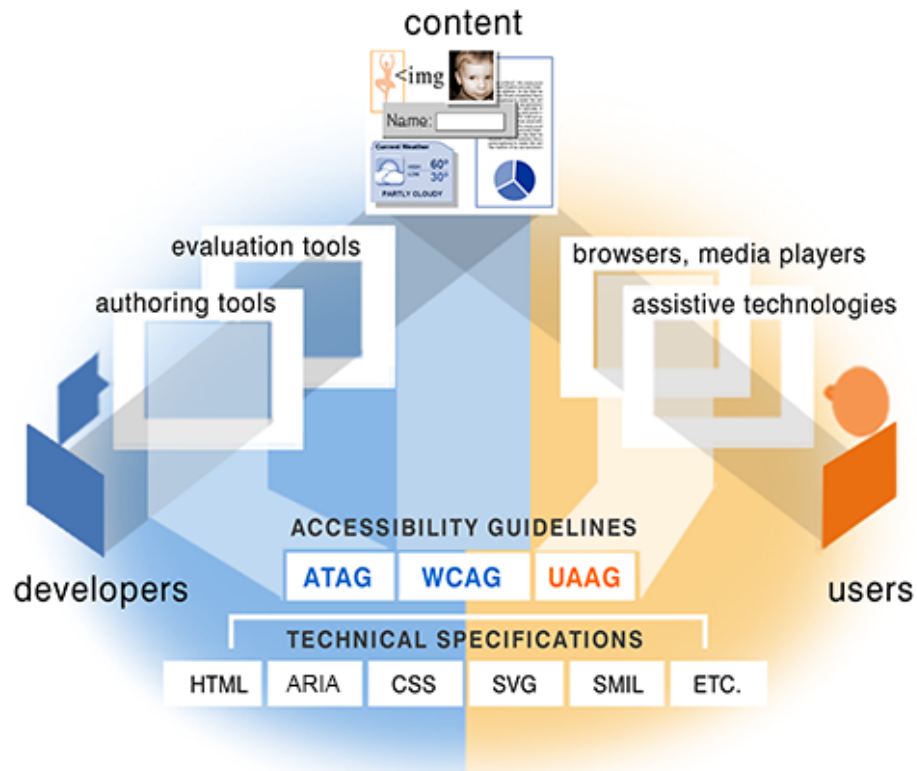
# Chapter 2

## An overview of accessibility resources

In order to illustrate the purpose of this research project, it is important to explain the context behind web accessibility, how “things” work and the reason why some things are nowadays “the way they are”. In this chapter the main resources that are related to web accessibility will be described, analyzing the reasons why their creation has been deemed necessary and the reasons why they are important.

### 2.1 Web accessibility standards and guidelines

Web accessibility is made possible by a complex equation that conjugates many heterogeneous aspects on two different, yet strictly connected levels: the content and all of its characteristics, and the end-to-end infrastructure that makes it possible to produce, deliver, distribute and consume such content. Just focusing on the infrastructure, this equation involves web technologies, user agents (such as web browsers), operating systems, assistive technologies and the ways in which they communicate. If we start considering content, the number of variables to be considered increases a lot: nowadays, we are surrounded by web content available through many differ-



A diagram showing various accessibility-related specifications, highlighting their relationship with their targets and the technologies they refer to.

Source: [Acc22a]

Figure 2.1: Accessibility specifications and their targets

ent means (e.g. web pages, mobile applications, other platforms) and in a wide variety of forms (e.g. text, images, pictures, videos, audio, and more). People are also involved in the equation: authors and web developers, who can make very significant decisions that have a very significant impact on the accessibility of the end results, and users, who may consume such content with a variety of assistive technologies, devices and user agents. Figure 2.1 illustrates some components involved in this complex equation, along with different specifications that have been created to *regulate* their involvement in making web content accessible.

Since its origin, the World Wide Web Consortium (W3C) has played a



very significant role in the history of the web, its evolution, and the standardization process of many technologies that are critical to its success. This is the case for web accessibility as well: over the years, in fact, the W3C through the WWeb Accessibility Initiative (WAI) <sup>1</sup> has produced a lot of accessibility-related documents. Resources produced by the W3C and its working groups can be classified in various groups, according to different criteria [Wora]:

- their type (recommendation, notes, guides, ETC.);
- their status in the publication process, keeping into account that each type of resource undergoes a different publication process;
- their purpose, distinguishing between normative resources (technical standards that have to be respected by the parties they are aimed at) and informative ones (such as guides and tutorials, documents that try to explain the more technical resources and the ratio behind technical standards, tutorials to provide examples and best practices, and so on).

When it comes to web accessibility, both normative and informative resources from the W3C should be examined to get a complete picture of the context in which this research project originates.

### 2.1.1 WCAG 2.1

Now at version 2.1, *Web Content Accessibility Guidelines (WCAG)* [CCKC18] is perhaps the most important resource, as it lies the foundation for web accessibility. It provides a set of *characteristics* that web content should satisfy in order to be considered accessible. These characteristics can be classified in three different layers, according to their level of abstraction. At the most abstract level, you can find the four fundamental principles that inspire the whole standard:

---

<sup>1</sup><https://www.w3.org/wai/>

**Perceivable** Information and user interface components must be presentable to users in ways they can perceive. In other words, users must be able to perceive the information being presented (and therefore it cannot be invisible to all of their senses).

**Operable** User interface components and navigation must be operable, meaning that users must be able to operate the interface (and therefore it cannot require interaction that a user cannot perform).

**Understandable** Information and the operation of user interface must be understandable. Users must be able to understand the information as well as the operation of the user interface (thus the content or operation cannot be beyond their understanding).

**Robust** Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies. Users must be able to access the content as technologies advance (therefore, as technologies and user agents evolve, the content should remain accessible).

Often abbreviated with the acronym *POUR* (from merging the first letters of the words Perceivable, Operable, Understandable and Robust), these are the fundamental principles for accessible content. If any of these principles is not respected, users with disabilities will not be able to use the Web. This is clearly explained in "Understanding" WCAG 2.1 [Acc22c], a support document from the W3C aimed at illustrating the general purpose, the structure and how to interpret WCAG 2.1.

At a lower level, for each principle WCAG 2.1 provides a set of guidelines that help to address that principle. While there are many general usability guidelines to make content more usable by all people, WCAG 2.1 only includes those guidelines that address problems only experienced by people with disabilities<sup>2</sup>. WCAG contains a total of twelve guidelines that have been

---

<sup>2</sup>While an in depth discussion about the differences between usability and accessibility

written to be *abstract enough* to be applied to various present and future web technologies, yet *practical enough* to be implemented in practice leveraging present and future technologies.

Finally, at the third layer WCAG 2.1 provides a set of *success criteria* for each guideline, describing specifically the goal that must be achieved in order to conform to the standard. Each Success Criterion is written as a statement that can be either true or false when specific Web content is tested against it; the Success Criteria are written to be technology neutral.

Depending on which and how many success criteria are satisfied, web content conformance to WCAG can be expressed in three different levels:

- Level A:
  - it is the lowest possible level of conformance;
  - removes major barriers for blindness, deafness and motor disabilities.
- Level AA:
  - being the next level of conformance, implies that content also conform to WCAG level A;
  - removes major barriers for low vision users;
  - offers a little help for cognitive disabilities and learning difficulties.
- Level AAA:
  - it is the highest possible level of conformance, and implies that content also conforms to WCAG level A and AA;
  - while this may seem counter-intuitive, it is not recommended requiring this conformance level as a general policy for entire sites

---

is out of scope for this PhD thesis, it is worth mentioning that in recent years there have been a lot of discussion about the subject both in the literature and the industry. This discussion has lead to even design philosophies, such as universal design and inclusive design, that fully embrace both aspects as a single one.

because it is not possible to satisfy all Level AAA requirements for some content.

At the time of writing this PhD thesis<sup>3</sup>, an update for WCAG (version 2.2 [ACM<sup>+</sup>22]) is ready as a *candidate recommendation snapshot*, and thus it is in the latest stages of the W3C standards publication process [Wora]. WCAG 2.2 will bring twelve new success criteria regarding navigability, input modalities, predictability and input assistance guidelines.

With WCAG 2.1 being technology-neutral, additional documents explaining how to comply with them in specific usage scenarios (such as when leveraging specific web technologies) are necessary. The document [Acc22b] is perhaps the most popular example of such documents, as it illustrates how to satisfy WCAG 2.1 success criteria while using specific web technologies. Notably, among others it provides:

- client-side script techniques, that explain how to meet WCAG 2.1 success criteria while using client-side scripting languages (such as JavaScript [ECM99]) to implement dynamic features in a web page;
- CSS Techniques, that explain how to meet WCAG 2.1 success criteria with Cascading StyleSheets (CSS);
- common failures, that document some of the most common failures on meeting WCAG 2.1 success criteria you can find in web applications, explaining the reason why they are problematic and often providing more accessible alternatives;
- HTML techniques, that cover specific aspects on how to meet WCAG 2.1 success criteria while using HyperText Markup Language (HTML);
- Server-Side script techniques, that explain how to improve web accessibility by leveraging server-side languages;

---

<sup>3</sup>This PhD thesis was written in between December 2022 and January 2023, but is expected to be published in mid-late 2023.

Given its nature, techniques for WCAG 2.1 [Acc22b] is not (and perhaps will never be) exhaustive, and it is subject to change frequently over time; in addition to this, it is not meant as a standalone resource, but rather as a support document to be used in conjunction with other accessibility-related resources.

Due to its structure, understanding WCAG 2.1 and how to implement it to produce accessible web pages and applications might not be immediate; this is the reason why several supporting documents to explain various aspects of this specification have been produced by the W3C as well. The document *WCAG 2.1 at a glance* [Wore], for instance, paraphrases the specification in order to make its content more “developer friendly”. Other documents, such as *understanding WCAG 2.1* [Acc22c], are aimed at explaining why the specification is structured the way it is, and how to use the various documents that have been produced to support it and get the most out of them. Finally, the tool *how to meet WCAG 2.1* [Word] provides an interactive, customizable view of all WCAG 2.1 principles, guidelines and success criteria that can be customized depending on the specific use case, allowing to visualize only information that is relevant to its requirements.

### 2.1.2 ATAG 2.0

Unfortunately, there can be situations in which WCAG and its supporting documents might not suffice to create truly accessible content. This is the reason why, in some cases, more specific standards and guidelines (along with additional supporting documents) have been produced over the years. Recognizing the importance of authoring tools<sup>4</sup>, W3C created Authoring Tools Accessibility Guidelines (ATAG) 2.0 [CCKC15], a specification aimed at ensuring the accessibility of these tools and all content produced by them. Like WCAG 2.1, ATAG 2.0 is organized around some fundamental princi-

---

<sup>4</sup>All software and services, including the ones that implement fully-automated processes, commonly used by “authors” (such as web developers, designers, writers and content creators) to produce web content (static web pages, dynamic web applications, ETC.)

ples, with different guidelines aimed at addressing them by specific tools. In particular, the ATAG 2.0 specification can be divided in two parts:

- Part A, aimed at ensuring that authoring tools themselves are accessible; this basically states that authoring tools user interface should comply with WCAG 2.1.
- Part B, aimed at ensuring that authoring tools support the production of accessible content. Guidelines to achieve this goal are centered around four fundamental principles:

B.1 Fully automatic processes produce accessible content. This principle has many practical implications; on one hand, for instance, if an automatic process detects WCAG violations in the content being processed, it should “require” the user to perform additional tasks to fix the detected issues; on the other, it means that such processes should preserve all accessibility-related information, so that the output document cannot be less accessible than the input version.

B.2 Authors are supported in producing accessible content. Once again, this principle has many practical implications: in fact, authoring tools should ensure that the production of accessible content is possible (providing all the necessary features), guide authors to produce accessible content, and assist them with managing alternative text for non-textual content (such as images, audio and video), accessible templates and pre-authored content.

B.3 Authors are supported in improving the accessibility of existing content. Authoring tools should assist authors in checking for accessibility issues in existing content and fixing them.

B.4 Authoring tools promote and integrate their accessibility features. This means that authoring tools should ensure that the features necessary for the production of accessible content are available,

and documentation promotes the creation of accessible content as much as possible.

Similarly to what happens with WCAG 2.1, for each principle in ATAG different guidelines addressing it and success criteria to test whether each guideline is “respected” or not are also provided. Authoring tools conformance to ATAG can be of two different types:

- full conformance, when the authoring tool can produce accessible web content without leveraging additional tools or components;
- partial conformance, when the authoring tool would require additional tools or components in order to conform as a complete authoring system.

In both cases, conformance to ATAG 2.0 can be expressed in three different levels (level A, level AA and level AAA), with each one implying that the tools satisfies the previous.

In order to help authoring tools developers with making their products conform to ATAG, the W3C provides two different supporting documents. The first one, *Implementing ATAG 2.0* [RST15], is a guide to understand the content of ATAG 2.0 and implement it correctly. The second one, *ATAG at a glance* [Worc], is a document that paraphrases ATAG 2.0 in order to make its content more understandable for web developers and content authors.

### 2.1.3 WAI-ARIA 1.1

Speaking of web applications, the advent of *Asynchronous JavaScript and XML (AJAX)* raised several challenges related to web accessibility. In fact, this set of technologies made it possible to create web applications on a completely different level of complexity, having user interfaces and features more similar to the ones you could find in desktop applications rather than what had been available on the Internet until then. In addition to this,

AJAX makes it possible to break some of the fundamental assumptions that the World Wide Web (WWW) had been based upon before of its advent:

- an update to the page requires a full *page refresh*, meaning that the web page has to be loaded again entirely for changes to be reflected on the screen;
- navigation actions (such as clicking on a link) originate *full page reloads*, meaning that they load new pages, or reload the existing one entirely;
- once the page is loaded, except for minimal aesthetic effects (think of the `marquee` HTML element), its content cannot be changed without reloading it.

AJAX broke all these assumptions, making it possible to *partially update* a web page: once the initial HTML is loaded, in fact, `JavaScript` code can “take control” and change the page dynamically in many unpredictable ways. It can even send additional HTTP requests to fetch new resources and display additional content on the same page. It is no longer true that navigation actions originate full page reloads, as `JavaScript` code can intercept native browser events and implement any custom business logic to handle them.

With the purpose of making this kind of web applications accessible, along with giving developers an opportunity to *enrich* the semantics of more traditional web pages, the Active Rich Internet Applications (WAI-ARIA) 1.1 specification [DMC<sup>+</sup>17] was born. It provides several attributes to be used with various web technologies to *expose* information that are very useful to assistive technologies (such as points of a page that change dynamically) that could not be exposed otherwise, as well as providing various mechanisms to *represent* the semantics of various user interface controls (from basic ones such as buttons and links, to the most complex of them like treeviews and grids) to assistive technologies. This can be done by leveraging various attributes of three different types:

- *Roles*, which indicate the type of the element they are applied to. This mechanism is very powerful, as it allows developers to *change* the se-



antics of elements (for instance, informing assistive technologies that a static `span` HTML element will behave like a button). Such association allows assistive technologies to present (and support all interactions with) the object in a manner that is consistent with user expectations about other native elements of that type.

- *Properties and states*<sup>5</sup>, which further refine the semantics of the specified *role* or provide additional information that is necessary to expose correctly to assistive technologies (for instance, think of the “slider” role to represent a slider, and the `aria-valuemin`, `aria-valuemax` and `aria-valuenow` attributes to specify its minimum, maximum and current value, respectively).

To support developers in producing accessible websites and applications conforming to the WAI-ARIA specification, the document *WAI-ARIA 1.1 Authoring Practices* [KKN<sup>+</sup>22] has been produced by the W3C. It explains the purposes of the specification, how to use it in specific scenarios, and even provides a lot of various code examples to demonstrate how to leverage its features effectively. The document has been demoted from a *working group note* on May 19, 2022, but is now superseded by the document *WAI-ARIA Practices Guide (APG)* [Wor22a]<sup>6</sup> maintained by the W3C as well.

Given the characteristics of the WAI-ARIA specification, various documents have been produced to better describe its usage in so-called *host languages* (or, in other words, all markup languages with which it can be used) and in more specific contexts where it can be useful. The W3C recommendation *ARIA in HTML* [FOL22] describes how it should be integrated within the HTML markup language [Web22c]. On the other hand, the specification

---

<sup>5</sup>There are subtle differences that allow to distinguish states from properties, but there are many exceptions even to that rule; therefore, in practice the differences between them becomes irrelevant. In the guide *Rules of ARIA* [Wor22d], indeed it is suggested to consider them as “attributes whose name begins with the prefix `aria-`”

<sup>6</sup>This change has been deemed necessary “to make the document more useful for developers”, and let it evolve much faster over time.

*WAI-ARIA graphics module* [BRDC<sup>+</sup>18b] further enhances the specification by adding various attributes to enable the creation of accessible complex graphics and diagrams. Finally, the *WAI-ARIA Digital Publishing Module 1.0* specification [GSGM17] allows embedding semantic metadata about long-form Web document structural divisions by means of WAI-ARIA roles specific to helping users of assistive technologies navigate through such documents.

#### 2.1.4 Assessing conformance to accessibility standards

Given the structure, complexity and abundance of web accessibility standards and guidelines, along with the heterogeneity of content they can apply to, assessing conformance to such standards is not an easy task. Therefore, several attempts to standardize both the testing and evaluation process and expressing such conformance have been made over the years.

Standardized as a W3C working group note, Web Accessibility Conformance Evaluation Methodology (WCAG-EM) [VAZ14] is an approach for determining how well a website conforms to WCAG. It is primarily intended for application to existing web content, and requires a thorough (and in some cases expensive) review of all content whose conformance to WCAG have to be assessed. WCAG-EM can be applied to all websites and applications, for both desktop and mobile platforms. It covers different usage situations, including self-assessment and third-party evaluation; WCAG-EM is independent of particular testing and evaluation tools, web browsers, operating systems or assistive technologies. The main evaluation procedure can be detailed into five main steps:

1. Define the scope of the evaluation. It should be established:
  - what is included in the evaluation;
  - the goal of the evaluation;
  - the WCAG conformance level (A, AA, AAA) against which content should be checked.

2. Explore the website. This includes identifying:
  - key web pages;
  - key functionality;
  - types of web content, designs, functionality, and so on;
  - required web technologies.
3. Select a representative sample. Guidance on how to select a random, yet structured sample of web pages when it is not feasible to evaluate every web page on the website is also provided.
4. Evaluate the selected sample. This is done by:
  - determining successes and failures in meeting WCAG success criteria;
  - determining accessibility support for website features;
  - recording all evaluation steps, possibly making the evaluation reproducible.
5. Report the evaluation findings. This requires:
  - aggregating and reporting evaluation findings;
  - making evaluation statements;
  - calculating overall scores.

The web service *WCAG-EM Report Tool* [Wor22e] has been by the W3C to guide users through all the steps of the WCAG-EM evaluation process, allowing to store all data to stop and resume the process at different times and generate the final report (that can be exported in various formats). The *ATAG Report Tool* [Wor22b], a similar service providing the same features, is available for guiding people through the test and evaluation process of authoring tools to assess their conformance to the ATAG 2.0 specification.

Testing and evaluating the accessibility of web content, though, can be automated to some extent; all the described methodologies, however, are not sufficient for implementing such tools. Therefore, an even more detailed set of rules to verify web content conformance to the WCAG 2.1 specification is provided by the resource *Accessibility conformance testing (ACT) rules* [Worb]. These rules are very specific to web technologies and can be seen as checks that can be directly performed on the Document Object Model (DOM) representation of the page being evaluated. Rules included in this resource undergo a severe review process; they must reflect guidelines and success criteria already present in WCAG 2.1. To make sure they are machine-readable, the format for creating and storing these rules has also been standardized with the W3C recommendation *Accessibility Conformance Testing (ACT) Rules Format 1.0* [FKMAZ19].

In order to express the conformance of web content to accessibility-related specifications, the *Evaluation and Report Language (EARL) 1.0* [AZS17] can be used: it consists of a Resource Description Framework (RDF) vocabulary [CWL<sup>+</sup>14, CSB14, GSB14] aimed at the more generic purpose of assessing the conformance of a web page to a given specification. It can be used to express accessibility evaluation results in a machine-readable, platform-independent, and vendor-neutral format; describing test results in a standardized format facilitates the exchange of information between different types of tools that are used to evaluate Websites for accessibility [AZ06]. EARL, its syntax and how to leverage it effectively are documented in the *EARL 1.0 Developer Guide* [VAZK17]. Hilera et al. [HOTS<sup>+</sup>18] proposed a methodology to combine and compare different accessibility evaluation reports by employing semantic web technologies as well.

### 2.1.5 Making other content types accessible

As said at the beginning of section 2.1, though, web content is not made only of text elements in web pages or user interface controls in web applications; instead, a very wide variety of content types have been spreading over

the years. Therefore, it is critical to ensure that they can be made accessible as well. This is the reason why some specific standards to help authors produce such content in conformance with WCAG 2.1 have been developed. Some of them will be briefly discussed below.

*Synchronized Multimedia Integration Language (SMIL) 3.0* [BJC<sup>+</sup>08] is a markup language based on Extensible Markup Language (XML) to describe multimedia presentations. Among other things, it defines the markup necessary to provide information about timing, layout, animations, visual transitions, and media embedding. It can be used to represent media items such as text, images, video, audio, links to other SMIL presentations, and files from multiple web servers.

Informally known as the *standard for digital talking books*, the *the Digital Accessible Information System (DAISY) 3.0* [DAI12] specification allows the creation of books in which audio and text can be synchronized to provide features like text searching, bookmarks management, extensive navigation (e.g. by line, word, sentence, paragraph, and so on), and controlling the playback without adding distortion. It is based upon the SMIL 3.0 specification, and it is nowadays a very popular format used to make and distribute accessible books for people with disabilities.

The Timed Text Markup Language 2 (TTML2) [ACD<sup>+</sup>18] is a content type that represents timed text, textual information that is intrinsically or extrinsically associated with timing information. It can be used either directly, as a distribution format for content to be referenced by HTML pages or SMIL documents, or as an interchange format to support exchanging information among systems currently in use for subtitling and captioning audio and videos. Currently available as a candidate W3C recommendation, the Web Video Text Tracks Format (WebVTT) [PPJH19] is a format used for displaying timed text tracks (such as captions and subtitles) using the `<track>` HTML element. Unlike TTML2, WebVTT has been with the more specific goal of adding text overlays to HTML `<video>` elements.

Mathematical Markup Language (MathML) 3.0 [CIM<sup>+</sup>14] is a markup

language intended for describing mathematical notation, capturing both its structure and content. Quoting its specification, “the goal of MathML is to enable mathematics to be served, received, and processed on the World Wide Web, just as HTML has enabled this functionality for text”.

Speaking of long-form documents, it is worth mentioning a couple of specifications for two of the most commonly used formats for their distribution and consumption. With regard to the accessibility of *Portable Document Format (PDF)* [ISO08] files, whose popularity is undiscussed, the situation is pretty complex for various historical and technical reasons whose discussion is out of scope for this PhD thesis. However, the main specification aimed at producing such documents in conformance to WCAG 2.1 is the so-called *PDF UA* [ISO14] standard produced by the International Standard Organization (ISO). Regarding the more modern *Electronic Publication (EPUB)* [GHC22], which is very interesting from an accessibility point of view as it is built upon existing and very well known web technologies (HTML, XML, CSS, JavaScript for interactivity, Scalar Vector Graphics (SVG), and more), the specification *EPUB Accessibility 1.1* [GKL<sup>+</sup>22a] establishes content conformance requirements for verifying the accessibility of EPUB Publications, as well as defining accessibility metadata requirements for their discoverability. The document *EPUB Accessibility Techniques 1.1* [GKL<sup>+</sup>22b], then, provides guidance on how to meet such requirements, along with practical code examples demonstrating what is explained.

### 2.1.6 Supporting accessible content

In the complex equation that makes web accessibility possible, an important role is played by all the software that make the web itself possible. In particular, operating systems, web browsers and assistive technologies are the three main actors that should be considered in order to understand why certain things “are the way they are”.

With regard to operating systems, each one provides a so-called *accessibility Application Programming Interface (API)* whose main purpose is to

provide mechanisms to achieve two different, yet strictly related goals:

1. let other software applications (such as native applications running on the system itself) expose information about their user interface and the content they manage; and
2. let assistive technologies consume such information and *manipulate* those interfaces to perform all tasks that are necessary to let people with disabilities control them.

While an in depth analysis of all major operating systems accessibility APIs to examine how they work and compare their similarities and differences (explaining the reasoning behind certain decisions and implementations) would perhaps require a PhD thesis on its own, it is worth discussing a few aspects that can have a significant impact on web accessibility. In fact, user agents (such as web browsers and assistive technologies) *interact* very often to enable people with disabilities to browse the Web: simplifying for illustrative purposes, web browsers can be considered kind of a “bridge” between web content (thus what needs to be accessible), with its accessibility characteristics as expressed by web developers and content authors, and assistive technologies and the way they need to consume this information and manipulate such content (e.g. to let keyboard users activate a link element on a page). Given that all accessibility APIs are different, and assistive technologies available on each platform are even more heterogeneous, several aspects of this process have been standardized to ensure that:

- web developers can have *reasonable* expectations that, if web content is produced according to established accessibility standards, it will be accessible on all platforms;
- people with disabilities can expect that, if content is produced in conformance with established web accessibility standards, assistive technologies will be able to provide the features they need to consume such content and manipulate it.

User Agents Accessibility Guidelines (UAAG) 2.0 [ALPS15] is aimed at helping developers in designing user agents (browsers, browser extensions, media players, readers and other applications that render web content) that make the web more accessible to people with disabilities. Quoting the specification,

a user agent that follows UAAG 2.0 will improve accessibility through its own user interface and its ability to communicate with other technologies, including assistive technologies.

The document *Core Accessibility API Mappings 1.1* [DSS<sup>+</sup>17], soon to be superseded by the updated version 1.2 [DSC22] currently available as a W3C candidate recommendation snapshot, establishes how user agents (including web browsers) should map HTML elements to native elements provided by the accessibility APIs available on the operating systems they run on, and the different interactions they should enable to make their semantics *match* the ones of the corresponding native element on the platform.

Given its characteristics, and the power it gives to web developers, specific documents have been published to regulate how information exposed through attributes provided by the WAI-ARIA specification should be mapped to the semantics made available by accessibility APIs on each operating system. The *WAI-ARIA 1.0 User Agent Implementation Guide* [SCSWL14] provides such information for the specification, while the document [BRDC<sup>+</sup>18a] illustrates such mappings for the graphics module [BRDC<sup>+</sup>18b].

Finally, the W3C recommendation *Accessible Name and Description Computation 1.1* [DGC18] establishes several rules on how *accessible elements* should be identifiable by users through assistive technologies. To better explain this concept, the example of screen readers and speech recognition systems can be considered: for each element, meaning any element exposed through accessibility APIs, it is important to let users clearly and uniquely identify it and its purpose. This is done by *computing* two strings, the so-called *accessible name* and *accessible description*. The accessible name of an element is a short string that clearly identifies that element within the user



interface and should let users immediately understand the element's purpose; if appropriate, the accessible name should let the user know what happens if the element is activated. Examples of accessible names include the label of a button or text field, the text of a link, and the alternative text of an image. The accessible description of an element, on the other hand, is meant to *complement* its accessible name by providing additional yet very useful information; for example, in case of a text field, it may consist of additional information explaining how to fill it. Finally, the role of an element is a very short string that identifies the *semantics* of an element, including all supported interactions. For instance, encountering an element with the *checkbox* role, users expect to be able to toggle (enable and disable) it; encountering an element with the *button* role, users expect to be able to activate it to make something happen; when encountering elements with the *link* role, users expect to be able to activate them in order to navigate to a different page. The combination of accessible name and role should uniquely identify an element within a user interface; this enables screen readers, for instance, to immediately describe the element with this combination when it is focused or allows speech recognition systems to execute specific commands to focus or activate that element.

Unfortunately, computing the accessible name and accessible description of a web page element can be very complex, as there are many mechanisms that can contribute to its creation; therefore, the *Accessible Name and Description Computation 1.1* [DGC18] specification provides the pseudocode of an algorithm to compute them. Determining the role, on the other hand, is more straightforward: in general, in fact, it is equivalent to the type of the native accessible element (from the accessibility API) that its semantics is mapped to.

## 2.2 Laws and regulations

The World Wide Web and its technologies play a very significant role in our lives. Recognizing its importance, different national and international institutions have enacted various laws and regulations to enforce the creation of accessible websites and applications in different scenarios. In this section, some of the most important ones will be examined, illustrating their most significant aspects.

With its ratification by the United Nations in 2009, the Convention on Rights of Persons with Disabilities (CRPD) [Uni06a] marked a turning point in establishing the minimum standards for the rights of people with disabilities. More specifically, the convention recognizes the web accessibility as a central element of the right to an independent living of people with disabilities. In addition to this, its goals and agenda inspires many laws enacted by countries all over the world to improve the life of people with disabilities. Through an optional protocol, [Uni06b]it also regulates the establishment of the Committee on the Rights of Persons with Disabilities (UNCRPD Committee).

The European Union (EU) is deeply aware of this issue and has built up a considerable corpus of disability laws and policies over the past three decades [FF18]. In line with the Charter of Fundamental Rights of EU, the European Commission unveiled the European Disability Strategy (EDS) 2010–2020 [Com10], later renewed in the Strategy 2021–2030 [Com21], which aims to ensure the full participation of persons with disabilities in society on an equal basis. To achieve this goal, it proposes a multi-year policy framework for the practical implementation of the United Nations’ Convention on Rights of Persons with Disabilities (UNCRPD). In this context, EU web accessibility regulations arise from two main directives. The first one, the *Directive (EU) 2016/2102* [Eur16] is intended to improve the functioning of the internal market, allowing websites and mobile applications of public bodies to be more accessible. It establishes minimum accessibility criteria (“strongly” inspired by WCAG 2.1 conformance level AA) that must be sat-

ified by websites and mobile applications developed by (or on behalf of) public entities such as administrative courts, police departments, public hospitals, universities, schools, and so on; it also encourages Member States to make more far-reaching regulations and extend the application to private entities. Also known as “European Accessibility Act (EAA)”, the directive (EU) 2019/882 [Eur19] is the second directive passed by the European Parliament and the Council in 2019 to improve the functioning of the internal market for accessible products and services by removing barriers created by divergent rules in Member States on that matter. According to the directive, by doing this both businesses and people with disabilities (and the elderly population) would benefit from significant advantages. On one hand, people with disabilities and elderly people should benefit from:

- more accessible products and services in the market;
- accessible products and services at more competitive prices;
- fewer barriers when accessing transport, education and the open labor market;
- more jobs available where accessibility expertise is needed.

On the other hand, businesses should benefit from:

- common rules on accessibility in the EU, leading to costs reduction;
- easier cross-border trading;
- more market opportunities for their accessible products and services.

EAA is very ambitious in its goals. In fact, products and services covered by the directive are the ones that have been identified (after public consultation with both stakeholders and accessibility experts) as being most important for persons with disabilities while being most likely to have diverging accessibility requirements across EU countries. Notably, these include:

- computers and operating systems;

- ATMs, ticketing and check-in machines;
- smartphones;
- TV equipment related to digital television services;
- telephony services and related equipment;
- access to audiovisual media services such as television broadcast and related consumer equipment;
- services related to air, bus, rail and waterborne passenger transport;
- banking services;
- e-books;
- e-commerce.

It is worth noting that EAA also applies to private bodies with certain characteristics; aligning with the UNCRPD, it establishes functional accessibility requirements without specifying technical implementations, thus allowing for further innovation and flexibility in the future.

When it comes to country-specific laws, some focus strongly on web accessibility, while others fall slightly behind [YC15]. Enacted in 2004, *Stanca Act* [Ita04] is an Italian law stating that “the Government must protect the right of every individual to access information sources and services regardless of disability, in line with the principles of equality established by the Italian Constitution”. It promotes accessibility of information technology. The law applies to Italian government websites, as well as other web services developed by (or on behalf of) public bodies in the country. The law has been amended several times over the years, and after the 2018 amendment it implements the latest EU directives related to web accessibility.

With regard to the United Kingdom (UK), the *Equality Act* of 2010 [Gov10] protects people in the United Kingdom from discrimination based

on various characteristics (including age, gender, disability) in the workplace and in wider society. It is a very comprehensive law that regulates many different, yet related, aspects about discrimination including definitions, complaint procedures, and duties. It applies to both public bodies and private organizations. More specifically, regarding web accessibility, the UK Equality Act requires that private website owners do not discriminate against or disadvantage people with disabilities, therefore enforcing the creation of accessible websites and applications.

In the United States (U.S.), normative references at the federal level about the rights of people with disabilities arise from two main acts. The first one, enacted in 1990, is the *Americans with Disabilities Act (ADA)* [U.S08b] and regulates accessibility at the federal level. It ensures equal opportunities for people with disabilities in the areas of employment, public services and commercial facilities. ADA was significantly amended in 2008 with the *ADA Amendments Act* [U.S08a] and acts as the foundation of additional, more specific U.S. laws and regulations enacted both by the federal government and single states. The second one, the *U.S. Rehabilitation Act* [U.S15] of 1973 was the first major federal legislation aimed at ensuring equality for people with disabilities; amended three times (in 1993, 1998, and 2015), two of its sections directly pertain to web accessibility:

- Section 504 is a civil rights law that protects qualified individuals from discrimination based on their disability. It applies to employers and organizations that receive financial assistance from federal departments and agencies, including hospitals, nursing homes, mental health centers;
- Section 508 mandates access for people with disabilities to Information and Communication Technologies (ICT) developed, purchased, maintained, or used by federal agencies, including websites. At the operational level, the law directs the federal agency responsible for disability design guidelines development (U.S. Access Board) to set mandatory standards clearly identifying what constitutes “accessible” electronic and information technology services.

In addition, eighteen States actually hold statutes specifically related to technology accessibility [SL18].

With regard to architectural barriers, in the U.S. the *Architectural Barriers Act (ABA)* [U.S94] stands as the first measure by the Congress to ensure access to the built environment for people with disabilities. The law requires that buildings or facilities that were designed, built, or altered with federal money, or leased by federal agencies after August 12, 1968, be accessible. It covers a wide range of facilities, including post offices, Veterans Affairs medical facilities, national parks, Social Security Administration offices, federal office buildings, U.S. courthouses, and federal prisons; it also applies to non-government facilities that have received federal funding. It is enforced through standards for accessible design that indicate where access is required and provide detailed specifications for ramps, parking, doors, elevators, restrooms, assistive listening systems, fire alarms, signs, and other accessible building elements. Facilities covered by the ABA must meet these standards.

In Canada, it is worth mentioning two different laws. Enacted in 2005 and amended different times, Accessibility for Ontarians with Disabilities Act (AODA) [Gov05] is a Canadian provincial law designed to improve access for Ontarian citizens with disabilities, giving them the opportunity to fully participate in all aspects of daily life. Building on earlier laws, AODA sets high standards in order to achieve its goal of an accessible Ontario, and that includes access to ICT systems. Regardless of size or industry, AODA applies to all organizations registered in Ontario and establishes accessibility standards in five different areas:

- customer Service;
- employment;
- information and Communications;
- transportation;
- public Spaces.

AODA requires that public web content created after 2012 (including websites, applications, and digital documents) must meet the technical requirements of the WCAG W3C specification. Enacted in 2019 after public consultation, the *Accessible Canada Act* [Gov19] is a federal Canadian bill whose purpose is to identify, remove, and prevent accessibility barriers in various areas, including:

- built environments, including buildings and public spaces;
- employment, including job opportunities and employment policies and practices;
- information and communication technologies, including digital content and technologies used to access it;
- procurement of goods and services;
- delivering programs and services;
- transportation, including air, rail, ferry, and bus carriers that operate across a provincial or federal border.

The bill has been crafted to achieve a number of specific goals, including:

- inherent dignity;
- equal opportunity;
- barrier-free government;
- autonomy;
- inclusive design;
- meaningful involvement.

Therefore, it gives the Government of Canada the ability to work with stakeholders and people with disabilities to create new accessibility standards and

regulations for sectors under federal jurisdiction; these sectors include banking, telecommunications, transportation, and the Government of Canada itself.

Passed in 1992 and last amended in 2018, the Disability Discrimination Act (DDA) [Gov18] is a comprehensive Australian Act whose goal is to prevent discriminating against a person in many areas of public life, including employment, education, getting or using services, renting or buying a house or unit, and accessing public places, because of their disability. Quoting the Australian Human Rights Commission, the DDA covers people who have temporary and permanent disabilities; physical, intellectual, sensory, neurological, learning and psychosocial disabilities, diseases or illnesses, physical disfigurement, medical conditions, and work-related injuries. It extends to disabilities that people have had in the past and potential future disabilities, as well as disabilities that people are assumed to have. It has a significant impact on web accessibility as well, as it requires government agencies and their websites to provide information and services without discrimination. Today they are also required to meet WCAG at Level AA.



## 2.3 Accessibility resources

Given the characteristics of accessibility standards and guidelines illustrated in Section 2.1, as well as the different laws and regulations enforcing the creation of accessible content as explained in Section 2.2, it is clear that many stakeholders should be interested in accessibility-related resources. Having many heterogeneous interests and goals, these stakeholders differ a lot from each other; thus they have very different expectations from accessibility resources. Therefore, it is not surprising that a plethora of accessibility resources have been produced over the years, trying to maximize their impact and effectiveness for one or more of these target segments by satisfying such specific expectations in the best possible way.

### 2.3.1 Resources for accessible content production

Accessibility resources described in Section 2.1 can be very useful, as they provide a lot of information to produce accessible content at any level, in any phase of the development process. But making sense of all the information contained in these incredibly valuable resources, as well as how each one fits into the big picture, necessitates a thorough understanding of “how things work under the hood” as well as hyper specialized knowledge of complex concepts that is difficult to obtain [RVL22]. As a result, a plethora of instructional materials have sprung up to assist developers in creating accessible content, understanding all of these standards, and providing more hands-on, practical material. Examples are the ones made available by *Web Accessibility in Mind (WebAIM)* [Web22b], *Deque Systems* [Deq22c], *TPGi* [TPG22b], and many blogs curated by members of the accessibility community.

Various articles in the literature attempted to present tools and methodologies to help stakeholders to *integrate* accessibility in the development processes in order to ensure that the produced artifacts are accessible. Hadadi [Had21] proposed *adee*, a plugin that can be integrated within tools com-

monly used by designers to create mockups for web interfaces; it highlights accessibility issues (related to target sizes, color contrast, font sizes, and more) from the earliest phases of the development process, when the project to be realized is still an idea. Guarino Reid and Snow-Weaver [GRSW09] proposed a work to better illustrate how to apply accessibility guidelines in the design phase; the *color palette analyzer* [Deq22b] by Deque Systems, INC., the *color contrast checker* [Web16] and *link color contrast checker* [Web22d] by WebAIM, *Accessible Colour Evaluator (ACE)* [TFA17], or the system by Sandnes [San21] can help designers analyze the chosen colors for an interface, eventually suggesting similar colors with an improved contrast ratio if necessary to comply with the desired WCAG conformance level. Martín et al. [MRCG10] proposed an innovative approach to *engineer* accessible web apps from the conceptual phases of the development process. Arrue et al. [AVA08] examined how heterogeneous accessibility guidelines can be integrated in the development process of a website. Romero-Chacón et al. [RCMCRG<sup>+</sup>19] proposed an adaptation of the *scrum* development process [Sch97, SS11] to develop accessible websites; Sanchez-Gordon and Luján-Mora [SGLM17] proposed a method to integrate accessibility testing in those environments employing agile development processes. Fogli et al. [FPB10] proposed a design pattern language to document and implement accessible websites. Rainville-Pitt and D'amour [RPD09] demonstrated the impact that a Content Management System (CMS) can have on the creation of an accessible website, yet realizing that the majority of such products do not conform to accessibility standards and guidelines.

Sometimes, resources targeted to very specific groups of people with certain interests have been produced, keeping into account their specific knowledge and competencies in order to better explain concepts that are related to web accessibility, thus help them make their products more accessible; this is the case for *A practical guide to improve web accessibility* by Ng [Ng17], aimed at libraries websites developers, *a framework for improving web accessibility and usability of open courseware sites* by Rodríguez et al. [RPCT17],

that proposes a conceptual framework and testing methodology to improve the accessibility of Open Educational Resources (OER) releases in the context of Open Course Ware (OCW) initiatives, the design methodology proposal by Navarrete and Luján-Mora [NLM18], that is targeted towards creating accessible open educational websites, the guidelines proposed by De Santana et al. [dSdOAB12] to eliminate web accessibility barriers from experienced by people with dyslexia, or *Game Accessibility Guidelines (GAG)* [Var18], a set of guidelines to develop accessible games based on web technologies.

Several attempts to *educate* people to produce accessible content from their beginnings in web development have been made. These include the ones documented by Kelly and El-Glaly [KEG21] for high school students, Katsanos et al. [KTTA12] for university students, and Wang [Wan12] for undergraduates. All three showed that this approach is effective in raising students' awareness of disabilities and leveraging their knowledge on accessibility while maintaining high interest in pursuing a computing degree. Several tools aimed at helping novices to perform accessibility testing and evaluation have been proposed as well, such as *Accessibility Evaluation Assistant (AEA)* [BP11] by Bailey and Pearson.

Some attempts to help content creators produce accessible multimedia content have also been made; these include the *Techniques for the publication of accessible multimedia content on the web* [AZMLM20] by Acosta et al. or collaborative efforts such as *TwitterA11Y* [GPM<sup>+</sup>20], a browser extension to improve the accessibility of images published on the Twitter social network by Gleason et al.

### 2.3.2 Resources for accessibility testing and evaluation

As it is very important to assess the conformance of web content to accessibility guidelines, many testing and evaluation tools have been developed for this purpose. There are many methods for testing and evaluating web content accessibility, ranging from checklists and guidelines to automated testing, including human testing with participants from different user groups

and hybrid solutions combining these strategies.

These tools can be grouped in two main macro-categories:

- Automated accessibility testing tools, which do not require human participation in order to *do their business*; once activated, they perform a fully automated process to test and evaluate the accessibility of a web page or application (in some cases evaluating more than one of them at the same time).
- Manual accessibility testing tools, which are meant to assist humans when performing *manual* accessibility testing and evaluation.

Hybrid solutions that try to combine features from both categories are being developed as well, constituting a new open and promising research branch.

Automated accessibility testing tools have been released in many flavors to meet the widest variety of requirements from various development workflows. Each flavor of accessibility testing tool has its own strengths and weaknesses that arise from their nature. For example:

**Web services.** Being web applications that run in the cloud and managed by a third party, they can be used without installing anything in the environment in which accessibility testing is performed. On the other hand, the fact that these services run in the cloud, implies that each evaluated page is uploaded to a third party, uncontrolled server; this can be a problem when the page is confidential, contains confidential data, or it is not desirable to upload its code to a third-party (even if trusted). In addition to this, testing *private* pages (such as the ones protected by authentication systems) may not be easy.

**Browser extensions.** As they are integrated within the web browser, testing and evaluating the accessibility of a web page can be more immediate. On the other hand, these extensions have to be installed before they can be used; they run in the environment used for accessibility

testing, with all the possible risks associated with any code being executed this way. While some of these extensions contain the logic to *do their business* locally, some of them just act as a *bridge* to a web service running in the cloud that actually perform the accessibility testing and passes back the results.

**Command line tools.** Generally speaking, they consist of standalone applications that can be run on one or more operating systems. Given their nature, they offer the flexibility to be integrated in more complex testing pipelines, such as the ones used in Continuous Integration (CI) and Test Driven Development (TDD) workflows. On the other hand, being executed in the same environment where accessibility testing is performed, they share the same security risks associated with executing any code in that environment. While some of these tools do their business *locally*, others act as a *bridge* to a web service running in the cloud that is responsible for performing the test and passing back the results.

**Frameworks and libraries.** As the name implies, they can be used to build entirely new tools on top of them; given their nature, they provide the maximum flexibility when it comes to being integrated in more complex testing pipelines (such as the ones used in TDD and CI workflows). The main disadvantage of this flexibility, of course, is their complexity; these tools often require programming skills to be integrated properly.

Generally speaking, automated accessibility testing tools provide a set of common features; however, their implementation along with some unique characteristics can differentiate them. It is very common for a single tool to be provided in all the described flavors to make its usage possible for the widest audience. While an exhaustive comparison of such tools would require an entire PhD thesis on this own, some of them and their unique characteristics will be discussed in this section.

Developed by the Inclusive Design Research Centre (IDRC), *achecker*

[GL10] is a web service that allows anyone to enter the URL of a publicly available web page that will be evaluated against the success criteria of various specifications (WCAG 2.1 level A, AA, and AAA, requirements arising from laws such as Stanca Act, and U.S. Section 508, ETC.). Similar free web services include *TAW* [Taw22], provided by the TawDist company, and *TotA11Y* [Kha22], developed by the Khan Academy. A similar web service, but provided as a commercial product by a company specialized in offering accessibility solutions, is *tenon.io* [Ten17]. It allows to evaluate a single page, or start a *crawler* to evaluate all pages reachable from the provided URL. In this category can be classified the tool *WCAG4All* [GP21] by Gaggi and Pederiva.

In general, these tools convey testing and evaluation results by means of a report that consists of a list of issues, in which each problem is accompanied by a short explanation to let the developer understand what is wrong, along with a pointer to find the issue in the source code of the evaluated page. A different approach is used by *Arc toolkit* [TPG22a] by TPGi and *Web Accessibility Evaluation Tools (WAVE)* [SW16] by WebAIM: while they are available in different flavors, when these tools are used as browser extensions they communicate accessibility issues by *injecting* specific icons around the affected elements. When such an icon is activated, more details about the problem are provided. Other times, such tools integrate their features within the developer console of the browser; this is the case of *lighthouse* [Goo20], developed by Google and included in the **Google Chrome** browser. Another browser extension available for accessibility testing is *hera-ffx* [FGGM09, FGM11] for Mozilla Firefox presented by Fuertes et al.

A particular case is the one of the *axe-core* framework [Deq22a]. Primarily developed by Deque Systems, INC., it consists of an automated accessibility testing engine implemented in **JavaScript** as an open source library, whose code can be executed in both client-side and server-side environments. An interesting characteristic of this framework is that it is implemented adopting the *zero false positives* principle: if an accessibility issue is identified

as such, it is certain that there is a problem to be fixed. Along with its permissive license, this enables `axe-core` to be used in all sorts of web development scenarios. Different products provided by the company employ this framework internally: these include a command line interface tool, and the *Axe Dev Tools* [Deq20] browser extension, which is available in a free version (providing only automated accessibility testing), and pro plans that make it possible to perform *guided* manual accessibility testing in combination with automated checks. Many accessibility testing tools developed by other companies (for instance, the *accessibility insights* browser extension from Microsoft [Mic22]) rely on this framework as well.

Given the abundance of accessibility testing and evaluation tools, the W3C has published several informative documents to help developers picking the ones that are most suitable for their needs. The document *Selecting Web accessibility evaluation tools* [Wor22c] describes the types of tools available and illustrates the main features that stakeholders can expect from them; the document *Web accessibility evaluation tools list* [Wor22f] contains a list of web accessibility evaluation tools with a short summary of their characteristics. Information in this list is not reviewed either by W3C or any third-party, and developers can *freely* contribute to the list by adding any tool. Since there are so many accessibility testing tools, each with its unique pros and cons, the *A11YTools* [Ada22] browser extension is an interesting idea: in fact, it combines many of such tools in a single package, so that the developer can install a single extension, yet get access to many tools at the same time.

Several automated accessibility evaluation tools have been proposed for long-form documents as well. Developed by the PDF/UA Foundation, *PDF Accessibility Checker (PAC)* [PDF21] is a free accessibility checker aimed at verifying the conformance of PDF document to the PDF/UA accessibility standard; Doblies et al. proposed *PDF Accessibility Validation Engine (PAVE)* [DSDH14], a comprehensive and free web application to identify accessibility issues in PDF documents and let users fix them. With regard to long-form documents in the EPUB format, Kim et al. presented an *automatic*

and semi-automatic two-tier Check System for EPUB accessibility [KPL17]; this is based on another work in which they introduced an *Accessibility Automatic inspector library for EPUB and its Components* [KL17]. In general, though, Eikebrokk et al. [EDK14] argue that the EPUB format may be more indicate to produce accessible long-form documents than PDF.

Different tools and frameworks have been developed to help developers test and evaluate the accessibility of mobile apps as well; these include *MATE* by Eler et al. [ERGF18], the guidelines by Ballantyne et al. [BJJ<sup>+</sup>18], the accessibility API proposed by Park et al. [PHB<sup>+</sup>19], the Enhanced UI Automator Viewer with improved accessibility evaluation features [PBS16] by Patil et al. the framework by Salehnamadi et al. [SAL<sup>+</sup>21], the testing API [DMDOFdAF17] by De Moura et al. and the features to support accessibility evaluation provided by the **Espresso** User Interface (UI) testing framework [Zel19]. Paiwa et al. proposed a checklist to evaluate both accessibility and usability of mobile apps [PBMZF20].



# Chapter 3

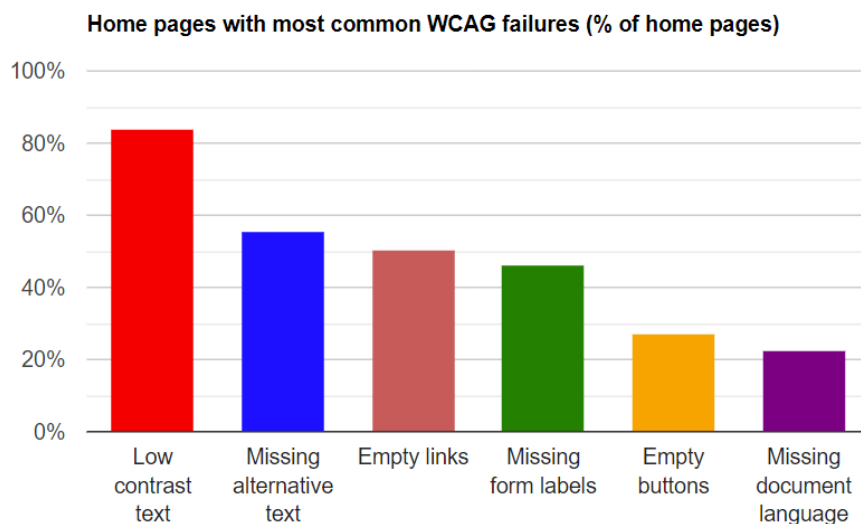
## The current status of digital accessibility

Having examined the main standards that make web accessibility possible, some laws and regulations influencing their adoption over the world, and a number of existing accessibility resources to help designers, developers, content authors, policymakers and all involved stakeholders produce accessible content, in this chapter the current situation of digital accessibility and the reasons leading to it will be discussed.

### 3.1 The current situation of web accessibility

As described in Section 2.1, there are many standards and guidelines to help designers, developer and content authors to produce accessible content; as explained in Section 2.3, even if in some circumstances these standards and guidelines can be difficult to work with, many resources to support the diverse and variegated target audiences of stakeholders have been produced. Nevertheless, it is very common to encounter web pages and applications that exhibit accessibility issues ranging from the most severe levels to the least severe ones.

Since 2019, WebAIM performs a yearly analysis of the most popular



A bar chart showing the main WCAG violation types detected during the home page analysis of the most popular 1,000,000 home pages.

Source: [Web22a]

Figure 3.1: Most common WCAG violation types in the top 1,000,000 home pages analyzed by WebAIM

1,000,000 home pages on the Internet. Some results demonstrate that the current situation with web accessibility is far from perfect. In fact, Across the one million of analyzed pages, 50,829,406 distinct accessibility errors (accessibility barriers detected by WAVE [SW16] having notable end user impact, and which have a very high likelihood of being WCAG 2 Level A or AA conformance failures) were detected, scoring an average of 50.8 errors per page [Web22a]. Figure 3.1 shows a bar chart representing the most frequent categories of WCAG violations detected in the analysis; the data shown in the chart can be read in Table A.1.

In many countries several laws and regulations enforce the creation of accessible websites and applications if they belong to certain categories (e.g. activities in the educational, health and essential personal services sectors) or are developed by (or on behalf of) specific subjects established by such laws (public bodies, with a trend towards including private subjects such

as service providers and large corporations). However, as demonstrated in this section many works in the literature demonstrate that such obligations do not always trigger the desired effect. Several studies have been made to analyze the accessibility of e-government websites, as well as the ones of health services companies, service providers, libraries, universities and other public bodies [MP22].

As there are a lot of studies evaluating the accessibility of websites, different selection criteria have been applied to decide the ones to be explicitly discussed in this PhD thesis; these include:

- only works published in 2008 or in newer years are considered;
- studies based on WCAG 1.0 have been excluded. WCAG 2.0 was released in 2008 and, being a major overhaul of the specification, it introduced significant differences with the previous 1.0 version;
- when multiple studies examined very similar websites, only a representative sample of them is discussed.

### 3.1.1 E-Government websites accessibility

Electronic Government (E-GOV) is defined as the Government use of web-based Internet applications or other information technology to enhance the access to and delivery of government information and services to the public, other agencies, and other Government entities; or to bring about improvements in Government operations that may include effectiveness, efficiency, service quality, or transformation. As will be discussed in this paragraph, many studies have analyzed these websites from different countries over the years, leading to the same conclusions: even if they should be accessible for both legal and ethical obligations, they are not.

Csontos and Heckl [CH21] evaluated the accessibility, usability and security of Hungarian E-GOV websites; results show that, inspite all applicable laws and regulations, none of the twentyfive websites of the examined Hungarian public sector bodies could completely fulfil the recommendations of

WCAG 2.1 and half of the websites had only the lowest level of compliance in usability tests. Furthermore, even if a lot of websites employ the so-called practice of *segregated web accessibility* (the practice consisting of creating a separate, accessible version of a website just for use by people with disabilities) which is proven to be bad for various reasons [LW11], even such theoretically accessible versions exhibit WCAG violations.

Kopackova et al. [KMC10] performed both manual and automated accessibility tests on thirty-nine local E-GOV websites in the Czech Republic in 2006 and 2008; unfortunately, results in 2008 were much worse than the previous ones, meaning that accessibility deteriorated.

Other studies about e-government websites accessibility in Europe include the ones by Buzzi et al. [BBR19], Valtolina and Fratus [VF22] and Gambino et al. [GPG16] for Italy, Basdekis et al. [BKMS10] for Greece, Akgül and Vatansever [AV16] for Turkey, Kuzma [Kuz10] about the UK, and Ilhan et al. [IIU20] in the Cyprus island.

King and Youngblood [KY16] analyzed the accessibility of e-government websites in Alabama, finding at least one WCAG level A violation per site.

Lujàn-Mora et al. [LMNP14] evaluated different e-government websites in South America with different automatic accessibility testing tools. Even if they used very strict selection criteria that excluded many possible issues, result show that all examined websites exhibit severe accessibility issues. Costa and Costa-Vargas [AAVLM18] performed both manual and automated accessibility testing on two websites, getting the same results: they did not conform to WCAG 2.1. Sanchez-Gordon et al. [SGLMSG20] performed a similar research to assess the accessibility of Ecuador's E-GOV websites. Serra et al. [SCF<sup>+</sup>15] studied the accessibility of mobile e-government apps instead, finding that they exhibit many accessibility issues (including the most basic and trivial to fix ones).

Paul [Pau22] performed a similar evaluation of Indian e-government websites using a sample of sixty-five websites of various ministries, finding that the majority of the analyzed websites do not meet even WCAG 2.1 level

A conformance. However, many studies about the accessibility of E-GOV websites and applications have been conducted in Asia. Both Latif and Masrek [LM10] and Isa et al. [ISS11] evaluated the accessibility of -egovernment websites in Malaysia; Al-Soud [ASN10] and Abu-Doush et al. [ADBMAAB13] did the same for websites in Jordan. Similar studies have been conducted by Al Mourad and Kamoun [AMK13], Paul [Pau22] and Agrawal et al. [AKS22], Baowaly and Bhuiyan [BB12], Bakhsh and Mehmood [BM12], Darmaputra et al. [DWA17], and Al-Faries et al. [AFAKARAD13], Akram and Sulaiman [AS17], Tashtoush et al. [TAAS16], Al-Khalifa et al. [AKBA17], about websites in Dubai, India, Bangladesh, Pakistan, Indonesia and Saudi-Arabia, respectively. More particular is the case of China: Rau et al. [RZSZ16] compared the accessibility of E-GOV websites in China in 2009 and 2013; unfortunately, they demonstrated that accessibility worsened over this five years time period.

In the literature the accessibility of e-government websites in Africa has been discussed as well; Werkijika and De Wet [VDW17] studied the accessibility of e-government websites in Sub-Saharan Africa (SSA), examining which macro factors influenced the accessibility these websites. Their findings indicated that the majority of government websites in SSA still had a long way to go to become accessible. More interesting, though, is that their cross-country analysis showed that there are three macro factors influencing e-government accessibility in SSA: Human Development Index (HDI), Corruption Perception Index (CPI), and percentage of the active population (15-64 years). Countries with high HDI levels and low CPI levels tend to have websites with fewer accessibility errors, while those for countries with high percentage of the active population have more accessibility issues. Adepoju et al. [ASB16] studied the accessibility of e-government websites in Nigeria; Nakatumba-Nabende [NNKKT19] did the same for Uganda, and Bussarhan and Daoudi [BD14] for Morocco.

Finally, several international studies have also been conducted; this is the case for Kesswani and Kumar [KK22] who analyzed e-government websites

across countries that are members of the *G7* and *Brazil-Russia-India-China-South Africa (BRICS)* international organizations, Lest and Smith [LS14], and Cisneros et al. [CHMP21] who did a systematic literature review on this specific topic.

### 3.1.2 Education websites accessibility

Many studies have been conducted to assess the conformance of websites from university and other institutions in the education field to accessibility standards and guidelines. As for E-GOV websites, the situation is not too encouraging. In fact, results show that these websites do not conform to accessibility guidelines. Such analysis have been conducted in different countries all over the world. For the sake of information accuracy, studies older than 2008 or referring only to WCAG 1.0 won't be considered.

With regard to universities, Barricelli et al. [BCDR21] demonstrated that, despite Italian laws requiring such websites to be accessible since 15 years (at the time the paper was written), universities struggle to satisfy even the minimum accessibility requirements. Máñez-Carvajal [MCCMFP21] analyzed the websites of top-ranking universities in Spain, Chile and Mexico; none of them fully satisfied the applicable accessibility guidelines. The same emerges from the analysis by Alamahdi and Steve [AD17], who evaluated the accessibility of top-ranking university websites in the world, Oceania and Arabia considering their initial, admission and course description web pages. Many more studies demonstrate that universities struggle to satisfy accessibility guidelines, standards and regulations in their websites; these include the ones by Akram et al. [AAS22] on Arabic universities, Kamal et al. [KWAAK16] about the ones in Jordan, Sodhar et al. [SBA19] on engineering university websites of Pakistan, Nacheva [Nac21] on Russian and Bulgarian university websites, and Arasid et al. [AAW<sup>+</sup>18] on thirteen universities' websites in West Java, Indonesia. Similar findings emerge from the studies conducted by Ismail and Kuppusamy [IK19] on popular U.S. college websites. Debevc et al. [DKH<sup>+</sup>15] demonstrated that applying a very simple heuristic method to test a limited

sample of pages from an educational website can be enough to demonstrate the presence of critical accessibility issues.

Recognizing the importance that multimedia resources are assuming in the education sector, Acosta et al. [AAVZMLM20] evaluated the accessibility of videos published by top ranking universities in the world, while Acosta-Vargas et al. [AVSUPMR19] did the same for selected universities in America. In both cases, videos were not fully accessible due to the lack of captions or audio descriptions. Campoverde-Molina et al. [CMLMG19] analyzed the electronic documents published by Ecuadorian educational portals; the situation with such documents may be even worse than that of web pages.

With regard to special education cooperatives websites, a study by Baule [Bau20] shows that only 25% of them satisfy minimum WCAG requirements, even if compliance is enforced by laws and regulations.

Some studies have analyzed the accessibility of school websites as well; these include the ones by Gonçalves et al. [GMP<sup>+</sup>13], May and Zhu [MZ10] about public schools of Texas (U.S.),

Open education is an umbrella term under which different understandings of open education can be accommodated [EU 22]. It goes beyond OER and open research outputs to embrace strategic decisions, teaching methods, collaboration between individuals and institutions, recognition of non-formal learning and different ways of making content available. The accessibility in the open education area has been discussed in the literature as well; such works include the websites analysis by Navarrete and Luján-Mora [NLM15], the accessibility evaluation of OER repositories by Perifanou and Economides [PE22], and the systematic literature review by Zhang et al. [ZTN<sup>+</sup>20]. The presence of many accessibility issues in this area clearly contrasts with the definition of open education provided by the EU Commission [SPCM16].

Other works in the literature that examined specific scenarios include the research by Shestakevych et al. [SPN<sup>+</sup>18], who evaluated the accessibility of web products for Ukrainian inclusive school graduates, Panda and Chakravarty [PC20], who tested the accessibility of IIT library websites in In-

dia, Spina [Spi19], who examined the accessibility of online content shared by various libraries and the benefits of new success criteria introduced in WCAG 2.1, and Calle-Jimenez et al. [CJSGLM14], who evaluated the accessibility of massive open online courses on Geographical Information Systems.

Finally, the accessibility of higher education institution websites has been evaluated in different countries, obtaining similar results; such analysis include the ones by İşeri et al. [İÜİ17] for the ones of the Cyprus island, Roig-Vila et al. [RVFFM14] for Spain, AlMeraj et al. [ABAQ21] for the state of Kuwait, Aziz et al. [AIN10] for Malaysia, Carvajal et al. [CPM18] for Chile, and the cross-country analysis by Kuppusami and Balaji [KB21], Kesswani and Kumar [KK16], and Shawar [Sha15]. Even Finland, which is one of the leading European countries in terms of high technology and digitalization, higher educational institutions' landing pages are not accessible [LLPK22].

No matter what method is used to analyze the accessibility of websites in the educational area (either relying on automatic tools, real users or accessibility experts), as stated by Campoverde-Molina et al. [CMLMG20] the literature confirms that “all the educational websites analyzed in the papers need to correct errors”. Furthermore, they state that “Educational websites do not meet any version of WCAG and their conformance levels”. Badbard et al. [BPC10] and Acosta-Vargas et al. [AVLMSU17] demonstrated that, while most universities have a Web accessibility policy, most policies have serious deficiencies; the deficiencies are of sufficient magnitude that many institutions are likely in violation of the ADA and at risk for lawsuits from people with disabilities, unless these policies get strengthened.

### 3.1.3 Service providers websites and mobile apps accessibility

Analyzing the accessibility of service provider websites only confirms what has been already illustrated in this section: all these websites exhibit one or more accessibility issues, and in most cases such issues are so critical as to prevent people with disabilities from interacting with them or taking



advantage of the services they offer.

Sohaib and Kang [SK17] analyzed the accessibility of various -e-commerce websites for people with disabilities; they confirmed that they exhibit different issues, and in many case such issues can prevent people from completing their purchases. Similarly, Isa et al. [ISA<sup>+</sup>16] examined the accessibility of home-stay websites in Malaysia, revealing that most issues can be related to a few critical errors including images' alternative text, information and relationships, color contrast, links purpose, and form field labels.

Accessibility issues characterize health-related websites as well. Most European healthcare websites feature significant accessibility issues [SLOM19], but including websites from countries outside the EU only confirms this trend [MCB21, AVHAV<sup>+</sup>20]. Birkun and Kosova [BK22] assessed the accessibility of online English-language courses in resuscitation, demonstrating that web pages of massive open online courses (MOOC) (online courses that are available to the general public for free, potentially causing massive participation) in Basic Life Support (BLS) had severe WCAG 2.1 violations concerning 9 of 13 (or 69.2%) guidelines. As demonstrated by Alismail and Chipidza [AC21], Niom and Lin [NL22], Ara and Sik-Lanyi [ASL22], Ewitt and He [HH21], and Yu [Yu21], even critical emergency services such as vaccine registration systems and information portals used in the ongoing COVID-19 pandemic exhibit critical accessibility issues

Financial services and banking websites present critical accessibility issues as well, in some cases even preventing people with disabilities from using them at all; this is clearly demonstrated by various studies, including the ones by Kaur and Dani [KD14], Martínez et al. [MDAG14], Oyefolahan et al. [OSAB19], Akgül [Akg18], Fatima et al. [FBB20], Nazar et al. [NSS17], Wentz et al. [WPF<sup>+</sup>19], Tadele et al. [TRW18], and others. Lorca et al. [LDAM16] argue that web accessibility implementation is an important and affordable way that could be used to increase the possibilities to explore online information and meet the Corporate Social Responsibility demands of stakeholders, showing that smaller banks are more prone to implementing

web accessibility; while this may seem counterintuitive, it could be explained by considering that this might help them to differentiate from their competitors and create strategic advantages.

While an in-depth discussion of mobile apps accessibility would require a separate PhD thesis on its own, for the sake of completeness it is worth mentioning that the situation is even less encouraging. Studies by Chen et al. [CCF<sup>+</sup>22], Alshayban et al. [AAM20], Yan and Ramachandran [YR19], Vendome et al. [VSLLV19] and others demonstrated that mobile apps almost always contain critical accessibility issues that can even prevent people with disabilities from using such apps at all. The same applies when considering the accessibility of Videos available on the most accessed Websites (including social media and video sharing platforms): as demonstrated by Rosas-Villena et al. [RVRGF15], just considering the issues faced by people with visual impairments, accessibility of this kind of content is not in good shape either.

## 3.2 Reasons behind the inaccessible Web

In light of the various laws and regulations requiring the creation of accessible websites described in Section 2.2, the standards and guidelines shown in Section 2.1, and the abundance of resources introduced in Section 2.3, it is difficult to explain why so much content available on the Internet is not accessible as outlined in Section 3.1. Other than a general and widespread lack of knowledge and interest among stakeholders, in fact, it is hard to find the reasons behind the existence of so much inaccessible content on the Internet precisely. In some cases, such as the one cited by Roig and Ribera [RR15] for the creation of accessible long-form documents, the process may be simply too complex for non-technical users to handle. However, it may be worth discussing certain circumstances to understand the factors leading to the current situation with web accessibility. Among others, these include the effectiveness of standards, guidelines, laws and regulations, how stakeholders (especially web developers) perceive them, and the appropriateness

of accessibility testing and evaluation tools.

When it comes to accessibility testing, there is no automated tool that can compete with a human in terms of quality and depth of the analysis. Unfortunately, manual accessibility testing is a time- and money-consuming operation, and requires specific and non-trivial competence that may not be available among stakeholders, and stakeholders may not be willing to acquire [RV20]. Over time, numerous automated tools have been suggested to help web developers and content creators find accessibility issues and choose the best fixes for them; unfortunately, evidence shows that only up to 50% of accessibility issues can be discovered by automated testing tools [VBC13]. The rules used by automated accessibility testing tools to detect errors are often not well documented, and even though a success criterion is said to be checked, this does not guarantee that the tool is able to detect all possible violations of that criterion. Therefore, it is difficult to discern advantages and limitations of the different tools [TA17].

The fact that different coding specifications for the guidelines and varied searching and matching techniques used by different automatic accessibility testing programs can result in different evaluation findings for the same page further complicates the matter. Therefore, effective accessibility testing methodologies combine automatic accessibility testing (typically with different tools) and manual evaluations by one or more experts [AAV19]. Finally, the evaluation by actual users performing real tasks could also be required in addition to automated accessibility testing and evaluation by accessibility experts, as it may unveil different issues [BPG14].

There is also evidence that developers struggle to grasp accessibility standards and guidelines, which are perceived as excessively technical and lacking in problem-solving assistance [HZ14]. Evidence shows that the majority of website accessibility advancements over time have been brought about by the adoption of more accessible technologies rather than an increase in interest in digital accessibility [HR13]. Therefore, it may be necessary to *rethink* the current approach to web accessibility testing and evaluation [LRV22].

According to Leite et al. [LSFE21], the lack of clear guidelines and well documented testing tools is a factor that discourages the adoption of web accessibility in mobile app development.

The general awareness of Web Accessibility among stakeholders of a project is a necessary precondition for creating an accessible website or application. Inal et al. [IRY19] show that User Experience (UX) professionals in Turkey believe they have received sufficient training and instruction in web accessibility, yet they are unaware of the assistive technologies used by persons with disabilities and the web accessibility standards, and they do not include them in their usability tests or projects. The literature shows that the majority of web practitioners involved in the development process, especially in technical roles, have acquired at least a very basic awareness of Web Accessibility. In contrast, non-technical and management roles tend to be less aware [YBVH15, FRF08]. This causes Web Accessibility issues to be often deprioritized compared to other project requirements [VNvdG17]. In addition to this, insufficient knowledge of web accessibility brings up a row of potentially harmful conceptions about it. As illustrated by Ellcessor [Ell14] these misconceptions (just to mention a few) include the ideas that web accessibility:

- compromises aesthetics;
- prevents from adopting technologically advanced solutions;
- either requires high (unaffordable) expenses or no cost at all;
- is only a concern for people with visual impairments;
- is a topic that is solely the developer's responsibility.

In contrast, other works in the literature demonstrate these *ideas* to be *myths* rather than truth, as they cannot be confirmed at all in reality. Moreover, findings from Mbipom and Harper [MH11] and Petrie et al. [PHK04] show that Web Accessibility has no negative impact on aesthetics, while the

ones from Schmutz et al. [SSS16, SSS18, SSS17], Aizpurua et al. [AHV16] and Vollenwyder et al. [VPI<sup>+</sup>23] demonstrate that it improves performance and perceived usability for all users, not only people with disabilities.

Another challenging aspect that should be considered is how to let web practitioners understand how the adoption of a specific design and implementation (e.g. for a widget in a web page) can address the needs of users with disabilities and the reasons why it should be preferred to other (inaccessible) ones. Broad classifications such as people with visual, hearing, motor, or cognitive and learning impairments offer web practitioners only a rough idea about potential accessibility barriers people with disabilities may face and very limited support in determining how to overcome them [VIB<sup>+</sup>19]. Understanding potential accessibility barriers becomes even more complex due to the way individuals use assistive technologies like screen readers, personalized stylesheets, screen magnifiers, voice-control and eye-tracking systems [Edw08, HRS08] and the skill levels with which they use them [VH14]: there can be issues that look like accessibility barriers, but depend on the fact that the user is not proficient with assistive technologies rather than the website or application itself.

But this is only a part of the story. In fact, the target users who may benefit from accessible websites and applications include people with temporary (impairments that are not permanent, such as a broken arm) or situational (impairments that depend on the context/situation you are in, such as being in a very crowded and noisy environment) disabilities, which increase the number of affected users to a much greater extent [Far11].

Furthermore, given that the majority of web developers and content authors are not people with disabilities, it can be argued that there is no *direct* experience of accessibility issues in most web projects. Non-disabled people cannot perceive the content of their work in the way disabled people would, thus cannot perceive personally and precisely the issues they have allowed to remain in their code. The usual edit-reload-watch cycle of most development efforts does not work for accessibility, because developers cannot directly

*watch* the effect of the latest edit cycle. Instead, they have to rely on indirect witnesses, be they people with disabilities enrolled as testers, validation tools, or third-party experts. Additionally, the more indirect this witness is, the more difficult it is to fix the issues that are found: either accessibility validation blocks all other development activities, therefore getting very expensive (in the context of usually late projects) or it is performed in parallel with other implementation activities that keep on modifying the code base being reviewed, making the review itself either pointless, outdated or unaware of additional accessibility issues being introduced in the meantime [RV21].

While many aspects related to web accessibility have been discussed in the literature, not much is available about the way in which accessibility testing and evaluation results are presented to web practitioners by automatic testing tools. The little information gathered, however, shows that the current approach (that basically is shared by all tools except for minor differences) may be problematic [TDR11].

Finally, various aspects directly arising from accessibility standards and guidelines (especially WCAG) have been questioned. First off, it is recognized that the heterogeneity of technologies used on the web is a major strength in terms of flexibility and adaptability, but also poses major challenges when it comes to actually develop accessible websites and applications [HC12]. Indeed, several objections to the fact that WCAG covers all possible scenarios (independently of the implementation technologies used) and guarantees the creation of accessible products that satisfy all requirements of people with disabilities have been raised.

Alajarmeh [Ala22] discussed the extent of accessibility coverage of mobile applications provided by WCAG 2.1 about the requirements of visually impaired people, finding that several critical aspects are not covered at all; Da Silva et al. [DSVE22] suggested a strategy to widen such coverage by expanding the automated tests written to verify that the UI of an Android app works as expected. Rello [Rel15] demonstrated the same lack of coverage for

the requirements of people with dyslexia to have truly accessible web content. After a thorough statistical analysis on WCAG to demonstrate the validity of success criteria aimed at satisfying the requirements of people with *cognitive impairments*<sup>1</sup>, Eraslan et al. [EYYH19] argue that “comparative studies between various types of cognitive disabilities are very much needed in order to gain a better understanding of the needs of various user groups with different cognitive impairments and assess the commonalities and differences in their requirements”.

Since version 2.0, one of the major goals of WCAG was testability: success criteria should be either *machine testable*, meaning that content conformance can be verified by an automatic testing tool, or *reliably human testable*. However, evidence shows that this is not always the case. Brajnik et al. [BYH12] conducted an experiment involving twenty-five accessibility experts to verify how different accessibility audits could *agree* in their findings. They found that:

- an 80% agreement between experienced evaluators almost never occurred, the average agreement was at the 70–75% mark, while the error rate was around 29%;
- trained, but novice, evaluators performing the same audits exhibited the same agreement to that of more experienced ones, but with a reduction on validity of 6–13%;
- expertise appears to improve (by 19%) the ability to avoid false positives;
- combining the results of two independent experienced evaluators would be the best option, capturing at most 76% of the true problems and

---

<sup>1</sup>While different clinical definitions of cognitive impairments are available, generally speaking they refer to a broad group of disabilities ranging from intellectual disabilities to age-related issues with thinking and remembering. They include mental illnesses, such as depression and schizophrenia, and learning difficulties, such as dyslexia and attention deficit hyperactivity disorder (ADHD).

producing only 24% of false positives.

In their experiment, Alonso et al. [AFGM10] made several students (thus not accessibility experts) manually evaluate the accessibility of a web page: only eight of twenty-five WCAG 2.0 success criteria could be considered to be reliably human testable when evaluators were beginners. Crabb et al. [CHJ<sup>+</sup>19] showed that there are gaps in the knowledge needed to develop accessible products despite the effort to promote accessible design, yet designers are aware of where these gaps are and can suggest a number of areas where tools, techniques and guidance would improve their practice.

Even more worrying is some evidence that WCAG conformance may not be enough to ensure that a website or application is not accessible. Rømen et al. [RS12] showed that only 49% of actual accessibility barriers experienced by people with disabilities might have been determined by verifying the conformance of a website to WCAG 2.0, suggesting that accessibility guidelines should also include usability. With a more specific analysis focused on barriers experienced by blind people, Power et al. [PFPS12] obtained the same result. Chandrashekar and McCardle [CM20] investigated the relationship among different guidelines and the usage of switch-control devices: they found most guidelines to be applicable, even those that were not specifically thought for removing barriers experienced by people with physical disabilities; they highlight how in some cases complying with WCAG 2.1 causes the border between accessibility and usability to be crossed. Power et al. [PPFS11] suggests that also WCAG implementation techniques should be evaluated to assess their effectiveness in removing barriers actually experienced by people with disabilities. Sala et al. [SAPV19] even demonstrated that E-GOV websites can exhibit accessibility issues even if they declare their conformance to WCAG and no WCAG violations are detected



### 3.3 Physical objects and indoor environments' accessibility

The usage of digital technologies to improve the accessibility of physical objects and indoor environments for blind and visually impaired people, and, more generally, for people with disabilities, is not new. In the literature, there are several examples of initiatives, applications, techniques, experiences, and so on that use digital technologies to overcome the limits that having a disability entails. Making an exhaustive literature review for this area is certainly a daunting task, and due to the number of proposed works it would likely be incomplete. However, in this section some projects related to the usage of digital technologies to improve the accessibility of physical objects and indoor environments will be examined. Instead of providing an exhaustive discussion, the focus of this section will be to document *work typologies* that are close to the ones for which prototypes have been created as part of this research project.

#### 3.3.1 Making indoor environments more accessible

Blind and visually impaired people have very specific requirements when it comes to moving independently in indoor environments, especially if they are not familiar with them. Other than employing walking aids (such as the white cane or guide dog), they face different challenges than the ones their sighted peers face, thus require specific indications and need to adopt particular behaviors to overcome them [JWW19, KOBKA18]. This is clearly confirmed by examining the methodologies with which *Orientation and Mobility (O&M) experts* teach blind and visually impaired people to walk independently [BWW10]. For instance, they teach them to rely on so-called *landmarks* to orient themselves in both indoor and outdoor environments; such landmarks act as *references* to understand whether they are moving on the correct path or not, and to understand their surroundings. Landmarks can be either things that are traditionally used as reference points

by sighted people as well (such as corridors, doors, shapes) or particularities not considered at all by their sighted peers (such as very slight changes in walking surfaces, smells, specific sounds, constant noises, direction changes apparently imperceptible to the average user). These considerations lead to the fact that creating an accessible system to improve the accessibility of indoor environments for blind and visually impaired people is a very complex task, as many specific requirements must be kept into account [SE07, HCM<sup>+</sup>15, FFP13, MSW11, EMC<sup>+</sup>20].

A thoroughly explored approach to make indoor environments more accessible consists of the creation of *indoor navigation systems*, applications whose implementation is very complex due to two main factors:

***Positioning system and algorithms.*** As the *Global Positioning System (GPS)* cannot be used to determine the user’s position indoors, the most disparate technologies have been employed to obtain the same results. Therefore, navigation systems that leverage positioning algorithms based on Wi-Fi signals (such as [HJLY14, CHCL16]), computer vision and Augmented Reality (AR) (such as [TLZX12, YLR<sup>+</sup>19]), Near Field Communication (NFC) ([OOCA11, Iva10, GST<sup>+</sup>14]) and Radio Frequency Identification (RFID) ([CTK08, FLF<sup>+</sup>10]) tags, *beacons*<sup>2</sup> (such as NavCog [AGR<sup>+</sup>16], AlmaWhere 20 [DMR<sup>+</sup>20], GuideBeacon [CNW17] and many more [KSS22]).

***Mapping.*** Generally speaking, creating an accessible indoor navigation system requires information unavailable in traditional maps representing indoor environments [PAK<sup>+</sup>17]: it is very hard to obtain the *semantics* of all elements that are present in such representations, which is critical to make a navigation system accessible [GGL<sup>+</sup>16].

Making it simpler for blind and visually impaired people to *explore* these spaces by leveraging digital technologies is another approach that has been

---

<sup>2</sup>Beacons are small Bluetooth Low Energy (BLE) devices that broadcast a customizable data packet at configurable time intervals with a settable transmission power over Bluetooth.

extensively studied in the literature to increase indoor accessibility. Unfortunately, the creation of accessible maps is a challenging process under different aspects. On one hand, as argued by Striegl et al. [SLSSW20], there are “no widely accepted open standards for the expression of accessibility information in indoor maps” and “a lack of methods to assess if indoor maps comply with the requirements of people with disabilities in terms of orientation and indoor navigation”. These findings are confirmed in the analysis by Froehlich et al. [FBC<sup>+</sup>19], which highlights additional challenges as well (such as the ones involved in the capturing of accessibility-related information required by accessible maps). On the other, the process of determining the most effective representation for accessible maps is still undergoing: Holloway et al. [HMB18], for instance, compared accessible maps created by using *tactile graphics* and *3D printing*, demonstrating the latter to be more effective. Melfi et al. [MMJS22] proposed a common symbols set to represent information in tactile maps independently of the printing method used to create them.

However, Buzzi et al. [BBLM11] discussed several strategies to improve the accessibility of visual maps for blind and visually impaired people involving novel interaction methods. Calle-Jimenez and Luján-Mora [CJLM16] proposed a prototype for an indoor map whose information is displayed in a SVG image enriched with accessibility-related information; a web application is then used to display the map in an accessible way, letting blind users explore it and compute the *routes* to reach specific points on the map. Melfi et al. [MBMS22] developed an audio-tactile system aimed at making indoor maps more accessible “to help blind people preparing a trip to unknown buildings”. Senette et al. [SBB<sup>+</sup>13] proposed a system to enrich graphic maps with *multimodal interactions* to make them accessible for blind people. Schmitz and Ertl [SE12] studied how to display a map interactively on a tactile graphics display to maximize its accessibility. Other hybrid, interactive solutions include the ones by Wang et al. [WLHH09] and Ducasse et al. [DBJ18].

Recognizing the complexity of creating accessible maps, some works in

the literature have been focusing on simplifying this process. Tang et al. [TTHZ16] proposed an automatic method to generate accessible maps for indoor environments based on *floor plans* created using the *AutoCAD* modelling software. Engel and Weber [EW22] proposed *Atim*, a system for the automated generation of interactive, audio-tactile indoor maps by means of a digital pen.

### 3.3.2 Making physical objects more accessible

When it comes to leveraging digital technologies to make physical objects more accessible for blind and visually impaired people, scenarios from everyday-living immediately come to mind: for instance, just considering the latest advances in assistive technologies and the digitization process of paper documents, it is clear that a lot of opportunities previously unavailable have raised, say, from thirty years ago. Even considering the current situation with digital accessibility (which as described in Section 3.1 is far from being optimal), blind and visually impaired people gained a lot of opportunities for accessing educational information, jobs and professional activities, culture, and independent-living in general. Digital technologies have enabled blind and visually impaired people to do *things* that were unavailable (or, in some sense, inaccessible) before.

In the aforementioned general progress, a significant trend to improve the accessibility of physical objects by means of digital technologies has been *object recognition*, that allows blind and visually impaired people to *identify* objects they cannot touch and get descriptions of their surroundings (e.g. recognizing people around them without having to wait for them to speak, avoid potential dangers, explore unknown environments). This is made possible by works like the ones by Jabnoun et al. [JBA14], Parlouar et al. [PDMJ09], the ones in the literature review by Budrionis et al. [BPDI22], and many countless more.

However, digital technologies can also be used to make accessible to blind and visually impaired people objects that, by their nature or characteristics,

could not be accessible as they cannot be touched; thinking to an area with these characteristics *cultural heritage* immediately comes to mind. Unsurprisingly, many works to improve its accessibility by exploiting digital technologies (especially AR and 3D-printing) have been proposed in the literature; these include *tooteko* by D'Agnano et al. [DBGV15], the prototype by Grillo et al. [GMRV22], and the studies by Neumüller et al. [NRRK14], Rossetti et al. [RFL<sup>+</sup>18], Ballarin et al. [BBV18], Anagnostakis et al. [AAK<sup>+</sup>16], and Vaz et al. [VFC20], and Montusiewicz et al. [MBK22].

Finally, there is an area in which digital technologies can make both physical object and indoor environments more accessible at the same time: the so-called *virtual tours*. Usually consisting of simulations of existing locations, composed of a sequence of videos, still images or 360-degree images and other multimedia elements such as sound effects, music, narration, text and floor maps, virtual tours are gaining momentum due to their huge potential (e.g. [LXX<sup>+</sup>22, SAB<sup>+</sup>20]). Works on accessible virtual tours for blind and visually impaired people have been focusing on specific, very well targeted areas: facilitating blind and visually impaired O&M training [FDVS20], giving them access to cultural heritage ([MS18, DE10]), or exploring indoor spaces (e.g. [GCO18]). Still, many challenges to ensure the accessibility of virtual tours have to be dealt with [GSEB<sup>+</sup>12].

In any case, for each of the described scenarios, even if different works have been proposed in the literature there is still no *generic system* ready to be deployed and used to make indoor environments and virtual tours accessible for blind and visually impaired people, offering compelling features to their sighted peers at the same time.



# Chapter 4

## The research project

Aim of this research project is to propose an approach to *rethink from the ground up* the way in which web accessibility is handled to make it *more accessible*. Despite the standards, laws, regulations, and other tools discussed in Chapter 2 to assist developers, designers, content authors, policymakers, and all other involved stakeholders in producing accessible content, the situation with web accessibility is far from ideal, as described in Chapter 3. While countless attempts have been done to improve the *status quo*, all of them lied on the foundation of existing philosophies.

### 4.1 The research project and its purposes

The main purpose of the research project documented in this PhD thesis is to create and implement the *technologies and methodologies* that have been deemed necessary to improve the current situation with web accessibility. The “ratio” behind the research project documented in this PhD thesis can be summarized with the statement *making web accessibility more accessible*. Indeed, the project can be divided into two parallel branches that have been investigated concurrently:

- improving the current web accessibility situation. This includes providing a *strategy to rethink web accessibility from the ground up*, developing

and evaluating all necessary tools and methodologies and complementary resources to realize this vision;

- leverage digital technologies to improve the accessibility of indoor spaces and physical objects, keeping into account the fact that general purpose, abstract and readily deployable solutions must be obtained. Each tool reflects a precise *methodology* to approach the problem it handles.

Concerning the first branch, the research focuses on finding innovative solutions for three key areas.:

- the development of accessible websites and applications by people who are not accessibility experts;
- making manual accessibility testing and evaluation easier for people who are not accessibility experts;
- minimizing the time and effort required to find any accessibility-related information that may be relevant to any stakeholder during any process in which accessibility may be engaged.

In terms of the development of accessible websites and applications, this research project assumes that current web technologies and user agents can do far more than they do to *enforce* the creation of accessible content. Part of this research project involved demonstrating the truth of this statement, explaining how this purpose can be achieved. Naturally, various decisions have been made to ensure that the production of a prototype for the proposed solution is not only conceivable, but also simple to install and immediately available for use by potential stakeholders. This has resulted in the creation of **AX**: a declarative framework of accessible web components.

Moreover, this project proposes a whole new approach to manual accessibility testing, with the aim of innovating the way accessibility testing is performed. Although automated accessibility testing has advanced significantly over the past several years, manual accessibility testing will not be phased out anytime soon. While several tools for simplifying accessibility testing and



evaluation have been presented over time, none of them have focused on entirely redefining how their users perceive accessibility issues and their impact on individuals with disabilities. “A picture is worth one thousand words”, they say; it is a fundamental idea in this project that this statement also applies when it comes to *direct* experience of accessibility issues compared to *indirect* reports. This has resulted in the creation of a novel *methodology* to let users perform accessibility testing and evaluation and *directly experience* the results; this methodology has been actually implemented in a browser extension called Sighted Architects Helper for Accessibility Notation (SAHARIAN).

The third key area focuses on making easier for any stakeholder to find any accessibility-related resource. It should be emphasized that there are currently a great number of both informative and normative resources, as well as tools and other materials of any kind with practical examples, to assist stakeholders in creating accessible web content. However, they are *scattered* across the Internet, with all the problems this brings to the table: for instance, these include the difficulty of finding them, a general lack of completeness of any resource considered out of its context, and the hard problem of enabling non-accessibility experts to *distinguish* authoritative information sources from non-authoritative ones. Therefore, as part of this research project an attempt to improve this aspect has been made as well. This has led to the creation of A11A, a structured, categorized collection of accessibility-related resources available on the Internet organized in a form that may resemble a wiki.

In developing all three key areas of this branch, a very strong focus has been put on *reducing the cognitive efforts* required in order to develop accessible websites and applications, perform accessibility testing and evaluation, and find accessibility-related information. Indeed, a critical aspect to *make web accessibility more accessible* involves making it more “affordable” for people who are not expert in this field. In this context, the proposed solutions intend to dramatically reduce the *entrance barriers* in all three areas by *re-*

*thinking* entirely the way in which they have been dealt with in recent years.

With regard to the second branch of the research project discussed in this PhD thesis, a couple of attempts at making indoor environments and physical objects more accessible thanks to digital technologies have been made as well. While many proposals from both industry and academy have been discussed in the literature, to the knowledge of this writer none of them satisfies the three following conditions at the same time:

- is readily available for large scale deployment;
- is implemented fully embracing the *universal design* philosophy;
- is implemented so that accessibility features lie at the foundation of the project and are developed alongside the ones required by abled users.

More specifically, two key problems have been dealt with:

- Creating a prototype for a generic system that, given the representation of an indoor environment in a specified format, can let users actually *explore* it by means of a virtual tour keeping into account their specific requirements. Such a system is conceived for easy deployment and conjugate the accessibility requirements of blind and visually impaired people with the features commonly expected from such systems by their sighted peers.
- Creating a way to *improve* the accessibility of physical objects by means of digital technologies. More specifically, a method to make *interactive, scientific, decomposable objects with a very high historical value* (thus *cultural heritage*) accessible again by means of digital technologies that complement tangible artifacts. In particular, this methodology has been conceived based on the characteristics of the *Brendel botanical models* and successfully applied to them by creating a working prototype.

While in this PhD thesis all parts of the research project will be discussed as an integrated approach, it should be noted that they have been thought

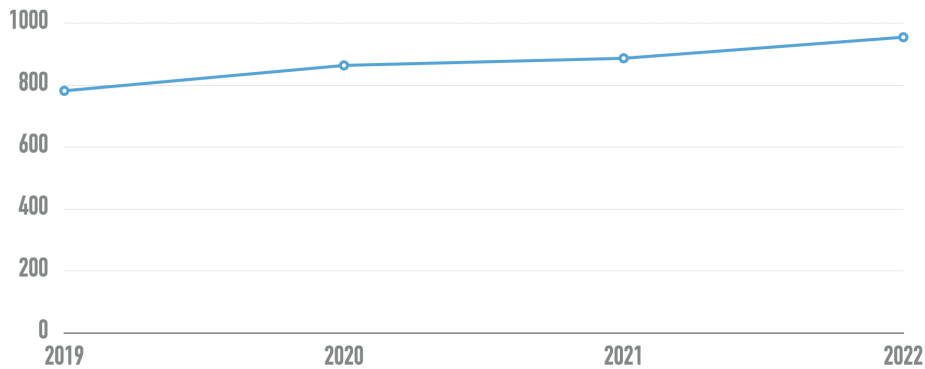
to be used independently one from the others as well. Undoubtedly, their impact is maximized when used in combination, and indeed the synergy in the various components (A11A, the AX framework, and SAHARIAN) and their similarities are evident. However, as each of them can satisfy different requirements that could arise in different phases in the development process, they have been carefully designed to work properly as *standalone* solutions as well. Of course, all common aspects arise from the fact that all tools originate from a single approach, that is *making web accessibility more accessible* by reducing as much as possible the cognitive efforts required by web developers to handle it correctly.

All the proposed solutions part of the research project documented in this PhD thesis, as well as their working logic, ideas and design principles behind their implementation, will be discussed in the remaining of this chapter.

## 4.2 Is web accessibility accessible?

The research project documented in this PhD thesis lies its foundation on the premise that currently web accessibility is not accessible. This may sound paradoxical, but it is the reason why it has been deemed crucial to adopt a novel strategy to rethink from the ground up the approach in which web accessibility is typically handled. This project aims to *make web accessibility more accessible* through a novel approach to *rethink web accessibility from the ground up* by means of innovative accessibility testing and evaluation tools, a declarative framework to support web developers, and a collection of the accessibility-resources already available on the Internet.

As discussed in Chapter 2 countless attempts have been made to make the web more accessible. Yet, the results show that such efforts require a very long time to produce the desired results, and in the meantime people with disabilities are prevented from accessing part of the Web (with all the consequences that this implies). Figure 4.1 shows a line chart indicating the number of HTML elements contained in the top 1,000,000 home pages



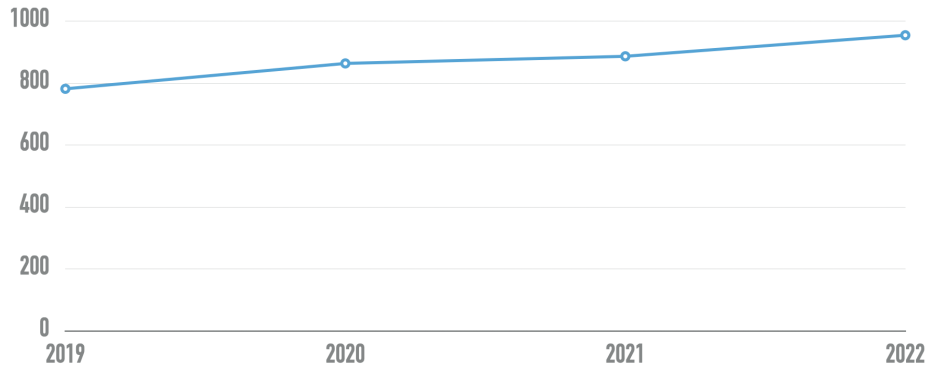
A line chart showing how the number of HTML elements in the top 1,000,000 home pages analyzed by webAIM is increasing over the years.

Source: [Web22a]

Figure 4.1: Number of elements in the top 1,000,000 home pages analyzed by WebAIM

analyzed by WebAIM in their annual accessibility report in the last five years: it clearly indicates that web pages complexity is increasing very fast. In contrast, Fig. 4.2 shows a line chart indicating the number of web pages that contain critical accessibility issues in the last five years from the same dataset: the trend indicates that this number had just a very, very slight decrease over time. Thus, it can be assumed that, while stakeholders are willing to invest in more complexity to get more sophisticated web pages, they are not doing the same about web accessibility. While many ethical and socio-economic factors might influence this decision, in this PhD thesis only the technical aspects will be considered. Indeed, these figures may confirm the evidence that accessibility improves when technology becomes more accessible, and not due to an increased interest among stakeholders.

The idea that currently web accessibility is not accessible arises from the consideration that, as shown in the literature (see Section 3.2), the majority of web developers, designers, content authors and stakeholders in general are not *accessibility experts*. In this scenario, this judgement is based on three main aspects:



A line chart showing how the percentage of the top 1,000,000 home pages analyzed by WebAIM affected by accessibility issues is decreasing over the years just slightly.

Source: [Web22a]

Figure 4.2: Percentage of pages in the top 1,000,000 home pages analyzed by WebAIM affected by accessibility issues over time

- the *usability* of accessibility testing and evaluation tools;
- the fact that web accessibility is not *enforced* at the implementation level (e.g. by web technologies and user agents), thus is a completely *opt-in* (facultative) process;
- the difficulty of finding authoritative accessibility resources for non-accessibility experts.

### 4.2.1 The usability of accessibility tools

With regard to the usability of accessibility testing and evaluation tools, many attempts to improve them have been done over the years. However, all these attempts assume that any web developer should gain some *specific knowledge* about web accessibility. Unfortunately, these specific competencies are not always available in development teams, acquiring them is not trivial, and developers may or may not be willing to acquire them. Further-

more, due to budget requirements or time constraints, sometimes improving the accessibility-related skills of a developer may be unaffordable. While in a perfect world all developers should have enough knowledge to develop accessible websites and applications from the beginning, and time constraints and budget restrictions should not justify the lack of accessibility, it should be noted that in reality things don't always go the way they should (or they are expected to go). In the context of usually late projects, maybe created by very small teams who need to reduce the *time to market* to its minimum (as per market demand), this may not be feasible.

Additionally, the majority of existing accessibility tools center around the idea of analyzing the code of a web page and generate a technical report that points out accessibility issues by indicating somehow the point of the affected page element. In general, this indication may be given pointing to the line of code where the element is defined or, in some cases, by *injecting* specific visual indicators close to the element. The problem, though, is that this approach requires additional and significant cognitive efforts by the tester to:

- identify where the issue occurs;
- carefully read the feedback provided by the tool, eventually integrating it with additional support resources, to understand what the problem actually is;
- understand how to fix the problem in that specific scenario (consulting additional resources, if necessary) and actually apply the fix.

Therefore, it is argued that this may not be the most effective way to communicate accessibility issues to web developers (who in most scenarios are not accessibility experts at all).

Furthermore, due to the way accessibility testing and evaluation tools are conceived and the ways in which they commonly present accessibility issues to their users, it can be said that developers can only have an *indirect* perception of accessibility issues: if they are not people with disabilities, they

have to rely the provided documentation to understand problems and their impact on people with disabilities. But given the complexity of accessibility testing, often they cannot even perform it in first person; instead, they have to rely on third-party analysis (with all the problems such approach might introduce, such as the ones about process timings and results reliability), who get involved when fixing accessibility issues becomes harder: as proven in the literature, the later accessibility gets involved in the development process, the more complicated, costly and less effective solving accessibility issues becomes.

Finally, it is worth recalling that there is not a *standard* accessibility testing and evaluation tool; instead, a plethora of them have been created. They differ in terms of issue coverage, the way in which they provide the results, and how they *support* the developer in fixing them. Thus, to the complexity of creating an accessible website or application, the difficulty of picking the right tool to test and evaluate its accessibility needs to be kept into account. The problem, though, is that these aspects are difficult to evaluate even for accessibility experts. How can an experienced web developer who is new to web accessibility be expected to deal with such complexities? The literature has no answer for this question; the innovative manual accessibility testing tools proposed as part of this research project can mitigate the problem.

### 4.2.2 Accessibility is not *enforced*

One of the factors leading to the current situation with web accessibility described in Section 3.1 may be the fact that *accessibility is not enforced at the implementation level*. To better explain this statement, two main aspects involved in any development workflow should be examined.

First off, as previously mentioned, it can be said that accessibility is an *opt-in* and completely optional feature. Indeed, while some user agents integrate features for accessibility testing and evaluation, they do not *activate* them automatically, nor promote their usage extensively. Thus, a web developer can end up producing inaccessible websites and applications without

even noticing it! To produce accessible content, one needs to be aware of the problem, of the existence of testing and evaluation tools, to *manually* engage such tools in the creation process, and finally handle the reported problems (gaining the necessary knowledge to do that).

To further complicate the situation, another aspect gets into play. In fact, the fundamental language with which all websites and applications are developed, thus the foundation for them, is undoubtedly HTML. However, it is neither sufficiently prescriptive to prevent abuses of the semantic characterization of its elements, nor sufficiently descriptive to provide all elements necessary to support features that are common and expected in many web applications.

Unfortunately, HTML allows web developers to create different equivalent implementations for each element. Such implementations may look the same (i.e. have the exact same visual appearance), have the same features, yet have different accessibility degrees. Listing 4.1 shows a very simple HTML page that is rendered properly by any web browsers, yet exhibit different accessibility issues:

- the language of the document is not declared;
- it does not contain any heading;
- it does not contain any *landmark* (or any semantic HTML element to represent the page structure);
- the form field is not associated to its label, which is implemented using a generic `span` tag rather than the appropriate `label` tag.

Listing 4.1: Example of a simple, inaccessible web page that renders fine

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
```



```
<meta name="viewport" content="width=device-width,
  initial-scale=1.0">
<title>Inaccessible form field example</title>
</head>
<body>
  <p>Inaccessible form example</p>
  <form>
    <span>Name:</span>
    <input type="text">
    <input type="submit" value="Send">
  </form>
</body>
</html>
```

But the situation is even worse. Considering the idea of fixing the accessibility issue of the text field contained in Listing 4.1; HTML offers at least five different possibilities, as shown in Listing 4.2:

1. replacing the `span` element with a `label` element, and making it wrap the `input` field;
2. replacing the `span` element with a `label` element, specifying the relationship between the input field and its label by means of the `id` and `for` attributes applied to the `label` and `input` element, respectively;
3. specifying an accessible name for the input field by means of the `title` attribute;
4. using the `aria-label` attribute from the WAI-ARIA specification;
5. using the `aria-labelledby` attribute from the WAI-ARIA specification.

Listing 4.2: Five options to implement an accessible text field in HTML

```
<!-- Option 1: label wrapping the field it is
      associated to !-->
<label>Name:
  <input type="text">
</label>
```

```
<!-- Option 2: using a label and the for attribute !-->
<label for="name-input">Name:</label>
<input type="text" id="name-input">
```

```
<!-- Option 3: using the title attribute !-->
<span>Name:</span>
<input type="text" title="Name:">
```

```
<!-- Option 4: using the aria-label attribute !-->
<span>Name:</span>
<input type="text" aria-label="Name:">
```

```
<!-- Option 5: using the aria-labelledby attribute !-->
<span id="name-input-label">Name:</span>
<input type="text" aria-labelledby="name-input-label">
```

While having all those representations makes it possible to create an accessible text field in different contexts, this introduces a cognitive effort for the developer to understand which one to choose, and the reason why one is preferable to the other in general and in a specific context. One may legitimately argue that this flexibility is required in order to support a multitude of features, such as allowing for better positioning and styling of both the field and its label, but this richness comes at a cost that in many cases is not acceptable. Better yet, are we sure that so much flexibility is really necessary? Couldn't astonishing designs be achieved differently?

With regard to HTML not being descriptive enough, HTML 5 can be con-

sidered a great step in the right direction. This version (and the following ones) added to the language many features commonly used in web applications that, not being available in a standardized implementation, had to be implemented leveraging external solutions (think of audio and video playback) with a varying degree of accessibility. However, many commonly used interactive widgets have not natively become part of HTML and still require markup and code that may or may not be accessible. Other language features have been included in later versions, but are not supported properly by browsers and/or assistive technologies: examples include the `dialog` element, that was designed to let developers implement modal and non-modal dialog windows, and the `datalist` element, that was intended to provide a mechanism to support native autocomplete widgets for entering data.

In addition to this, many controls commonly used by web applications (for instance, tabs and associated tab panels, menu bars, toolbars, trees, to name a few) have still to find a good and agreed upon native element to implement them in HTML to build good accessibility support upon. In addition to this, even in the best possible scenario, no guarantee is made about whether all these widgets will be implemented consistently across user agents (e.g. browsers and assistive technologies), or that styling them will be supported in the same way.

Due to these considerations, many controls required by complex web applications have to be implemented by leveraging generic HTML tags (e.g. non-semantic elements, such as `span` and `div`) enhanced with JavaScript code. As explained in Section 2.1.3 such elements can be made accessible thanks to the WAI-ARIA specification, which allows to enrich the semantics of HTML elements by adding markup (e.g., specific attributes, and in particular the role attribute) that define the semantics of the element in terms of accessibility. However, in this scenario the developer is responsible for manually implementing the exact behavior expected by assistive technology users for each element: marking an element as having a certain WAI-ARIA role is a promise, but the developer is responsible for fulfilling it: differently

than HTML elements, WAI-ARIA roles do not directly cause browsers to provide keyboard behaviors or styling [KKN<sup>+</sup>22].

While many implementations for dynamic behaviors are possible when JavaScript is involved (e.g. dynamic controls created by manipulating the `dom`, event delegation or event handler callbacks on existing HTML elements, and so on), and testing for their accessibility can be hard due to the nature of JavaScript (after all, it is a Turing-complete programming language), verifying the presence of WAI-ARIA attributes on elements *enhanced* by means of JavaScript could be done automatically: unfortunately, user agents do not even do that.

### 4.2.3 Finding accessibility resources

As illustrated in Section 2.3 many resources have been created to help designers, web developers, content authors, policymakers, managers and all other stakeholders involved in web development create accessible websites and applications. These resources can be classified according to different criteria:

***Their topic.*** Of course, due to the complexity intrinsic into making web content accessible and the necessity to *target* these resources to the stakeholders that could be most interested in them, various accessibility resources available on the Internet are focused on covering different topics and requirements.

***Their type.*** As there are so many requirements by different stakeholders in different phases of the development process to create accessible websites and applications, it is not surprising that a variety of accessibility-related resources are available; among others, these include:

- laws and regulations, *to enforce* the creation of accessible websites and applications depending on different factors (e.g. the type of product being developed, its purpose, its developer, etc.);

- technical standards, to *regulate* how web accessibility can be made possible and, in some countries, to determine more details on how the content of accessibility laws and regulations should be applied (e.g. to digital products and services, or to remove architectural barriers from buildings and other environments, or physical objects) and all the details that are necessary to do that;
- tools, a wide range of accessibility resources aimed at supporting stakeholders in achieving tangible, practical goals; these include resource to bridge the gap between more “theoretical” and the practical ones (such as conceptual frameworks or methodologies), accessibility testing and evaluation tools, tools to assist in the development process (e.g. linters and frameworks), and can come in many different types (browser extensions, command line tools, desktop applications, web services, and more);
- “learning resources”, including any support material; these include a very wide range of resources, such as the ones aimed at explaining complex concepts in simpler terms, or better illustrate how to leverage other existing accessibility resources to get the most out of them. As different formats may be more effective for “educating” different people or better suit the requirements of different stakeholders, learning resources available in a wide variety of formats including videos, written guides and tutorials, podcasts and infographics.

***Their complexity.*** As it may not be clear for a person approaching web accessibility for the first time, not all accessibility-related resources have the same complexity. In addition to this, the complexity of making accessible solutions created with different process or technologies is different as well. Unfortunately, you need to grasp the “big picture” before understanding that and the place where each resource fits in the puzzle.

*Their source.* “All animals are equal, but some animals are more equal than others”, proclaimed the pigs who controlled the Government in the novel “Animal Farm” by George Orwell. Unfortunately, not all sources of accessibility-related resources are equally authoritative. Quality checks must also be done to *assess* the truthiness and effectiveness of each resource, but this can be done only if you have a certain amount of experience with web accessibility, and in some cases even requires non-trivial competencies and abilities. Thanks to the latest advancements in digital technologies it has become much easier to become “content creators” over time; as the quantity of available content increases, the more urging and relevant the problem of ascertaining its quality becomes.

Even if the creation of innovative development and accessibility testing tools is key in the research described in this PhD thesis, many great and immensely useful resources that relate to web accessibility are available on the Internet. However, such resources are often fragmented, scattered across different websites, and therefore are hard to find if you do not know what to look for, where and how. In addition to this, fully understanding the content of such resources often requires previous knowledge about accessibility-related topics; people may not have such knowledge and, even worse, may not realize they need it to avoid misunderstanding their content.

Therefore, the creation of a well-structured, categorized collection of accessibility-related resources has been included in this research. Such a repository is built with the purpose of helping developers getting started with web-accessibility, as well as acting as a reference for all high quality accessibility resources available on the Web that may serve experienced members from the accessibility community as well.

## 4.3 Making web accessibility more accessible

Having outlined all the reasons leading to the approach to *rethink web accessibility from the ground up* proposed in this research project, we can now discuss it in more details. In this section the design principles, decisions and philosophy behind the creation of the **AX** framework and a couple of side-projects related to it, the innovative accessibility testing and evaluation methodology implemented in the **SAHARIAN** browser extension, and **A11A** will be discussed, explaining how they can help to solve actual problems faced by web developers while creating accessible websites and applications.

More specifically, the approach proposed in the research project documented by this PhD thesis can be summarized in three distinct, yet strictly related principles:

1. Guarantee that the markup used in any website or application to implement any necessary element is always accessible. This can be achieved by extending HTML through the **AX** framework, a set of web components specific to the logical, structural and semantic characteristics of their intended use, and whose markup is fully accessible *by construction*.
2. Provide innovative manual accessibility testing and evaluation tools to make it easier for non-disabled developers to *perceive* directly accessibility issues in their code, as well as their impact on people with disabilities. This is achieved by a specific methodology (implemented in the **SAHARIAN** browser extension) with two ambitious goals:
  - replacing the normal presentation of the page with a special visual representation based only on the accessibility markup: styling and positioning choices are deactivated and replaced with ones totally and completely based on the accessibility information conveyed to assistive technologies. Accessibility issues are represented in a way that lets the developer *immediately* understand that something is wrong, what the issue is, and the affected element;

- mapping the usual interactive behaviors allowed by web browsers onto the corresponding (and equivalent) actions that the page allows people with disabilities to perform via assistive technologies. In this way, a non-disabled developer can keep interacting with the page or applications as it is used to, but *immediately* understand whether the same interactions are possible for assistive technology users or not;
3. Making existing accessibility-related resources more accessible by letting non-disabled developers find and “make sense” of them, as well as the context in which they exist, much easier. This is achieved by creating **A11A**, a categorized, structured collection of accessibility resources available on the Internet proposed in a website that *recalls* different characteristics of a wiki, but the interactivity of a blog. Resources are classified according to various criteria: their type, purpose, intended audience, and so on.

Therefore, the final result of this research project, *making web accessibility more accessible*, is achieved by means of three different artifacts (the SAHARIAN browser extension, the AX framework, and the A11A website), a couple of supporting (yet independent) additional tools (the *Missing WAI-ARIA Role Explorer (MARE)* and the `aria-service` library).

### 4.3.1 Introducing SAHARIAN

To *make web accessibility more accessible*, The first topic that has been dealt with as part of the research project documented in this PhD thesis is that of improving the current situation about manual accessibility testing of existing websites and applications. While part of this process can be automated by means of specific tools, it is a key idea behind this project that manual accessibility testing is not going away anytime soon: in fact, evidence shows that there are issues that can be assessed properly and effectively only by humans.



Existing accessibility testing and evaluation tools center around the idea of *presenting* the detected accessibility issues by *communicating* the element affected by them. Such communication can happen by injecting specific visual indicators close to that element, or simply providing the line(s) of code introducing the issue. In addition, such messages can contain useful information such as the type of error and pointers to additional resources on how to fix them. However, to *effectively* use these tools, developers need to be aware of how they work, the types of accessibility issues they can detect, and have the necessary knowledge to understand how to fix them or the reasons why a reported issue is actually a problem for people with disabilities. Thus, it can be argued that they use an *indirect* approach, as they do not enable web developers to *perceive* accessibility issues and their impact on people with disabilities in first person.

In contrast, the innovative methodology to perform manual accessibility and evaluation proposed in this project originates from a completely different point of view, that of making web developers *identify* accessibility issues by leveraging their existing knowledge about web technologies and how “things should work”, and actually *experience* the impact of accessibility issues on people with disabilities in first person. In other words, accessibility testing results are represented in a way that allows web developers to *intuitively* understand the conveyed information, and to get actionable feedback about the accessibility barriers experienced by people with disabilities.

The theoretical foundations of the proposed approach originate from a simple and widely documented (especially in the educational area), yet still underestimated principle: since childhood, humans learn by associations and similarities. In the same fashion, the proposed methodology consists of mapping accessibility-related information on concepts that developers are already familiar with. Keeping into account the most used interaction tools used by non-disabled people and their habits, mouse operability and visual rendering have been selected as the most obvious and intuitive means of communication for this methodology.

## Sandwich Condiments

- Lettuce
- Tomato
- Mustard
- Sprouts

Figure 4.3: Visual rendering for a checkbox group for non-disabled users as can be expected by non-disabled developers

Concerning the latter, in fact, the so-called “at a glance” perception of a website is often enough to instantly recognize if its visual rendering exhibits critical issues. This can be exploited to communicate how accessible a web page is: thanks to modern web technologies, swapping the actual visualization of a web page is an easy step. The non-trivial part, however, is finding a representation that lets developers *immediately* discern whether a certain page fragment is accessible or not and, in greater detail, understand what accessibility issues are present and provide the necessary guidance to fix them.

To accomplish this result, two related “properties” of page elements have been exploited:

- the visual representation of user interface elements;
- the fact that, stylistic opinions aside, there is a common sense of what a widget is (i.e. checkbox, link, button, toolbar, etc.) and how it can be immediately recognized by looking at its visual representation within a web page.

Figure 4.3 shows a reasonable visual representation of a checkbox group that can be expected by non-disabled developers.

Therefore, the challenge becomes building a visual representation of such widgets so that it fully satisfies the following requirements:

- when the widget is fully accessible, its representation “looks all-right” and reasonably fulfills developers’ expectations on how the widget should be rendered;
- when an element exhibits any accessibility issues, its visual representation differs from what is expected in ways that allow developers to easily spot what is wrong;

But of course page elements can also be interactive, meaning that they support specific interaction patterns depending on their nature and purpose. However, this perfectly aligns with the objective of the proposed methodology: in fact, complex concepts (like keyboard focusability and operability), once again, can be mapped to developers’ expectations. To better illustrate this relationship, a simple example can be considered. If a mouse user (and most non-disabled developers likely are) encounters a checkbox, by clicking on it, he or she expects that:

- the checkbox state is toggled;
- the checkbox visual representation is updated to reflect its state change;
- the updated visual representation of the checkbox lets him or her understand whether it is enabled or not.

When it comes to interaction patterns, it is well documented in the literature that people with disabilities carry even stronger expectations on how widgets should work; continuing with the checkbox example, for instance, a screen reader user expects that:

- the checkbox can be focused using the keyboard or similar commands provided by the screen reader;

- pressing the space bar key when a checkbox is focused (or similar commands provided by the screen reader), the checkbox state can be toggled;
- the screen reader informs about the state change;
- the conveyed update provides information on whether the checkbox has been enabled or disabled.

Paying closer attention to the described scenarios, it is easy to spot similarities on the fact that both non-disabled users and people with disabilities expect to be able to interact with each widget in specific ways depending on its nature. This makes a perfect candidate in order to map the interactions expected by people with disabilities (that should be supported by each widget in order to be fully accessible) to interactions expected by non-disabled developers. Continuing once again with the checkbox example to better illustrate these mappings, for instance, the behavior of a checkbox can be altered so that it can be toggled using a mouse *if and only if* it can be toggled via the keyboard as well.

The most obvious consequence that naturally arises from this approach is that whenever a non-disabled developer tests a widget for its accessibility, he or she is using his or her very well established knowledge; most notably, when he or she finds out that something is not working as expected, in a cool fashion it becomes obvious what to look for.

The same reasoning that has been illustrated for the checkbox example can be extended to all the main widgets commonly found in web pages and applications. This means that the proposed approach is suitable for application on a wide variety of page elements. By definition, if the type of widget that is being dealt with (thus its expected interaction patterns) cannot be precisely determined, assistive technologies wouldn't be able to do it as well: as this method is intended for usage in manual accessibility testing, this issue can clearly and prominently be reported to the developer.

The end result of applying this method is that the developer keeps on

using the mouse, keyboard and eyes to test the web application that is being designed, but in a different visual context that is completely based on the accessibility markup, that makes the page understandable and usable proportionally to the correctness of the accessibility markup only. A detailed discussion of the features implemented in the SAHARIAN browser extension and the accessibility requirements supported for testing and evaluation of web pages and applications can be found in Appendix B.

### 4.3.2 Introducing the AX framework

In the vision behind the research project documented in this PhD thesis, other than helping developers test and evaluate the accessibility of their code, it is critical to *support them* to produce content that is accessible by default. As explained in Section 4.2.2, HTML provides all the mechanisms that are necessary to create such widgets. Yet, for each element multiple implementations with different accessibility degrees are available, and the accessibility of a web page does not influence at all its visual rendering.

Therefore, as part of the research project an approach to change the *status quo* has been pursued: while reinventing HTML is of course unfeasible, the goal of *rethinking its usage from the ground up* becomes possible thanks to the introduction of a new, innovative declarative framework called AX.

By definition, the AX framework should enforce the generation of accessible markup as much as possible. Being a declarative solution helps with this, as the correctness of hierarchical relationships (i.e. preventing the usage of a child components out of its designed parent element) can be easily enforced as required in order to generate accessible markup. Additionally, specifying required parameters when instantiating components (take the case of form control labels as an example) can be enforced as well. When such conditions are not met, the framework should render the offending component in a way that makes it obvious that something is wrong and providing instructions on how to fix it instead.

Implementing the AX framework is made possible by the extension of HTML

through the use of components. Introduced and promoted by all major web development libraries currently in favor, components are small, autonomous modules containing markup, styling and executable code that can be aggregated and composed to build full web applications with reliable and sophisticated functionalities.

Therefore, when using the **AX** framework, **HTML** is replaced by an open set of elements each of which is mapped onto a complete component that is responsible for handling all the requirements necessary for its actual implementation, including:

- generating the most appropriate markup to represent it;
- support all interactions commonly expected from that component;
- allow developers to customize its appearance.

In this way, the **AX** framework can act as an intermediate layer between the developer and the actual markup rendered by user agents, de facto becoming responsible for ensuring the accessibility of the generated markup. As part of this process, it can also determine the most suitable **HTML** markup representation for each element among the many that are possible, shifting this burden away from developers. By doing so, it also shifts away from the developer the responsibility of looking at guidelines and techniques for implementing that component in a way that is accessible, and most of the effort to determine which solution is the most appropriate to each case.

By adopting this approach, the **AX** framework can *complement* the set of elements natively provided by **HTML** as well: indeed, the fact that an element is not natively supported by **HTML** is not a limitation, as the component to support it can be provided as well. This is the way in which the **AX** can fill the gap of providing elements that are commonly needed during web application development, yet not provided by **HTML**. This brings away from developers the burden of dealing with *custom implementations* for such elements, understanding how to expose them *semantically* to assistive technologies, de-

terminating the interactions commonly supported by assistive technology users (e.g. keyboard navigation support), and implementing them.

For example, consider the instantiation of a text input field within a web page. In contrast to the scenarios described in Listing 4.1 and Listing 4.2, it becomes as simple as shown in Listing 4.3:

Listing 4.3: Creating a text input field using the **AX** framework

```
<text-field label="Name:">
```

Upon rendering the page on the browser, or through a compilation process, the code fragment shown Listing 4.3 is automatically converted into HTML markup, with eventual styling and JavaScript code to add presentation and interactive features respectively, with the guarantee that the generated code is accessible!

As is often the case for newly introduced frameworks, it is very important for **AX** to be able to coexist with parts of a web page that do not use it. This would allow developers and content authors to gradually adopt the framework, as well as letting consumers use it since its early development phases (even if it does not include every component they need). Unfortunately, this consideration made the implementation process much more complex; however, whenever possible this requirement has been fulfilled.

Another important point to note is that the **AX** framework is not intended as a replacement for very well established JavaScript libraries designed to facilitate web application development (React [Met22], Vue.JS [Vue22], Angular [Ang22], to mention a few). Instead, it has been designed to be low-level enough to be used in combination with such libraries.

One might argue that in such a situation a developer could easily mess up with the framework internals, thus vanishing the original efforts in guaranteeing the accessibility of its generated markup. In order to minimize this risk, UI state management (operations such as enabling a checkbox) is built-in into the framework, so that a developer does not need to manually change and/or alter the markup generated by the framework. Rather, the ability to provide callbacks for being notified and act upon significant events is provided at the

framework level in order to minimize unwanted side effects caused by their custom implementation. Whenever necessary to guarantee the accessibility of a certain component, handling of significant events (for example support of specific keyboard shortcuts) is built-in into those components.

Finally, in order to provide the accessible equivalent of a majority of components whose use is nowadays widespread in web development, a strong and possibly controversial principle has been adopted: making an opinionated decision is better than not making a decision at all. The application of this principle should be restricted to the minimum, so that the framework does not condition unnecessarily the developer, yet its adoption is critical in order to provide working components even in situations in which multiple solutions may be acceptable but would need a conscious implementation strategy. For instance, there are many different ways to implement accessible date pickers in HTML, each of which would require a different markup approach. The AX framework will only provide one of them, yet providing for the necessary flexibility to support different requirements (e.g. time granularity) and visual appearance customization.

Last but not least, it is worth noting that the kind of *static analysis* implemented by the AX framework cannot catch all possible issues that may arise during its usage, either due to technical reasons (e.g. the user agent specific behaviors) or the necessity of providing developers more flexibility. Things such as visual appearance customization, which are indeed critical to the success of a web application, should be extensively supported. To conciliate such requirements with the necessity of ensuring the accessibility of the generated markup is not altered by such flexibility (for instance, think of avoiding the introduction of color contrast issues by means of CSS rules), AX also integrates a standalone, automated accessibility testing engine to catch all issues that cannot be detected otherwise. While the framework has not been developed during this research project, the way in which results are presented and their integration in AX have been.

Ideally, it would have been great to include in the AX framework all the



components that developers require in order to implement all sorts of websites and applications; unfortunately, given the amount of such elements and the necessity of *demonstrating* the feasibility of the proposed solution (which is necessary as it is part of an innovative research project) this was not possible. A detailed discussion of the implemented components, as well as the reasoning that led to picking them instead of others, is available in Appendix C.

### 4.3.3 Introducing A11A

Other than creating innovative resources, the goal of *making web accessibility more accessible* of the research project documented in this PhD thesis is also achieved by making existing accessibility resources much more accessible for people who are not accessibility experts. While there are objective technical difficulties to make a website, application or any content accessible, as discussed in Section 2.3 a very large number of standards, guidelines, laws, regulations, educational resources and support tools are available to help developers and content creators achieve the desired outcome. Unfortunately, there is evidence that developers have difficulties finding these resources as they need to know:

- what to look for, where to look it for and how;
- how to *select* the most suitable resources across a larger set of them;
- how to distinguish the most authoritative sources of information from the others;
- how to evaluate the correctness and effectiveness of the available resources to select the best ones;
- how to understand whether such resources are relevant to your project or not.

Undoubtedly, this becomes part of the complexity of creating accessible content, increasing the additional effort (in terms of time, money, and cognitive efforts) required to achieve the desired result.

This is the reason why as part of the research project documented in this PhD thesis the creation of A11A has been included as a fundamental goal to make web accessibility more accessible. A11A can be thought as a sort of *accessibility wiki* to collect all the useful accessibility resources available on the Internet. In order to facilitate their retrieval, resources on the website will be classified according to different criteria:

- their nature, distinguishing between guides, tutorials, frameworks and libraries, other utilities, command line tools, plug-ins for existing tools, videos, and so on;
- *tags*, designed to group resources according to the matters they refer to; such tags will group only resources that are related to them, creating useful *entry points* to discover the most suitable accessibility resources for your needs.

Each accessibility resource published on A11A is accompanied by a short presentation explaining how to get the most out of it (e.g. to indicate clearly in which context is intended to be used, its intended target audience, eventual known limitations, and so on). While some repositories of accessibility resources do already exist, A11A features three main differences when compared to them:

1. It will be a comprehensive, 360-degree wiki, covering all topics that can be related to digital accessibility;
2. all resources published on the website will be manually tested in order to assess their effectiveness, and validate that they work as advertised.
3. The way the resources are organized and the terminology used throughout the website to introduce them is carefully crafted to facilitate web developers in finding available resources.

Additional resources that explain how to use the more technical ones are also included in A11A, along with information on the role they play. Together with other usability enhancements (such as an advanced search engine, related content suggestions, and so on) and some features to make the website more interactive (such as comments), they can make A11A an accessibility-related resource whose purpose is to make other accessibility-related resources more accessible.

Finally, from an educational standpoint there is nothing that can be more effective than good examples. This is the reason why A11A is committed to ensuring the adoption of the latest Web Accessibility best practices throughout the website: while this may seem counter-intuitive, unfortunately it is not uncommon for accessibility-related websites to exhibit minor accessibility issues themselves.

## 4.4 Introducing MARE and ARIA-Service

As illustrated in Section 2.1.3, the *WAI-ARIA 1.1* specification plays a very significant role in letting developers produce accessible websites and applications. Indeed, it can be used to *enrich* the semantics of HTML elements by means of “special” attributes (roles and states) that alter the way in which user agents expose them to assistive technologies. This gives designers and developers a great power, as they can alter the way in which HTML elements (and, even if not specifically covered in this PhD thesis, artifacts implemented using other host languages such as *SVG*) are *mapped* to native platform accessibility APIs, thus how they are perceived by assistive technology users. But, “with great power comes great responsibility”. Consequently, using this specification is not as trivial as leveraging native HTML elements to implement accessible websites and applications. Its complexity and the way in which it impacts assistive technology users makes it necessary to evaluate very carefully whether to use it, when and how.

As already outlined for other artifacts that are part of the research project

documented in this PhD thesis, removing unnecessary complexities (or helping developers overcome them) is critical to *making web accessibility more accessible*. Therefore, two different tools to help developer use the WAI-ARIA specification have been created:

- *MARE (the Missing WAI-ARIA Role Explorer)*, an interactive tool to let developers explore roles, states and properties provided by the WAI-ARIA specification and how to use them most effectively;
- *ARIA-Service*, a JavaScript library that implements all dynamic behaviors required by specific WAI-ARIA roles.

While these two tools were not part of the original research project, their necessity became evident during the development of the **AX** framework and the **A11A**, as well as by general feedback on developer's perceptions of this specification gathered by myself during this PhD course. Indeed, W3C and other members from the accessibility community (both individuals and enterprises) has produced great resources to support developers in understanding and using this specification properly. However, it is argued that it still remains very complex, and perhaps in some cases even *unaffordable* to implement correctly (think to the hundreds of code that are necessary to implement properly keyboard support for a tree view, for instance). Both the Missing WAI-ARIA Role Explorer (MARE) and *ARIA-Service* can provide a significant contribution towards simplifying the usage of this specification and ensuring that it is used correctly.

#### 4.4.1 Introducing MARE

Aimed at helping web developers understand what attributes the WAI-ARIA specification provides and their purpose, MARE is an interactive tool that lets anyone *explore* the WAI-ARIA specification interactively. More specifically, MARE allows to get an overview of all roles provided by the specification, an overview of all available attributes, and most of all a clear indication on which attributes can be used with which roles: indeed, not

all attributes are supported by all roles. Using unsupported attributes on specific roles can have at least two effects, in order of increasing severity:

1. thinking that information is exposed properly to assistive technologies, while indeed it is not (as they expect to find data by querying different attributes from than the ones used);
2. assistive technology users cannot properly interact with the widget having a certain role and one or more unsupported attributes, as assistive technologies cannot deal with such a combination; while it would be expected this scenario not to be more critical than the previous one, in the complex equation that makes web accessibility implementations possible<sup>1</sup> there are simply too many things that can go wrong to expect it to be robust enough to let users interact “reasonably well” with such incorrect markup.

For each role in the WAI-ARIA specification, MARE allows any user to know:

- A brief overview of its semantics, such as information on the type of widget it is intended to represent;
- *Required attributes*, the list of attributes that *must* accompany an element where the role is applied in order to complete its semantics;
- *Supported attributes*, a list of WAI-ARIA attributes that are not mandatory for any element where the role is applied, but can be specified to further enhance its semantics;
- Information on how the accessible name of an element with that role is computed (e.g. whether it is computed from its children, or the author must explicitly provide it);

---

<sup>1</sup>As a quick recall, the complex equation that makes web accessibility implementations possible involves a two-way information exchange in between the operating system, the browser and assistive technologies by means of one or more platform APIs.

- Explicit and very clear information on parent-child relationships among WAI-ARIA roles, whenever they exist and are necessary. Consider an element with role `radio`: its parent or owning element must have the `radiogroup` role; vice-versa, an element with the `radiogroup` role should only own or have child elements with the `radio` role.
- All keyboard interactions that should be supported by an element having that role.

While this information is available in the WAI-ARIA specification and its supporting documents, along with practical code examples on how to use various roles properly, MARE may make accessing such information easier: it is collected in a single place, interactively filterable to see only information relevant to the role you are interested in, and *simplifies* some concepts that by their nature are explained in a pretty complex way in the WAI-ARIA documents suite.

MARE, however, provides specific information for all WAI-ARIA attributes as well; notably, for each attribute MARE indicates its purpose, the *allowed values* (whenever applicable), and the roles or elements it can be applied to.

#### 4.4.2 Introducing ARIA-Service

As explained in Section 2.1.3, leveraging the WAI-ARIA specification to make custom implementations of elements and widgets accessible allows the developer to *enhance* the semantics of such implementations, so that they can be exposed properly by web browsers (through the platform accessibility APIs) to assistive technologies. However, only denoting the semantics of an element is not enough to make it accessible. In some cases, in fact, assistive technology users have *strong expectations* on the type of keyboard interactions it should support depending on its semantics.

Recalling one of the fundamental principles behind the WAI-ARIA specification, *a role is a promise; it is the developer's responsibility to fulfill that*

*promise.* Developers must provide correct implementations for all behaviors expected by assistive technology users for elements with a certain role. For instance, consider the case of implementing a list of tabs (with their associated tab panels), a widget for which HTML does not provide native elements. Using attributes from the WAI-ARIA specification is the only way to implement it in an accessible way; however, other than having the appropriate attributes, such widget should support the following interactions to implement properly *keyboard navigation*.

- pressing the *tab* key:
  - when focus moves into the tab list, places focus on the active tab element;
  - when the tab list contains the focus, moves focus to the tab panel (which must be next in the tab sequence).
- When focus is on a tab in the tab list, pressing the *Right Arrow* key:
  - moves focus to the next tab in the tab list;
  - if the currently focused tab is the last in the tab list, moves focus to the first one instead;
  - optionally, activates the newly focused tab.
- When focus is on a tab in the tab list, pressing the *Left Arrow* key:
  - moves focus to the previous tab in the tab list;
  - if the currently focused tab is the first in the tab list, moves focus to the last one instead;
  - optionally, activates the newly focused tab.
- When focus is on a tab in the tab list, pressing the *Home* key moves focus to the first tab in the list, optionally activating it.
- When focus is on a tab in the tab list, pressing the *End* key moves focus to the last tab in the list, optionally activating it.

Implementing this logic is not an easy task, as the `JavaScript` code to do that is tens of lines in length, prone to errors, and in some circumstances even requires *truly* understanding assistive technology users expectations from real users, as the documentation does not provide all the necessary details (and as they say, often “the devil is in the details”). While documents such as the WAI-ARIA Authoring Practices Guide [Wor22a] provide practical code examples to implement such logic, they may contain bugs and adapting them to the specific usage in a project may not be trivial. Additionally, it should be noted that the logic behind keyboard navigation for a tab list is not the most complex one; elements such as menubars, treeviews, grids and collapsible comboboxes do require support for interactions whose implementation can be considered at least an order of magnitude more complex if done properly.

Therefore, as part of the research project documented in this PhD thesis, the creation of a `JavaScript` library to shift away this complexity from developers has been designed and implemented. Called `ARIA-Service`, the main idea behind this library is that if developers use *correct* WAI-ARIA markup on custom widgets, the library (after being included in the web page, of course) *automatically* provides the code that is necessary to support all keyboard interactions necessary to make them accessible for assistive technology users. Whenever the documentation on such keyboard interactions is not clear or incomplete, or the provided code examples are buggy, the ones provided by native operating systems have been integrated<sup>2</sup>. In this way, from the developer’s perspective, implementing accessible widgets leveraging the WAI-ARIA specification becomes as simple as implementing their counterparts using native `HTML` elements. This makes a step forward towards *making web accessibility more accessible!*

---

<sup>2</sup>Being both a blind computer scientist, thus a screen reader users, who uses all major operating systems on a regular basis helped a lot with this task.



## 4.5 Introducing A11YVT and the *SCAMP* methodology

The second branch of the research project documented in this PhD thesis involved using digital technologies to make indoor environments and cultural heritage more accessible. More specifically, two specific problems have been dealt with:

- A11YVT, the prototype of a system to let blind and visually impaired people, as well as their sighted peers, *explore* indoor environments;
- using a combination of digital technologies, scientific and art skills to make some physical objects (scientific decomposable artifacts with a high historical and educational value, thus cultural heritage) more accessible.

These two contributions, the methodologies behind them and the design decisions leading to the artifacts that have been produced will be discussed in this section.

### 4.5.1 Introducing A11YVT

When it comes to leveraging digital technologies to make indoor environments more accessible, as discussed in Section 3.3 a very wide variety of possibilities have been explored in the literature; from ways to improve the accessibility of maps to indoor navigation systems, including special aids and methodologies on how to help blind and visually impaired people move independently. As part of the research project documented in this PhD thesis, an approach to make indoor environments more accessible leveraging virtual tours has been explored.

Consisting of simulations of existing locations, composed of a sequence of videos, still images or 360-degree images and other multimedia elements such as sound effects, music, narration, text and floor maps, virtual tours are

gaining momentum and their potential is only starting to be unleashed. They are tools that could be effectively used in many different contexts (games, tourism, house-hunting, etc.), and the COVID-19 pandemic has let emerge how they can be useful and efficacy in those situations where it is not possible to access physical places and attend events. Given the importance of visual elements, considering that they let users interact with the virtual environment and its points of interest through them, and their very remarkable emphasis on the need for navigation and orienteering mechanisms within the virtual environment, virtual tours can represent digital barriers for people with disabilities, in particular those who are blind or visually impaired.

Therefore, as part of the research project documented in this PhD thesis, the prototype for *Accessibility for Virtual Tours (A11YVT)* has been developed. A11YVT is a generic, interactive tool whose main purpose is to enable people to explore environments they are not familiar with, so that they can discover all the characteristics of a certain space (its shape, history, etc.) at their own pace. As the acronym implies, A11YVT accomplishes this goal by letting people experience *virtual tours* of a given space so that they can inspect all the *Points of Interest (POI)* that it contains. Being a generic solution, A11YVT makes no assumptions about the nature of the POI that can be present in a certain environment, so that many different scenarios (with different requirements) can be supported in a single tool.

While different systems with similar characteristics do indeed exist, none of them takes into account the *special requirements* of blind and visually impaired people in this area. In fact, as widely discussed in the literature, due to the lack of sight (or having only a residual that does not provide a good visual perception of a certain space), blind and visually impaired people tend to employ extremely specific strategies when it comes to familiarizing themselves with environments with which they are not familiar. Therefore, to be truly accessible (and effective) in reaching its goal, it was clear that *A11YVT* had to provide specific features to support such strategies.

At a first glance, providing these features while at the same time support-

ing the (quite different) needs of sighted people for environmental exploration may seem unaffordable. However, with careful decisions taken from the early design phases, this goal can be achieved; thus, it is worth providing a comparison of these requirements, discussing how they impacted the design of A11YVT, and how they can be combined to provide an engaging user experience for both blind and visually impaired people and their sighted peers.

First off, when it comes to letting blind and visually impaired people explore environments that they are not already familiar with, outdoor and indoor spaces can pose challenges that are significantly different. These challenges arise from many distinct factors, including:

- differences in size and shapes of the environment. In general, outdoor spaces are much larger than indoors, and they exhibit typical “patterns” (such as streets and squares) and “landmarks” (crossroads, traffic lights, sidewalks, and so on) that you cannot find indoors.
- Different hazards you may face while exploring. Outdoor and indoor environments can expose blind and visually impaired people to hazards that may or may not become a danger depending on various circumstances; when they are not familiar with the environment, they require special attention. Some of these hazards pose comparable challenges (think of road signs placed in the middle of a sidewalk, compared to pillars positioned in the middle of a corridor), while others can be vastly different in their nature (think of street crossroads, compared to random object that may be left unattended alongside corridors).

While different hazards and environmental characteristics can all be dealt with, blind and visually impaired people have to pay special attention to them, in particular while exploring new environments with which they are not already familiar. Consequently, they tend to employ strategies that may seem similar, but are indeed widely different; as they say, often evil is in the details. This implies that the set of features required to support indoor environments exploration can be different from those required to support

outdoor spaces exploration; this is the reason why it has been decided to focus only on indoor environments.

As widely discussed in the literature, blind and visually impaired people cannot have what we may define as the *perception as a whole* of their surroundings. While sighted people can get a picture of an environment *at a glance* (e.g. simply by having a look around), blind and visually impaired people have to *construct* this picture by exploring the environment in a physical, tangible way so that they can acquire enough *landmarks* to determine their exact position at any given time and *compute* paths to reach their destinations. As extensively discussed in the literature, such landmarks can be anything, including *details* that sighted people often ignore.

Finding a balance between providing enough context to let blind and visually impaired people acquire their landmarks, while representing only data that can be useful to their sighted peers to avoid overwhelming them with information, was quite a challenge. Therefore, several assumptions to *simplify* the environment representation and make it effective had to be made.

To determine which assumptions would be *reasonable ones* and avoid making A11YVT ineffective, the way in which blind people approach exploring an unknown indoor environment has been studied empirically: after several observations, a pattern emerged clearly. First off, they establish a *focus point*, which is a point in the environment that they feel comfortable reaching (i.e. they know how to reach it confidently), it can be a door, the center of the room, a pillar, or anything else on a purely subjective basis. After that, they try to determine the position of the other landmarks relative to the *focus point*, so that they can calculate short paths from the focus point to each landmark. By iterating the process on all landmarks, and eventually including paths with more than one landmark, blind people can construct the equivalent of the picture of the environment that their sighted peers can get *at a glance*.

Therefore, it has been confidently assumed that in order to let blind and visually impaired people explore an indoor environment they are not familiar

with effectively, A11YVT should support this pattern through its interactive features. We determined that, at any given moment in the *tour*, the user must be positioned in a landmark. The initial landmark from where a virtual tour starts is established by the data model that contains the information about the environment being represented. Since A11YVT was intended to be interactive, it was easy to let users *move* in the virtual environment representation. In order to let users *perceive* the position of landmarks, we used the so-called *clock navigation* method<sup>3</sup>.

In order to make the creation of a prototype easier, A11YVT has been conceived as a web application to be used from desktop operating systems. Therefore, various keyboard commands have been implemented:

- to move *point by point* clockwise and counter-clockwise, allowing to explore the surrounding of the current focus point;
- to change the focus point, *simulating* the user movement to reach it;
- to get additional, useful information about landmarks.

But indoor environments are obviously three-dimensional spaces: for the environment representation to be realistic, *A11YVT* had to provide the features necessary to explore a *3d model* of the modelled space. Once again, the design of these features has been influenced by the way in which blind people explore indoor environments. The concept of a *z-axis* has been introduced (with specific commands to move along it) so that the user's focus point can be moved at different heights. In this initial prototype version, only three different height levels are supported: a floor level, an intermediate level, and a third (higher) level. Landmarks can then be placed at the most appropriate height; obviously, they can fit at a single height level (think of a table), span two height levels (e.g., a locker), or all three (such as a pillar).

---

<sup>3</sup>Basically, the person's position is assumed to be the center of a clock, so that at most twelve additional points can be placed around that point in the exact positions that the corresponding numbers would occupy on a clock face.

Interestingly, all the features offered to let sighted people explore an environment are quite similar to the ones required by blind and visually impaired people, however, they need to *act* differently. Assuming that blind and visually impaired people will always use assistive technologies, by leveraging the tools that current digital technologies offer these differences can be dealt with pretty easily.

### 4.5.2 Introducing the SCAMP methodology

Finally, as part of the research project documented in this PhD thesis an attempt to leverage digital technologies to make physical objects (more in particular, cultural heritage) accessible has been done. The decision to work on specific objects has not been randomly made: they have been chosen as they represent a *proxy* towards reality. Improving their accessibility, in fact, would improve the accessibility of a specific field that has not been traditionally the center of attention for accessibility efforts. Additionally, a very strong inspiration led to this choice.

It was a sunny day. Birds were chirping. And in the midst of all it was the Herbarium in the Botanic Garden and Herbarium<sup>4</sup> of Alma Mater Studiorum - Università di Bologna<sup>5</sup>, holding all sorts of flowers, plants and herbs in a stunning grass. You can touch them, smell them, feel the vibes only such an atmosphere can give you.

Unfortunately, a PhD thesis cannot bring you the exact same atmosphere of that day, while walking on the grass and learning about these artifacts. But you can imagine how surprising it was to find out that part of this atmosphere

---

<sup>4</sup>The Botanic Garden and Herbarium of Alma Mater Studiorum - Università di Bologna, <https://sma.unibo.it/en/the-university-museum-network/botanic-garden-and-herbarium/botanic-garden-and-herbarium>.

<sup>5</sup>Alma Mater Studiorum - Università di Bologna, <https://www.unibo.it/en/homepage>.

could be recreated. In the late 19<sup>th</sup> and early 20<sup>th</sup> centuries, Robert Brendel founded the Brendel company: his goal was to produce highly accurate, very realistic models to aid in the teaching of Botany. He was a brilliant model maker, as his main goal in creating these *Brendel models* was to accurately capture the sensations of touching the real plants, the colors you would see if you looked at them, and the shapes you would see if you looked at them more closely.

One of the main strengths of these models is the fact that they were realized with *interactivity* in mind: they were designed to be passed around the classroom by students who would then manipulate and disassemble them to examine every little aspect. They have been realized with different materials (e.g. papier-mâché, wood, wax, metallic wire, canvas, gypsum, feathers, textile fibers, cardboard, rope) de facto being thought as a thing you should *experience* rather than you simply observe. Surprisingly, as time passed, they were left in a closet, nearly forgotten by everyone, taking powder, and being unavoidably ruined by the effects of time. Nothing was done to preserve or analyze them, and highlight their extraordinary characteristics: in deed, these models have an indisputable aesthetic, educational and historical significance. Due to their current state, the Brendel models are no longer manipulable; as a result, their most crucial feature (that is, enriching botany learning by haptic exploration) is lost.

After learning that there were many Brendel models with these characteristics and in the same situation lying around all over the world, and that there are other scientific, manipulable artifacts with a high historical value that could benefit from such a treatment, it has been decided to create Scientific Collections Accessibility Making Process (SCAMP); it consists of a generic methodology to make them *accessible again* by means of digital technologies but *preserving* their physical nature. Such methodology consists in a theoretical and practical *modelling* process involving the usage of multimedia technologies to *enhance* the specificities of modelled objects. The expected outcome of the process is that, after applying it to an artifact, the end result

*embodies* the same values as the modelled object *enhanced* in one or more ways. Given the importance and the intrinsic characteristics of such objects, the described process can be considered both as a way to preserve them and make them *accessible again*. In this case, though, the term *accessible* assumes two different, yet strictly related meanings:

1. *Make them more accessible to the general public.* By definition, due to their historical significance these objects are no longer suitable for being manipulated. This is the reason why preserving measures are generally adopted: unfortunately, these necessary measures make them inaccessible to the general public and, in particular, to blind and visually impaired people.
2. *Make them accessible for blind and visually impaired people.* The strong focus on the tactile feelings which model makers often put onto these models make them fascinating artifacts with regard to their accessibility for blind and visually impaired people, as they use their touch to *explore* objects and *perceive* their characteristics. By definition, such objects act as *proxies* for explaining more complex (eventually abstract) concepts they represent which, in turn, may be even less accessible (for instance, due to a lack of tactile resources illustrating them). Therefore, improving their accessibility for blind and visually impaired people to these objects can potentially improve the accessibility of other scientific areas.

It is argued that by designing *an interactive, guided experience* which aims to let the user *perceive* all the characteristics of the original objects through a physical, tangible artifact realized with this purpose in mind, both problems can be solved in a *universal design* fashion. Each of these experiences must be created using contemporary technology and multimedia content to highlight all of a model's distinctive features; adopting such a multimodal strategy entails making its use more inclusive and accessible for people with disabilities, while also improving the user experience for the general public.



To build at least one prototype to prove the feasibility and assess the effectiveness of the described process, it has been decided to apply it to one of the Brendel models; to pick the right one, though, some criteria have been carefully established::

- simple enough to reproduce, yet representative for the most peculiar characteristics you can find in the Brendel models; this way, the successful implementation of that specific model proves the possibility to apply the same process for other models and other similar artifacts too;
- not too technically challenging in the fine details, yet challenging enough to act as a proof of scalability of the process to other objects;
- allowing the design of a simple, yet complex interactive user experience.

The natural aging process has made some models deteriorate over time; we decided to exclude them from our selection, as handling them would require knowledge not available in our team at the moment. After evaluating all the available models in the Bologna Brendel collection and excluding the most deteriorated ones due to the natural aging process, it has been decided to apply the process to create a prototype for the Brendel Model of *Phaseolus vulgaris* Linn. which is shown in Fig. 4.4 and has all necessary characteristics to be representative for all objects the process is intended to be applied to.

The proposed process can be broken in two distinct phases: the *reproduction of the model*, and the *design of a user experience* that exploits the obtained artifact to create a sophisticated, highly interactive experience that enhances the characteristics of the original object.

The main purpose of the model reproduction phase is to get a tangible artifact that can replace the original object. The obtained reproduction must accurately reproduce all characteristics of the original objects (especially their shapes and tactile feelings). Unfortunately, when dealing with these artifacts information on how they have been made or on the exact materials that have been used to create their parts is often unavailable. Therefore, in order to get a *tangible* item that reflects the characteristics of the object



Figure 4.4: The Brendel model of *Phaseolus vulgaris* Linn, illustrating the early stages of the germination process of a bean plant.

Source: Sistema Museale di Ateneo (SMA). Alma Mater Studiorum - Università di Bologna

being modelled, it is necessary to leverage an immaterial (digital) intermediate representation of the original object. When it comes to creating 3d digital representations of physical objects, *3D scanning* and *photogrammetry* are the elephants in the room. As accessing a 3D scanner was not possible, photogrammetry has been leveraged in this prototype phase: it basically consists of taking several pictures of an object with a high-resolution camera to capture a 360-degree view of it, so that a 3D-model of the object can be constructed by processing them with specialized software. Once obtained the digital 3D model, the problem of creating a tangible artifact becomes a matter of employing an accurate, fine-tuned 3D printing process.

With regard to the design of a user experience for each object, the process can be broken down into different phases:

1. *Analyze the original artifact.* It is very important to understand the *ratio* behind each model: which were the intentions of the author when creating it? What should the object illustrate? How is it supposed to do that? While answering these questions may seem obvious, reality showed that this is not always the case.
2. *Design multimedia content to accompany the recreated model.* It should be in line with the results obtained from the previous point. The user experience should *enhance* the original, intrinsic characteristics of the object, and not alter its nature; it should be seen as a way to add *multimodal* content into the mix.
3. *Design the interactive, guided user experience suitable for the recreated model.* It should exploit all the properties of the recreated model and multimedia content to let the user enjoy the characteristics of the original object in a new interactive, multimodal way.

With these considerations in mind, ensuring that the provided experience makes the original object more *accessible, both for people with disabilities and the general public* becomes a matter of keeping into account the needs of

these people while creating multimedia content, as satisfying other necessary requirements automatically arise *by definition* from the process itself.

# Chapter 5

## Implementing the research project

As illustrated in Chapter 4, the high level goals of the research project documented in this PhD thesis consist in making web accessibility more accessible and using digital technologies to improve the accessibility of indoor environments and physical objects with a high historical value (aka cultural heritage) for blind and visually impaired people. Having provided an overview of all the produced artifacts and the design decisions behind their development, in this chapter some technical details of their implementation will be discussed. Challenges faced in the process and the solutions adopted to overcome them will be illustrated as well.

As outlined in Section 4.3, the proposed approach to make web accessibility more accessible encompasses a combination of three different artifacts:

- the SAHARIAN browser extension, to implement an innovative accessibility testing and evaluation methodology to let developers *directly* perceive accessibility issues in their code and their impact on people with disabilities;
- the AX framework, to support the development of applications and websites that are *accessible by default*;
- A11A, a categorized and structured repository of accessibility-related developers to make finding them much easier for people who are not

experienced in web accessibility, and perhaps serves as a reference for accessibility experts as well.

Their implementation, the challenges met in the process and the decisions that have been made to overcome them will be discussed in this section.

## 5.1 Implementing the AX framework

As already explained in Section 4.3.2 part of the research project documented in this PhD thesis consists of helping developers to *create* applications and websites that are *accessible by default*. All the decision and the philosophy that makes this goal possible have already been examined in Section 4.3.2. In this section, though, the more technically involved aspects will be analyzed; or, in other words, the implementation how the AX framework and the challenges faced in the process (along with the solutions to overcome them) will be discussed.

First off, given the characteristics and the goals of the framework, the *web components* [GI14] specifications naturally constitute a solid foundation to build such a framework. Indeed, being standalone and self-isolated pieces of code that can encapsulate all they need to function properly (markup, styling, business logic, and so on), web components are a perfect match for the framework. While *web components* have been made popular by very well known JavaScript libraries (`react` [Met22], `Vue.js` [Vue22], `Angular` [Ang22], to mention a few), fortunately given their success and the many benefits they bring to modern web developer the W3C has enacted several standards that make the creation of web components possible *natively* in web browsers (i.e. without depending on such libraries). More specifically, the combination of W3C standardized technologies that have been leveraged to implement the AX framework include:

- *Shadow DOM*, to create a *DOM* representation for each web component. Essentially, this representation contains the *markup* generated by the component and is managed by its business logic;

- *Custom HTML elements*, that allows the HTML to map web components to *custom* HTML tags, effectively allowing the language to be extended;
- *HTML templates*, to facilitate the creation of the internal DOM managed by each component.

This combination of web technologies makes it possible for the AX framework to work either in conjunction with the aforementioned JavaScript libraries that initially promoted web components (thanks to additional tricks adopted during the implementation, such as special attention to the logic behind events dispatching), or *independently* as a standalone solution. Figure 5.1 shows a *Unified Modeling Language (UML)* class diagram that represents the internal structure of some components in the framework. While it represents a limited subset of the components made available by the AX framework, it allows some crucial characteristics of its internal architecture to be highlighted.

First off, as per the actual web components specifications, it is evident that the logic to implement each of them is *self-contained* within a single JavaScript class. It is worth discussing some relationships among these classes, though. As it can be easily noticed, all classes in the AX framework extend (either directly or indirectly) the `AXElement` class: this allows for the logic common to all components and boilerplate code necessary to make them work properly to be implemented only once in a single class, therefore facilitating code maintenance over time. In addition, this decision makes it possible to implement various utilities shared across all components to facilitate their development and optimize their performance. In turn, the `AXElement` class extends the `HTMLElement` class natively provided by the DOM implementation: this allows to leverage certain features (such as handling of event dispatching and CSS classes) exactly as they work in HTML without having to reimplement the necessary logic manually. One may argue that this *exposes* all components to intentional tampering from web developers (e.g. by DOM manipulations): while indeed this is true, current web technologies provide very limited support to prevent such abuses; additionally, even considering

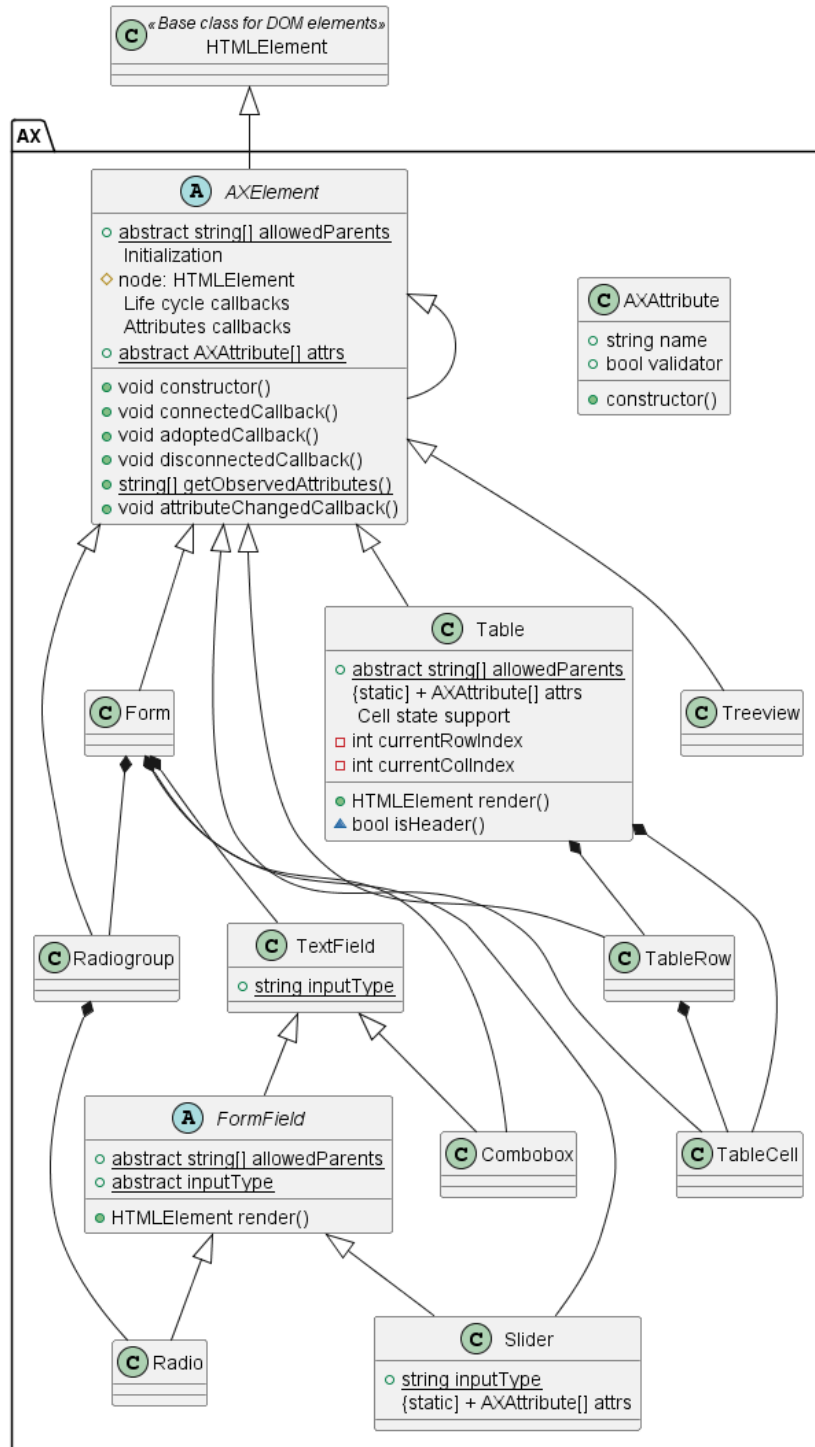


Figure 5.1: A class diagram showing some components in the AX framework and their relationships



this drawback, it is argued that the AX framework is still a huge step forward towards the creation of websites and applications that are *accessible by default*. In fact, this kind of tampering requires the developer to *intentionally and consciously* mess up with the framework internals.

The existence of the `AXElement` class finds even stronger justifications if very specific implementation details such as performance optimizations, subtle differences in the way in which different browsers handle web components, attributes handling, and the “hacks” that can be necessary to make the AX frameworks web components interoperable with other popular JavaScript libraries. For instance, consider the very simple requirement of *customizing* a component by providing some input, or passing it the *data* that should represent. Two related, yet different mechanisms have to be coordinated to get the desired effect:

- Attributes, whose values are defined *in a declarative way* in the HTML document. They are placed inside the element’s opening tag and always come in name/value pairs (originating from the syntax `name="value"`). When browsers parse HTML to create the DOM of a web page, they recognize the attributes and create DOM properties from them. An important limitation that must be considered is that generally speaking (but of course, the Web would not be the Web without exceptions to the general rule) the value of an attribute can only be a string;
- DOM properties, which are all properties of objects that constitute the DOM of a web page. They may arise from parsing HTML attributes, but can also be provided with other dynamic mechanisms;

Handling the relationships within properties and attributes in web components is not easy, as it must be kept into account that:

- there can be properties with no corresponding HTML attributes;
- there can be attributes with no corresponding DOM properties;

- if an HTML attribute and a DOM property with the same name exist, they are not synchronized automatically; instead, it is the component's responsibility to provide the necessary logic to do that and *reflect* the changes in the rendered markup.

This property reflection and value synchronization logic can be complex and prone to errors, and therefore it has been abstracted in the `AXElement` class wherever possible.

Speaking of hierarchical relationships among classes, it is also worth noting the particular case of *form fields*. Indeed, while developing the AX framework, it became apparent that different *groups* of web components shared very similar characteristics, thus required a very similar approach to be implemented. This is the case of form fields: indeed, accessible form fields require very similar markup to be implemented in an accessible way. This is the reason why the `FormField` class has been created to encapsulate such common logic, so that its subclasses have to deal only with the specificities of the field they represent. An evident exception is the case of the `Radio` class, which is intended to represent a radio button; as its specificities prevail on the similarities with other form fields, it extends the `AXElement` class even if it technically represents a form field.

To conclude the analysis of the class relationships among various web components of the framework, the *Table class* is emblematic of a situation that also emerged in other areas: various classes have to *strictly* cooperate to enforce the generation of accessible markup for the component they represent. Table cells, for instance, need to *coordinate* themselves with the table in which they have to be nested in order to determine whether they represent a row header, a column header, or a standard data cell. Similar reasoning applies to the `TableRow` class too. These entanglements are so crucial that careful decisions have been made to establish where each piece of this common logic had to be implemented, as well as designing a clear, yet effective API to make their existence (which cannot be avoided) sustainable from a code engineering point of view to facilitate the maintenance and

future development of the framework itself in the long run.

Finally, the UML class diagram shown in Fig. 5.1 gives an opportunity to illustrate of the *declarative* nature of the AX framework has been exploited to *enforce* the generation of accessible markup. First off, given all the characteristics, the problem can be broken down in two parts:

- ensure that all information necessary to the generation of accessible markup (for instance, think of alternative text for images or labels for form fields) is provided. This can be done by *validating* the attributes encapsulating this information, rendering *visually* appropriate error messages in case of problems;
- ensuring the *correctness* of hierarchical relationships (for instance, that radio buttons are contained in a radio group element) among the components nested within a page. The `allowedParents` field of each component class indicate which components can act as a parent for it, so that the correctness of nesting relationships can be statically validated while rendering the page.

But this is not the end of the story. In fact, two more factors get involved in this process:

- such analysis can be expensive, therefore having a bad impact during the rendering of complex web pages and applications (especially during dynamic DOM updates, which may result in visual artifacts due to slow code execution);
- unfortunately, there are checks that are *critical* to the generation of accessible markup that cannot be performed statically, either because external and uncontrollable factors come into play (such as user agent customizations on visual appearance of a page), or additional flexibility to customize the AX framework components is required (think of CSS styling to customize their visual appearance) that may introduce additional issues.

Therefore, to solve all issues the markup rendering process is divided in two different phases:

1. the actual rendering. In this phase the most appropriate markup is generated for each web component, only performing those quick checks that do not have a significant impact on the performance of the process;
2. The validation phase. In this process, which is started asynchronously after the rendering phase ends, the correctness of *hierarchical* relationships and other expensive accessibility validations can be performed. These include running the `axe-core` [Deq22a] automated accessibility testing engine to catch all issues in the generated markup that could not be caught otherwise.

While in some circumstances these optimizations can introduce problems (e.g. not being able to complete if the components are rendered multiple times in very short time intervals), heuristics have been put in place to recognize such scenarios and *unwind* them to make the entire rendering process blocking and synchronous.

Before concluding, two additional considerations on the internal architecture of the AX framework should be made. First off, due to the way it has been designed and implemented, it can be adopted *gradually* within existing process: in this way, developers can rewrite their code to leverage it by breaking down the process in various chunks. Secondly, all components in the AX framework do not take control of any aspect that is related to their visual appearance, unless that is strictly necessary to provide a basic working implementation of their business logic; developer can *freely* customize all aspects related to their look, while relying on the accessibility of the framework that prevents them from introducing accessibility issues in the process.

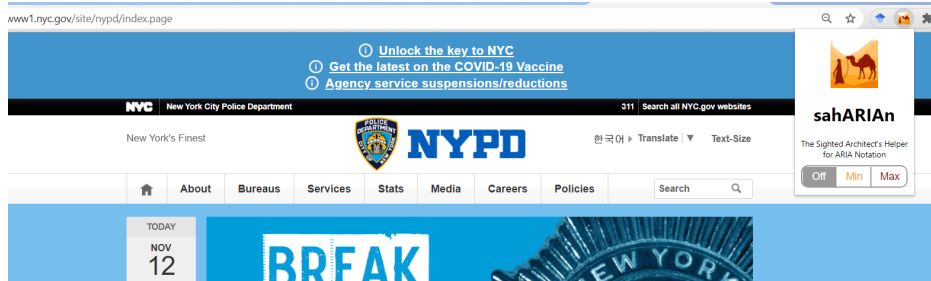


Figure 5.2: The first screen of the interface for the SAHARIAN browser extension

## 5.2 Implementing SAHARIAN

Part of the strategy to *make web accessibility more accessible* introduced in the research project documented in this PhD thesis involves an innovative accessibility testing and evaluation methodology, capable of letting developers *perceive* directly (i.e. experiencing in first person) accessibility issues and their impact on people with disabilities. As already explained in Section 4.3.1, this is achieving by *mapping* the accessibility of a web page on concepts that developers are already familiar with:

- the visual rendering, so that the *appearance* of the page reflects its accessibility;
- mouse operability, so that elements can be used with a mouse (or any similar touch screen device) *if and only if* the equivalent action can be performed by assistive technology users leveraging keyboard navigation.

To demonstrate the feasibility of this methodology, it has been implemented in *SAHARIAN*, a browser extension currently available for the Google Chrome browser. Figure 5.2 shows the initial screen that is shown immediately after activating the extension on any web page.

*SAHARIAN* can be activated on any web page rendered in the browser, independently of its origin server or any other of its characteristics (e.g. being protected by firewall rules or other access restrictions). The extension's

business logic is executed entirely client-side (locally in the browser): no cloud service or central infrastructure is involved in any phase of SAHARIAN's lifecycle. This implementation choice provides many significant advantages:

- it can be used safely on confidential web pages (e.g. for projects that are not publicly available yet or never will be);
- if specific certifications are necessary to integrate SAHARIAN in accessibility testing and evaluation processes, the absence of a central infrastructure can ease obtaining such certifications;
- it guarantees that obligations arising from various laws about testing web pages that may contain sensitive data (for instance, think of General Data Protection Regulation (GDPR) in the EU or other privacy laws) are fulfilled, as no data transfer happens;
- there is no way for SAHARIAN's developers, or any stakeholder involved with it, to have a clue on the information being processed (thus websites and applications being evaluated) by the extension.

Whenever SAHARIAN is activated, it achieves its goal by performing three tasks:

1. replacing the visual rendering of the page with a representation generated to reflect information conveyed by its DOM to assistive technologies;
2. intercepting all mouse events, handling them with a more complex logic to mimic their equivalent keyboard events and dispatching them to the original event target;
3. listening for DOM changes, so that it can update the generated visual representation of the page whenever its content changes (for whatever reason that caused the change to happen).

These steps are executed in order to implement (thus respecting the corresponding principles) the strategy discussed in Section 4.3.1. To provide an example of how SAHARIAN works, thus a practical example of the proposed accessibility testing and evaluation methodology in action, Fig. 5.3 shows the visual rendering of the New York City Police Department’s official website as shown in any web browser. Instead, Fig. 5.4 depicts its representation generated by SAHARIAN: the page is gradually despoiled of style and images in favor of a structure visualization functional to the accessibility analysis.



Figure 5.3: Homepage of the New York Police Department’s official website as shown in a web browser

In order to make SAHARIAN work in many scenarios out in the wild, several significant technical challenges have been faced. First off, web pages can nowadays be extremely dynamic: the actual DOM structure being executed at a certain point in time can be completely different from the one corresponding to the code sent by the server while replying to the initial page request. In addition to this, the DOM can be manipulated at any point in many unpredictable ways. To deal with this, SAHARIAN always reflects the information conveyed to assistive technologies by the DOM; its mutations are monitored as well, so that the representation can be updated whenever it changes. This is done by leveraging the *Mutation Observer API*, with a fallback timer mechanism for edge-cases in which it would not suffice or

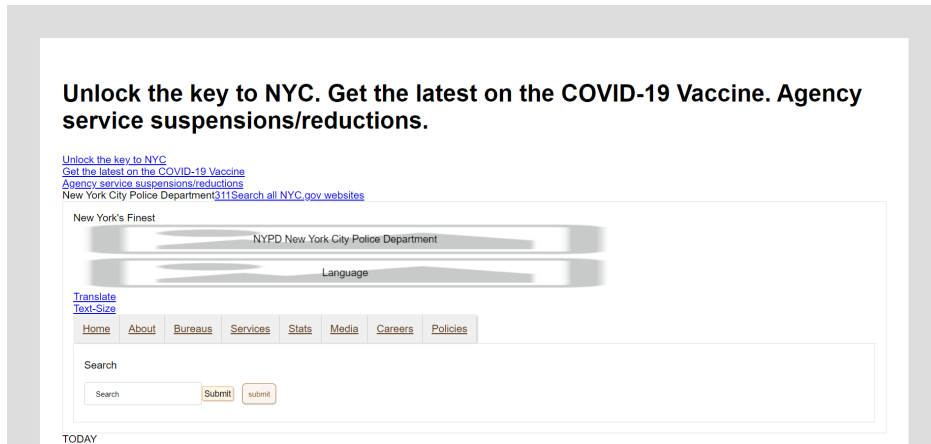


Figure 5.4: Visual representation of the Homepage of the New York Police Department's website generated by SAHARIAN

its implementation is unreliable (e.g. due to race conditions in concurrent changes).

Moreover, the page representation generated by SAHARIAN has been carefully crafted in order to fully implement the methodology illustrated in Section 4.3.1. Therefore, specific designs have been developed for each supported widget. These designs have been created keeping into account the fact that a widget could be implemented by leveraging a variety of HTML elements (specific HTML tags or generic ones whose semantics have been enriched leveraging attributes from the WAI-ARIA specification). They have been conceived to try to make the visual representation as similar as possible, regardless of the implementation method.

Finally, special attention has been given to handling mouse events. In fact, this is the key to implement the mapping of expected interactions by non-disabled developers to their equivalents used by assistive technology users via keyboard navigation as described in Section 4.3.1. When SAHARIAN is activated, it intercepts all mouse events and converts them into appropriate keyboard events deemed equivalent to the original one according to various criteria; this event is then dispatched to the initial target of the original event or (under some circumstances where it is most appropriate) to its



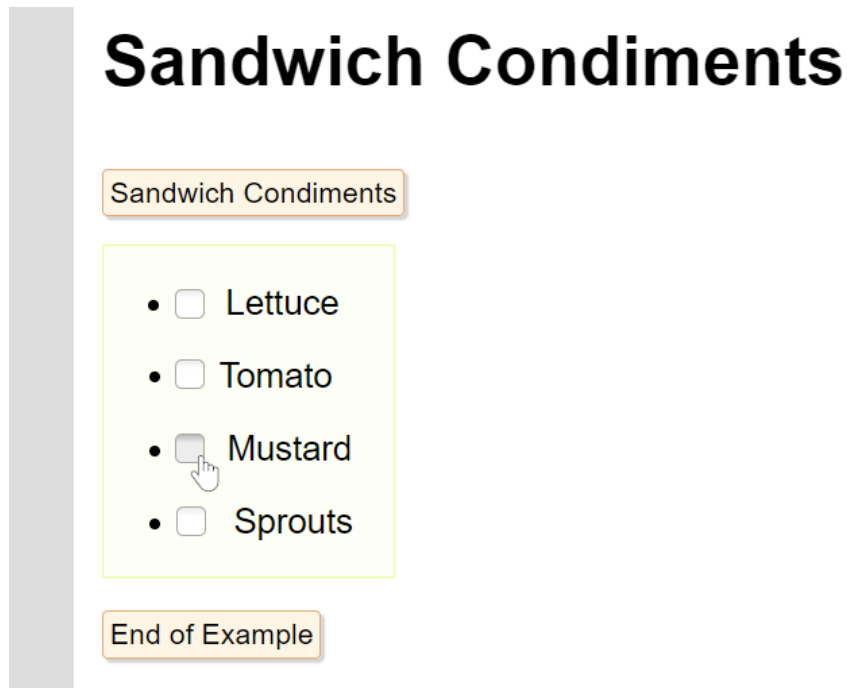


Figure 5.5: The visual representation generated by SAHARIAN for a checkbox implemented correctly

children or siblings. Given the complexity of the WAI-ARIA specification, deciding the event target follows rules that are not always trivial to keep into account the possibly complex hierarchical relationships (think to the concept of *owned elements*, which does not always match with the *parent-child* elements nesting).

To provide an example of such mappings, Figure 5.5 shows the visual rendering of a correctly implemented checkbox as rendered by SAHARIAN, while Fig. 5.6 shows how a checkbox that cannot be toggled via the keyboard is rendered by SAHARIAN.

While developing SAHARIAN, it became apparent that implementing some features could look trivial at a first glance, but as they say, often *the devil is in the details*. For instance, this challenge emerged clearly when dealing with images (determining their associated alternative text), or computing the accessible name and description of each element. While standards and

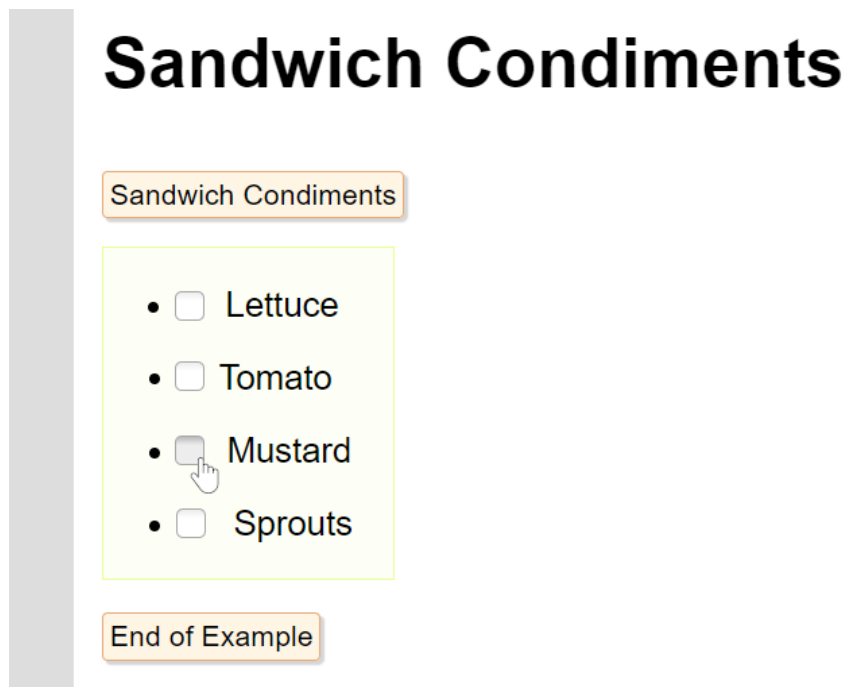


Figure 5.6: The visual representation generated by SAHARIAN for a checkbox which does not support keyboard navigation properly as it cannot be toggled

guidelines describe how they should be computed, and even provide an algorithm through pseudocode, different edge-cases are not covered; furthermore, practical experience showed that sometimes assistive technologies behave in slightly different ways than expected by reading and understanding the standard. In those cases, the decision on whether it was better to strictly follow the specifications or to behave as assistive technologies do has been made on a case-by-case basis, picking the alternative that has been deemed most appropriate for the ultimate goal of the SAHARIAN browser extension.

Finally, it is worth mentioning that in order to develop and test SAHARIAN, in these three years a repository of practical accessibility examples has been created: in fact, for each widget to be handled at least an example using native HTML elements, one with attributes from the WAI-ARIA specification, and other ones with different (and known in advance) accessibility errors were necessary. This dataset will be published in the future, hopefully acting as a reference for further work in this field.

## 5.3 Implementing A11A

As already illustrated in Section 2.3, many resources have been produced over the years to support designers, developers, content authors, managers and any stakeholders in the creation of accessible websites and applications. These come in different forms (written articles, videos, tools, etc.), and can satisfy the requirements of different user targets. Unfortunately, there is evidence that developers and other stakeholders have difficulties finding these resources: it is clear that you need to know what to look for, where to look it for, and then *select* the most suitable resources across a larger set of them. Such selection should be made keeping into account different criteria including the “authority level” of the information source, the relevancy of the resource to the project and its suitability to reach the established goal. Additionally, making conscious decisions about web accessibility often requires understanding and comparing “contrasting” and incomplete information to

get the “big picture” for the topic being considered.

Undoubtedly, this becomes part of the complexity of creating accessible content, increasing the “cost” (in terms of time, money, and cognitive efforts) required to create accessible websites and applications. Therefore, as part of the research project documented in this PhD thesis, it has been decided to create **A11A**: a structured, categorized repository of accessibility-related resources. It is argued that, other than creating innovative tools and methodologies, *making web accessibility more accessible* cannot be done without making it easier for stakeholders to find and *use effectively* the existing resources is essential. In this section some technical details about the implementation of **A11A** will be discussed, highlighting some specificities of the project arising from its characteristics.

In order to *collect* existing accessibility resources on the website, a hybrid approach in between a wiki and a blog has been chosen: indeed, the website contains a page for each accessibility resource to be published, including both an accompanying text written by the website creator<sup>1</sup> explaining its purpose and how to leverage it effectively, and useful links point to it. Each resource published on **A11A** has been personally tested and reviewed by the author of this thesis to assess its effectiveness, completeness and correctness wherever appropriate. All pages, though, contain features that can be commonly found on blogs; these include the ability for users to leave comments and discuss their content and share them on their social networks.

Figure 5.7 contains a screenshot that shows the Homepage of the **A11A** website. Apart from decorative elements (such as the top bar containing the website logo), it is made of six distinct components:

- the main menu, that contains all links to get to the various sections of the website;
- the search box, that allows to perform free text searches among all pages published on the website;

---

<sup>1</sup>All pages published on **A11A** until now have been authored by myself, Vincenzo Rubano; however, the website can be opened to external contributions in the future.

- the main content area, which occupies the largest portion of the screen as it shows the main content of the page;
- the sidebar, which shows complementary (and context-dependent) information to the content shown in the main content area;
- the footer, which contains additional links to useful pages of the website.

Being developed with a responsive, mobile-first design, the positioning of these components may vary depending on the device used to access the website and some of its properties (e.g. window size, screen resolution, etc.).

Speaking of the sidebar, its content actually varies depending on the type of page in order to provide only complementary information that is actually useful for a reader of that page. In general, it can contain various *cards* showing the list of the newest published resources (i.e. the last ones that have been put on the website in chronological order), the newest topics treated on the website, and several *calls to action* to invite users to engage more with the website and stay up to date with its content (e.g. by subscribing to its Really Simple Syndication (RSS) feed or following its social network pages). A special card that is worth mentioning is the one that shows *related content* to suggest users *similar* pages to the one they are reading on the website whenever appropriate; as this is based on various criteria (including the page content itself), this is believed to be useful in letting users discover additional accessibility resources related to the one they are reading about (for instance, think of getting as related content the guide to understand a specific standard while reading the page that introduce that standard and explain its purpose).

The variety of accessibility resources available on the Internet is reflected by the various sections of the website (which as explained before can be reached from the main menu). More specifically, these include:

***Frameworks and libraries.*** This section contains pages related to frameworks and libraries that can be leveraged to improve the accessibility of

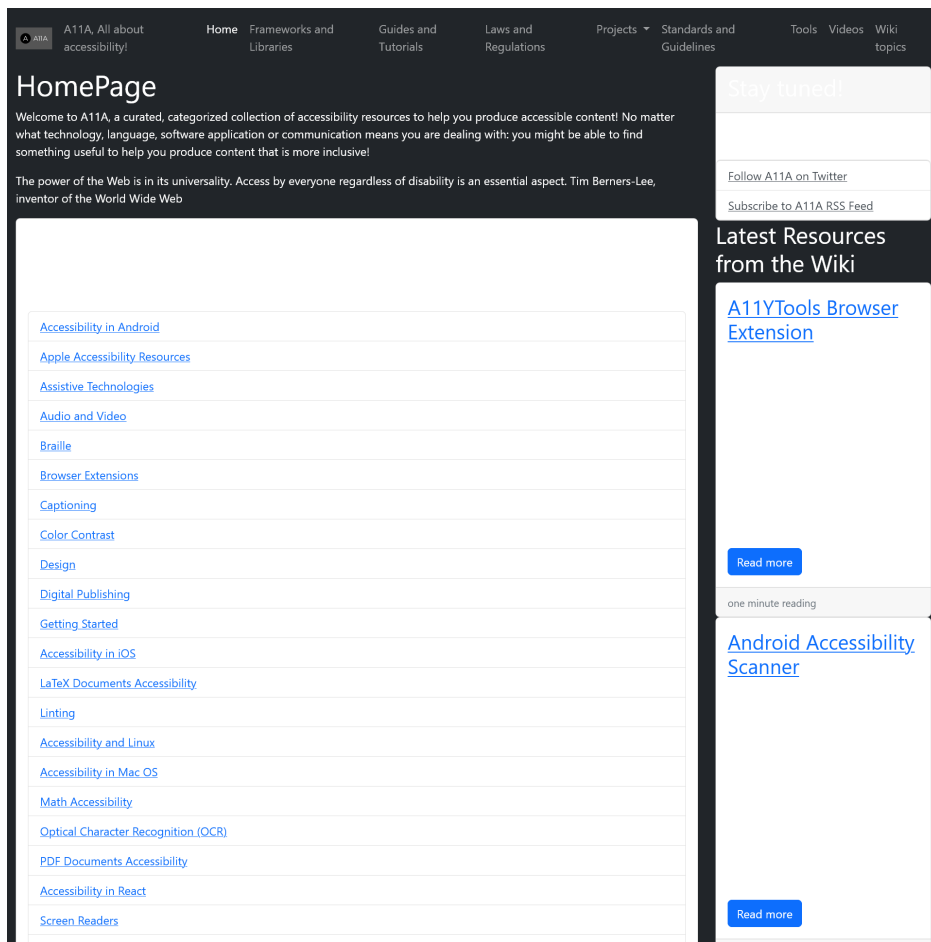


Figure 5.7: A screenshot of the A11A Homepage

projects written in many languages and using different digital technologies; these include both the ones developed specifically for accessibility-related purposes, and the ones who may facilitate the creation of accessible content *indirectly* (e.g. as they generate more accessible output when compared to their counterparts);

***Guides and tutorials.*** This section contains guides and tutorials that explain a very wide variety of topic, including resources on how to achieve specific goals (e.g. producing accessible Math equations), explaining how to use complex tools (e.g. user manuals for accessibility testing and evaluation tools), or provide support for understanding and leveraging other resources (such as guides illustrating how to *interpret* technical standards);

***Laws and regulations.*** In this section pointers to different laws and regulations enacted by various countries in the world to enhance digital accessibility are described. Wherever possible, the accompanying text tries to *summarize* the essential content of the described law and/or regulation, as well as its practical effects;

***Standards and guidelines.*** This section includes references to the more technical standards and guidelines (such as the ones that make web accessibility actually possible) that cover different digital technologies. It includes the ones that are *standards de facto* even if they cannot be technically considered as such (think to the Accessibility API reference of an operating system, for instance);

***Tools.*** Perhaps the most heterogeneous section, it includes all kind of tools to help stakeholders (designers, developers, content authors, managers, and more) to create accessible products leveraging different digital technologies. They range from browser extensions to command line tools, including web services and standalone apps with all the forms in between them;

**Videos.** As obvious as it seems, this section contains references to accessibility-related videos covering different topics; these include technical discussions, conference presentations, tutorials, webinars and more. Whenever possible, each page contains a player to let users watch the video without actually leaving A11A;

**Wiki topics.** This section contains all *topics* covered by at least one page published on A11A. Wiki topics can be thought as the typical *tags* that have been historically used in blogs to group related posts by common subjects they refer to. One can find topics to classify all resources published on A11A by a very heterogeneous set of criteria: examples include tags for iOS or Android accessibility, common frameworks used in web development projects (Vue.JS, React, Angular, etc.), specific subjects they relate to (Math, LaTeX, etc.), and more. In combination with the website sections, wiki topics should reflect the various criteria with which accessibility-related resources can be classified, organizing them to make for stakeholders as easy as possible to find what they could be interested in.

To actually implement A11A, it was decided to leverage a Static Site Generator (SSG); this decision arose after considering some advantages provided by leveraging such a tool to build a website:

- *low server maintenance costs*, as SSG output consists of static HTML pages that can be rendered by web browsers directly (i.e. no server side rendering is required);
- *very high performance and scalability*, due to the characteristics of SSGs output;
- *quick prototyping time*, allowing for very fast experimentation (for instance, you can dramatically change the structure of the website in minutes);



- security, for various reasons arising from the characteristics of SSGs; notably these involve the web server, as it must only *serve* static HTML pages, and content, as it can be easily backed up in multiple pages;
- *authoring flexibility*, as they allow a wide variety of formats and storage mechanisms to be used for site content; in case of A11A, it has been decided to leverage `markdown` [Gru12] to write its pages, and store all content (along with layouts and the code implementing the infrastructure to build the website) in a `git` [Spi12] repository.

Hence, A11A has been built as a JavaScript APIs and Markup (Jamstack) website by means of the `hugo` [GoH] static site generator. Hugo has been chosen over its many competitors due to its superb performance, onboarding experience<sup>2</sup>, and its extremely well written documentation. Server Side Rendering (SSR) techniques are leveraged only for pages that cannot be built by SSGs for obvious reasons; in case of A11A, such pages include the ones that handle form submissions.

Speaking of form submissions, a very particular infrastructure has been implemented exploiting to the maximum the template system provided by `hugo`; such infrastructure allows to easily *specify* the structure of forms by means of very simple Ain't Markup Language (YAML) [EBKdN17] files. During the site build process, these files can be used to:

- generate the markup that is necessary to render the form within the page, along with suitable client-side constraint validations to ensure the correctness of submitted data; and
- generate the code for a page that, by means of SSR techniques, handles the submission of the form by performing server side validation of the submitted data and executing custom tasks for each form (e.g. send emails, store data, perform searches, and so on).

---

<sup>2</sup>There are SSGs that require you to spend hours understanding how they are supposed to work before getting the first line of output.

Listing 5.1 shows the YAML code that is necessary to generate the contact form used on A11A as shown in Fig. 5.8.

Listing 5.1: YAML code from which the A11A contact form is generated

```
method: post
action: /contact
name: contact
fields:
  - name: name
    type: text
    label: Given name
    required: true
    pattern: "[\\pL]+"
    minLength: 3
  - name: lastName
    type: text
    label: Family name
    required: true
    minLength: 2
  - name: mail
    type: email
    label: EMail address
    required: true
  - name: subject
    type: text
    label: Subject
    required: true
    minLength: 5
    maxLength: 200
  - name: body
    type: textarea
    label: Message Text
```

The screenshot shows a contact form on a website. The form has the following fields: 'Given name:', 'Family name:', 'EMail address:', 'Subject:', and 'Message Text:'. Below the 'Message Text' field is a blue button labeled 'Send message'. The page has a dark theme. At the top, there is a navigation menu with links: 'Home', 'Frameworks and Libraries', 'Guides and Tutorials', 'Laws and Regulations', 'Projects', 'Standards and Guidelines', 'Tools', 'Videos', and 'Wiki topics'. The page title is 'Contact'. Below the form, there is a footer with 'Home', 'About A11A', 'Contact', and 'Legal'. Copyright information is provided: 'A11A is copyright © 2021-2023 by Vincenzo Rubano, some rights reserved. Except otherwise noted, content on this website is licensed under CC BY-NC-ND 4.0'. There is also a Creative Commons license logo.

Figure 5.8: A screenshot showing the A11A contact form

```
required: true
submitButtonText: Send message
successPath: /thanks/contact
```

It is argued that A11A's implementation is interesting from a software engineering point of view. Given that it has been decided to minimize the dependency on external cloud services (which is one of the main drawbacks of leveraging SSGs to build websites with dynamic features), for instance, the internal content search engine has been implemented by leveraging FTS-5 [Sql22], a technology aimed at enabling users to perform fast searches on large quantities of text data. Thus, a SQLite database is generated during the site build process, so that the search engine can be implemented just by executing appropriate queries on it according to user input and rendering the results. Another aspect that is worth noting is that A11A's graphical layout is

based on the `bootstrap` [TWB22] web framework; however, as maximizing the performance and reducing the page footprint were two wanted requirements, the `CSS` stylesheet is then processed by means of a pipeline based on `PostCSS` [Pos] and various plugins that *compile* the input stylesheet and remove unused code, optimize the output and minimize its size (e.g. by removing unnecessary whitespace).

## 5.4 Implementing *MARE* and ARIA-Service

When it comes to creating accessible websites and applications, the WAI-ARIA specification is a very powerful tool that can be leveraged by web developers to determine the semantics with which their content is exposed to assistive technologies. Unfortunately, as it is often the case, being this standard so powerful and flexible makes it complex to use it correctly. This is highlighted by one of its fundamental usage rules listed in the WAI-ARIA practices guide [Wor22a] document maintained by the W3C (that is part of the WAI-ARIA documents suite): “No WAI-ARIA is better than bad WAI-ARIA”, meaning that from an accessibility standpoint it is better to avoid leveraging this specification at all if people do not know exactly what they are doing.

Unfortunately, evidence shows that this complexity often results in actual implementations of this specification being wrong (i.e. using wrong attributes, or using them in inappropriate ways) or incomplete (i.e. not implementing the keyboard navigation interactions to fulfil assistive technology users expectations for each role); often developers do not even realize the critical side effects that such bad implementations can have on people with disabilities while using them. Therefore, as part of the process to *make web accessibility more accessible*, it has been decided to deal with this specific problem proposing two different innovative tools:

- the *Missing ARIA Role Explorer (MARE)*, an interactive tool that lets developers explore all the information they need to effectively leverage

WAI-ARIA roles and provide complete implementations;

- **ARIA-Service**, a JavaScript library that can be embedded in any web page to “automagically” provide all the necessary keyboard navigation support for all WAI-ARIA roles.

While this very specific problem was not part of the research project documented in this PhD thesis as initially conceived, it is argued that making technical standards more accessible (in the sense of reducing cognitive efforts required by stakeholders to effectively leverage them), and that the necessity of such tools clearly emerged during the development of the **AX** framework and the **SAHARIAN** browser extension (described in Section 5.1 and Section 5.2, respectively), it has been decided to tackle it. In this section various technical details about their implementation, as well as some challenges and how they have been overcome, will be discussed.

#### 5.4.1 Implementing *MARE*

Before introducing *MARE*, it is worth highlighting that, as explained in Section 2.1.3, many resources to help web developers effectively leverage the WAI-ARIA specification have been published over the years. Along with the technical standard itself, these include rules to decide *if and when* the specification should be used, support resources illustrating how to use it properly, and practical code examples explaining how the expected keyboard navigation interactions can be implemented. Unfortunately, this precious information is contained in documents that are very well written, yet due to their nature are lengthy and possibly not readily usable for practical problem-solving, or to act as a quick reference to the WAI-ARIA specification. Therefore, as part of the research project documented in this PhD thesis it has been decided to develop *MARE*, a tool specifically designed to fill this gap.

The main idea behind the design and implementation of *MARE* is that of making all information available about the WAI-ARIA specification from

official sources (i.e. the ones published by the W3C) available in a more convenient format, hopefully better suited as a practical tool for everyday usage in web development projects as a quick reference. Thought to be hosted on A11A as part of the website itself (more specifically, currently reachable from the “projects” section described in Section 5.3), MARE has been implemented as an interactive web application.

Implemented as a Single Page Application (SPA), when it loads MARE shows some introductory text to illustrate its purpose, as well as an accordion of collapsible “cards” in which each card corresponds to a different role of the WAI-ARIA specification. In contrast to the official documentation, though, there is a fundamental difference in the way in which MARE treat such roles from a conceptual point of view. As the WAI-ARIA specification is developed as a RDF ontology, WAI-ARIA roles are organized as a hierarchy in which a role can *extend* different roles (similarly to what happens with multiple inheritance when dealing with classes in source code). Additionally, the ontology contains *abstract* roles, which are not intended for direct usage by developers; instead, they have been introduced for architectural purposes (e.g. to define attributes that are common to multiple roles in a shared ancestor). It is argued that this representation can be simplified: while it is certainly elegant and functional to the specification authoring requirements, it may not be the most suitable information representation when it comes to practical problem-solving and the necessities of a quick reference tool.

Instead, MARE visualize WAI-ARIA roles by *flattening* such hierarchy; in particular:

- abstract roles are not shown at all;
- if a role *inherits* attributes from a role it extends, those attributes will be shown in the information about that role directly.

For each role, MARE displays a collapsible card; this enables users to *see* only the information they need, as they can decide which roles to expand (thus visualize the details about) and the ones to leave collapsed. For each

role, MARE contains the following information:

- a short description to illustrate the *semantics* of the role (for instance, the widget it represents);
- how the *accessible name* is computed (whether from its content, or by specific user input through appropriate attributes);
- if applicable, pointers to similar concepts in other frameworks (e.g. native HTML elements);
- specific information about the expected relationships (such as the expected role of the parent or owning element, or the role that child or owned elements must have);
- the complete list of WAI-ARIA attributes that can be used in combination with the role to *enhance* its semantics;
- the list of *required attributes*, the ones that *must* be specified to specify information that “complete” the semantics of the role;
- if appropriate, the expected keyboard navigation interactions that should be supported.

Gathering the necessary information made several challenges emerge. First off, the WAI-ARIA specification is available in two different formats: the technical standard (which is a traditional textual document), and a RDF ontology that should represent the same information contained in the standard. Unfortunately, this representation exhibit several limitations:

- in contrast to the technical standard (which is a normative resource), it is published as an informative resource: this means that W3C does not provide any guarantee either on the completeness or the correctness of the information it contains;
- it is indeed incomplete, as the technical standard provide much more information about roles than the one that is present in the ontology;

- it contains no information at all about WAI-ARIA attributes (e.g. expected value type and possible values).

Luckily, information on WAI-ARIA attributes can be obtained by combining the RDF ontology with another informative resource published by the W3C, in form of a XML Schema Definition (XSD): while it is aimed at a completely different scope (validating the correctness of WAI-ARIA attribute values), it contains precious information that is essential for MARE. Finally, additional important information (such as the expected keyboard navigation interactions expected for each role) is simply unavailable in any machine-readable format; therefore, it has been necessary to create and store such data.

As MARE is generated as part of building **A11A**, its build pipeline generates all the expected output from a single JSON file that contains all the necessary data; in turn, this file is generated by an external `python` script which combines the three data sources (the RDF roles ontology, the attributes XSD, and the information collected manually) in a format suitable for building MARE easily: hierarchical roles relationships are flattened (denormalizing the ontology), WAI-ARIA attribute IDs are normalized to match the ones used in the attributes XSD, and so on. The MARE build phase takes as input the JSON file in this format and outputs an HTML page (and some supporting JavaScript code) that actually constitute the tool itself. While this may seem a complex pipeline, it has been adopted as it allows to *minimize* the maintenance required to ensure that MARE works as expected and the efforts required to update it for reflecting changes in the specification or adding new features.

### 5.4.2 Implementing ARIA-Service

When the research project documented in this PhD thesis was conceived, the creation of **ARIA-Service** was not part of it. However, as time passed, experiences gained during the development process of the **AX** framework and the implementation of the proposed methodology to innovate the way in which



accessibility testing and evaluation is performed by means of the SAHARIAN browser extension, various personal considerations and discussions with other web developers made it apparent that it was a necessity. In fact, many web developers do not realize that leveraging WAI-ARIA attributes to enhance the semantics of their HTML code to make it more accessible is only part of the story; evidence shows that they have a hard time understanding basic WAI-ARIA usage rules (such as the one stating that each role is a promise, and it is the developer's responsibility to fulfill it).

Yet, the code that is necessary to fulfill this promise for each role is very complex: while support resources in the WAI-ARIA documents suite and additional educational materials provide a lot of practical code examples, it is argued that fully understanding the reasoning behind it and its necessity is beyond the average developer's understanding: other than the technical aspects, implementing all dynamic behaviors requires grasping knowledge such as how people with disabilities use assistive technologies, how assistive technologies work, how widgets on native platforms that correspond to WAI-ARIA roles behave, assistive technology user's expectations, and more.

Evidence and personal experience of this author shows that this is a problem even for developers of very popular web development frameworks and libraries: when they provide widgets that *must* be made accessible by leveraging the WAI-ARIA specification (either due to technical choices or because there is not a native equivalent HTML element), they do not implement such behaviors, or provide buggy and incomplete implementations to say the least.

Assuming that such behaviors for each WAI-ARIA role should mimic the user experience provided by their corresponding equivalent native widgets available in desktop and mobile operating systems, sometimes even code examples provided by very authoritative sources follow incorrect logic, do not cover edge cases (which are more frequent than one may think), are incomplete or contain critical bugs.

With all these considerations in mind, it has been decided to tackle this

problem with an innovative approach. Instead of focusing on new educational resources, as part of the research project the **ARIA-Service** has been developed. Essentially, it consists of a **JavaScript** library that, once loaded in a web page, provides the expected dynamic behaviors that WAI-ARIA roles should support: basically, if the markup of a page is correct, **ARIA-Service** enables all the features required by assistive technology users in terms of keyboard navigation. Of course achieving such goal introduced some challenges, which will be discussed in this section.

First off, it has been necessary to carefully examine the WAI-ARIA specification to determine the complete list of WAI-ARIA role dynamic behaviors to be implemented. Surprisingly, this analysis revealed that there were some roles requiring specific keyboard support, yet lacking in terms of practical code examples to implement it. However, by combining the experience of the author as an assistive technology user of different operating system, together with analogies from similar roles (for instance, links and buttons are very similar when it comes to keyboard navigation), this lack of resources has not been a problem. Other surprising findings from this analysis include the fact that in some cases the documented keyboard support is not complete, as there are WAI-ARIA roles that require more keyboard shortcuts than the ones documented in support resources.

One may argue that such a library is unnecessary if there are practical code examples that demonstrate how to achieve the desired goal. The problem, though, is that these code examples are not *generic enough* to be readily available for integration (i.e. as wrong as it is, by copy-and-paste) in development projects; instead, developers are expected to fully understand what the code is doing and implement themselves those features keeping into account the specificities of their project. **ARIA-Service**, though, originates from a simple, yet effective observation: whatever technology is used to build websites and applications, whatever library they leverage, at the end of the day this should result in the creation of a DOM within a browser. Supporting all available tools that allow the creation of websites and applications in a

single library is of course unfeasible; however, creating a library that provides support for all keyboard navigation commands necessary to implement the WAI-ARIA specification properly by relying only on the DOM is another story. It is a challenging process, but totally doable.

In particular, the design and implementation of **ARIA-Service** has been inspired by two fundamental principles:

- *load and forget about it.* The main idea is that, after embedding **ARIA-Service** in a web page, no additional efforts are required by the developer to *enable* keyboard support for all WAI-ARIA roles. Undoubtedly, this brings WAI-ARIA roles much closer to native HTML elements in terms of supporting web accessibility, with the additional benefits of being usable to implement widgets not available as HTML elements (for example, think of treeviews), to overcome styling limitations of native HTML elements, and being suitable to be used in custom existing implementations where code refactoring (to switch to native HTML elements) is not feasible;
- *minimize the overhead.* As this library has the potential to be loaded in many different projects, it has been decided to try to minimize its footprint as much as possible. This can also increase the likelihood of it being integrated in popular web development frameworks, should their developers be willing to do so.

Of course, a precondition for **ARIA-Service** to work properly is the *correctness* of WAI-ARIA attributes usage in a web page, and in general of its markup. The library does not provide any mechanisms to correct this kind of mistakes, and at the moment its behavior in these scenarios is undefined (and perhaps will remain as such even in the future). **ARIA-Service** provides the expected support for keyboard navigation *if and only if* the underlying WAI-ARIA markup is correct.

Anticipating the complexity of the code to be developed, it has been decided to implement **ARIA-Service** using **TypeScript**, a language built as a

superset of JavaScript to add support for a strong, statically analyzable type system. While `typescript` code cannot be directly executed within a browser, converting it to JavaScript can be automated by means of the `typescript` compiler<sup>3</sup>. While the presence of a build pipeline undoubtedly introduces a layer of indirection and complexity within web development projects, the choice of `typescript` does not impact significantly on the project complexity: indeed, a build pipeline to bundle and minimize the code is expected to be used by any modern JavaScript library, and is essential to achieving the goal of minimizing the footprint. Thus, type checking only becomes a step in a more complex pipeline that would be necessary even if using plain JavaScript code, with the additional confidence that statically typed code can guarantee.

Successfully implementing `ARIA-Service`, though, required overcoming three main challenges:

- determining how to *activate* DOM elements when keyboard commands equivalent to mouse clicks are performed, without requiring additional efforts by the developer adopting the library;
- storing the necessary state information to implement keyboard navigation for all roles;
- supporting the hierarchical relationships that the WAI-ARIA specification defines.

With regard to the first challenge, after a careful analysis of all keyboard commands to be implemented, it became apparent that overcoming it was a matter of *simulating mouse clicks*: in fact, when keyboard commands have to *trigger* actions on the elements with roles they are applied to, due to the way in which websites and applications are created the effect can be obtained by *performing a click* on it. Thinking again about it, that is exactly what

---

<sup>3</sup>Technically speaking, `typescript` is rather a transpiler than a true compiler. However, this differentiation is often discarded in its official documentation, thus in this PhD thesis the two terms will be used interchangeably when referring to it.

happens for mouse users: it is reasonable to expect that this support works reliably. Unfortunately, simulating mouse clicks is not easy as it may look like. While DOM node classes representing HTML elements inherit an intriguing `click()` method, unfortunately invoking it does not achieve the desired result: for security reasons, in fact, web browsers do not trigger the usual callback sequence they invoke when an actual mouse click occurs, therefore making the method pretty useless for ARIA-Service's purposes. To further complicate the situation, the exact sequence of callbacks invoked when actual mouse clicks occur (as well as the structure of the dispatched event objects passed to them) can vary across web browsers, and there are details which are not clearly documented. However, keeping in mind the original purposes for which ARIA-Service needs to emulate mouse clicks, a *reasonable* approximation might suffice. Indeed, such an approximation can be built by *constructing* the same event that a true mouse click would generate, and *dispatching* it appropriately. This is exactly what ARIA-Service does to simulate mouse clicks in a secure, reliable way.

In regard to the second challenge, after carefully examining the requirements behind ARIA-Service keyboard navigation implementations for all roles, it was determined that the state-storage issue could be overcome with some careful thinking. The first consideration is that, unlike traditional implementations on the Internet for WAI-ARIA keyboard navigation support, relying on the DOM as much as possible is a viable option, and reduce the necessity of using *state* in the first place. Second, additional state information is (or can be) actually stored in WAI-ARIA attributes applied to the elements themselves, therefore eliminating another category of information to be stored. Third, while this is not strictly necessary, some state information is *essential* to optimize the library if solely relying on the DOM without storing information in the code (e.g. using internal JavaScript classes or variables). Luckily, should the need for storing state arise and be unavoidable for any reason (working logic, performance, and so on), HTML provide a great mechanism that can be leveraged for this exact purpose: *custom data*

*attributes.*

Adopting such an approach for developing **ARIA-Service** allowed de facto the creation of *stateless* implementations for the keyboard navigation handlers necessary to support all WAI-ARIA roles, which in turn introduced three main benefits regarding:

***Performance.*** Being stateless, **ARIA-Service** handlers can be attached to the `document` object rather than to each element that could invoke them, therefore saving (potentially a lot of) memory and dramatically reducing the library initialization time.

***Simplicity.*** Being made of stateless handlers, **ARIA-Service** does not require a sophisticated, complex architecture. It is not even based on classes: all handlers, in fact, can easily be implemented in a functional fashion;

***synchronization.*** Being de facto implemented in a stateless fashion, **ARIA-Service** does not need either to *monitor* the DOM of a page for its changes, or provide a mechanism to make developers “report” significant DOM updates to the library implementation. Instead, the handlers’ implementations can rely on the fact that they always have the most accurate view of the DOM, as they basically construct it in a very efficient way each time they are invoked.

Finally, to make **ARIA-Service** work in all scenarios supported by the WAI-ARIA specification, the third challenge had to be overcome. In fact, unlike in **HTML** where hierarchical relationships among components are expressed by *nesting* them one inside the other (to form the DOM tree), according to the WAI-ARIA specification these relationships can be expressed by means of attributes as well (for instance, think of `aria-owns`) without any nesting requirement for the involved elements. Therefore, when querying the DOM for information, additional logic has been introduced to keep this into account. In other cases, though, these attributes can have more significant side

effects: think, for instance, to the `aria-activedescendant` case. While supporting keyboard navigation for elements leveraging this attribute requires a different logic from ones that do not use it and rely on proper hierarchical nesting of HTML elements, `ARIA-Service` can happily support both cases with no issues: due to various careful technical decisions in the way the code has been written, it does not even ship duplicate logic for handling the two cases differently.

## 5.5 Implementing A11YExamples

In order to pursue the goals of the research project documented in this PhD thesis, and more specifically to develop the proposed innovative methodology for accessibility testing and evaluation (implemented in the `SAHARIAN` browser extension) and the `ARIA-Service` JavaScript library, it was essential to have a repository of *accessibility examples* that could provide a wide set of *examples* of elements commonly available in websites and applications implemented in different ways, with different accessibility degrees, but *knowing* the accessibility issues affecting each example. Unfortunately, after a thorough research, such a repository to be exhaustive enough to cover all elements supported by the `SAHARIAN` browser extension and the `ARIA-Service` JavaScript library was not available. Therefore, as part of this research project, it has been essential to build such a repository called *A11YExamples*. In this section, the process to build it will be documented, along with the functional and non-functional requirements that should be satisfied, the challenges faced during its development and some opportunities for its future.

As already mentioned, building *A11YExamples* has been deemed necessary in order to successfully develop and implement both the `ARIA-Service` JavaScript library, and the proposed methodology to let developers test and evaluate the accessibility of their code by relying on concepts they are already familiar with. The functional and non-functional requirements that should be satisfied by this repository naturally arise from the *ratio* behind its

creation. While the two *driving factors* for the *A11YExamples* implementation may seem different, they indeed have a lot of characteristics in common. Therefore, it has been decided to develop a single artifact to satisfy both use cases. To describe such requirements it is better to start considering the ones arising by the fact that the *A11YExamples* repository has been created to support the development and implementation of the **ARIA-Service** JavaScript library.

As described in Section 4.4.2, **ARIA-Service** is a JavaScript library that can be leveraged by developers to provide the necessary support for keyboard interactions that is expected in cases where certain widgets are implemented leveraging the WAI-ARIA specification. While it is the developer's responsibility to implement these dynamic keyboard navigation behaviors, developing robust implementations to provide assistive technology users the same experience they would have if using its native widget counterparts is not trivial, can be prone to errors, and is often overlooked. In contrast, as illustrated in Section 5.4.2 **ARIA-Service** provides the ability to implement these behaviors by *embedding* the library in any web project. Building the code to implement all the expected behaviors contained in **ARIA-Service**, though, was a challenging task and could not be done without a repository of *possible widgets* that should be supported. Therefore, it has been decided to introduce in the *A11YExamples* repository at least one example for each widget that can be implemented leveraging the WAI-ARIA specification and requires custom JavaScript to be written in order to support keyboard navigation interactions.

Therefore, with this consideration in mind, it has been decided to create an example for each role included in the WAI-ARIA specification whose semantics represent a widget for which custom keyboard navigation interactions have to be implemented using JavaScript. A great source of inspiration for creating these examples have been very useful resources such as the APG [Wor22a] and similar educational resources, which in general provide practical code examples “ready to use” for implementing such behaviors;



however, it has always been necessary to edit such code before integrating those examples in the repository for different reasons:

- to fix bugs that were present in the provided code;
- to make the provided implementations *complete*, to closely match their native equivalents;
- to “modernize” the provided code by leveraging more recent JavaScript features and constructs to make it more maintainable;
- to remove unnecessary verbosity introduced in the code for educational purposes only;
- to simplify the provided code, to get so-called *Minimal Working Examples (MWE)*.


As it was necessary to compare WAI-ARIA widgets with their native HTML equivalents, another requirement used in the development of *A11YExamples* immediately arises: whenever an HTML element equivalent to an WAI-ARIA role is available, an example using that element should be created as well. This allowed to *immediately* compare both the native version and the WAI-ARIA widget, so that the user experience could be reproduced very closely. To expand the *A11YExamples* repository to contain other useful examples, whenever multiple techniques were available to implement the same widget (for instance, think of the necessary markup to implement a control using the *rowing tabindex* technique or the `aria-activedescendant` variant), an example for each technique has been created.

At this point, in which *A11YExamples* already contained the necessary examples to develop and implement the `ARIA-Service` JavaScript library, making it suitable for the development of the proposed innovative accessibility testing and evaluation methodology proposed in the research project documented in this PhD thesis only became a matter of *expanding* it. In

fact, all the examples could benefit the creation of the SAHARIAN browser extension as they were, and only adding more of them was necessary to make it more complete and robust.

The first addition to the repository was very straightforward, and naturally arose from the structure of the WAI-ARIA specification itself and the final purpose of the SAHARIAN browser extension. In fact, WAI-ARIA 1.1 contains some attributes (so-called *global attributes*) that can be applied to any HTML element independently of its role or native semantics; different examples to cover these attributes and their possible values have therefore been created. Furthermore, as the purpose of SAHARIAN is to map accessibility issues on concepts that developers are already familiar with, the second criteria for the expansion of the *A11YExamples* repository was to provide *variations* of all the examples created, in which *predictable and known in advance* accessibility issues were introduced. This allowed the creation of a very high number of examples to test the SAHARIAN browser extension and actually develop the *visual rendering techniques* used to represent widgets and reflect accessibility issues found in web pages. Similarly, introducing various examples in which *keyboard navigation* was broken in controlled and expected ways allowed the development of the complex system to map mouse operability (i.e. the operations typically used by non-disabled developers to test websites and applications with a mouse) to keyboard operability (the equivalent keyboard actions used in SAHARIAN and the methodology behind it).

Figure 5.9 and Fig. 5.10 shows various implementations of the same checkbox with different accessibility degrees (on the left), and the code fragments necessary to implement them (on the right). As it can be easily noticed, even if the code implementing the various checkboxes is very different, they all *look the same*: this is made possible by means of the stylesheet shown in Listing E.1 in Appendix E. Analogous stylesheets have been developed for all examples, as they inspired how the SAHARIAN browser extension renders each element when replacing the visual rendering of the page being evaluated.



## Checkbox examples

Here you can find different checkbox examples with various accessibility degrees, implemented both by using both native HTML Elements and the WAI-ARIA specification.

### HTML checkbox

*This checkbox implementation leverages native HTML elements.*

I am not a robot

**CORRECT**

```
<label>
<input type="checkbox">
I am not a robot
</label>
```

### Correct ARIA checkbox

*This is a correct checkbox implementation that uses attributes from the WAI-ARIA specification.*

I am not a robot

**CORRECT**

```
<span id="chk1" role="checkbox" aria-checked="false"
tabindex="0" class="checkbox">I am not a robot</span>
<script>
new class {
constructor(elementID) {
const element = document.getElementById(elementID)
element.addEventListener('blur', this.onBlur)
element.addEventListener('focus', this.onFocus)
element.addEventListener('click', this.onClick)
element.addEventListener('keydown', this.onKeyDown)
this.element = element
this.enabled = false
}

onBlur(event) {
event.target.classList.remove('focus')
}

onFocus(event) {
event.target.classList.add('focus')
}

onClick = event => {
if (!this.hasModifiers(event)) {
this.toggle()
}
}

onKeyDown = event => {
if (this.hasModifiers(event)) {
return // nothing to do
}
switch (event.key) {
case 'Spacebar':
case 'Enter':
this.toggle()
default:
return // nothing to do
}
event.stopPropagation()
event.preventDefault()
}

toggle() {
if (this.enabled) {
this.disable()
}
else {
this.enable()
}
this.element.classList.add('focus')
this.element.focus()
}

disable() {
this.element.classList.remove('checked')
this.enabled = false
this.element.setAttribute('aria-checked', 'false')
}

enable() {
this.element.classList.add('checked')
this.enabled = true
this.element.setAttribute('aria-checked', 'true')
}

hasModifiers(event) {
return event.altKey
|| event.ctrlKey
|| event.metaKey
|| event.shiftKey
}
}('chk1')
</script>
```

Figure 5.9: A screenshot showing the same accessible checkbox implemented with various HTML elements (on the left) and their code (on the right)

### Checkbox with no keyboard support

This is a checkbox implementation that does not provide the expected keyboard support. Notably:

- it is not keyboard focusable;
- pressing the *Spacebar* or the *Enter* key won't toggle its state when focused.

I am not a robot

**INCORRECT**

```
<span id="chk2" role="checkbox" aria-checked="false"
class="checkbox">I am not a robot</span>
<script>
new class {
constructor(elementID) {
const element = document.getElementById(elementID)
element.addEventListener('blur', this.onBlur)
element.addEventListener('focus', this.onFocus)
element.addEventListener('click', this.onClick)
this.element = element
this.enabled = false
}

onBlur(event) {
event.target.classList.remove('focus')
}

onFocus(event) {
event.target.classList.add('focus')
}

onClick = event => {
if (!this.hasModifiers(event)) {
this.toggle()
}
}

toggle() {
if (this.enabled) {
this.disable()
}
else {
this.enable()
}
this.element.classList.add('focus')
this.element.focus()
}

disable() {
this.element.classList.remove('checked')
this.enabled = false
this.element.setAttribute('aria-checked', 'false')
}

enable() {
this.element.classList.add('checked')
this.enabled = true
this.element.setAttribute('aria-checked', 'true')
}

hasModifiers(event) {
return event.altKey
|| event.ctrlKey
|| event.metaKey
|| event.shiftKey
}
}('chk2')
</script>
```

Figure 5.10: A screenshot showing the same checkbox with various accessibility issues implemented with different HTML elements (on the left) and their code (on the right)

More specifically, Fig. 5.9 and Fig. 5.10 show:

- a checkbox implemented using native HTML elements;
- a checkbox implemented using *generic span* HTML element, made accessible by leveraging the WAI-ARIA specification;
- a checkbox in which the WAI-ARIA attributes are used properly, but no keyboard navigation support is provided;
- a checkbox in which state updates are not conveyed properly (i.e. they are exposed using the `aria-selected` attribute instead of the appropriate `aria-checked` one);
- a checkbox implemented with a generic `span` HTML element in which no accessibility support is provided at all.

One may reasonably wonder the reason why it has been necessary to apply a stylesheet to make all these examples look the same. The answer is that, during the development of SAHARIAN, this made it easier to *immediately and clearly* know whether the extension was testing the right thing appropriately. In fact, by activating the extension on the page containing these examples, the issues they exhibit should become obvious.

Finally, it is worth noting the presence of the code generating each element which is shown side-by-side to the widget it generates. This feature has been implemented by means of a fragment of JavaScript code which is injected in each web page of the repository. Other than to simplify the development and testing of the SAHARIAN browser extension, this feature may also be useful in the future if *A11YExamples* is used as an educational resource on web accessibility; in fact, as it contains various possible examples, it could clearly illustrate best practices (accessible examples), bad practices (inaccessible examples), and various possible problems in the middle (i.e. examples with accessibility issues).

## 5.6 Implementing A11YVT and the *SCAMP* methodology

As discussed in Section 3.3, digital technologies have a huge potential when it comes to improving the accessibility of indoor environments and physical objects. This is a very wide topic, and therefore the most disparate implementations are possible; the second part of the research project documented in this PhD thesis focused on contributing to this field by:

- designing and implementing the prototype of A11YVT, a system to provide *accessible virtual tours* of indoor environments for blind and visually impaired people (for instance, by fully supporting their requirements to explore environments they are not familiar with) while providing all the features commonly expected by their sighted peers;
- developing and implementing a methodology to make physical, tangible scientific artifacts with a high historical value more accessible both for blind and visually impaired people and the general public.

In this section some interesting technical details about the implementation process of these contributions will be discussed, illustrating some challenges faced and how they have been dealt with thanks to a combination of careful design decisions and development strategies.

### 5.6.1 Implementing A11YVT

As discussed in Section 4.5.1, part of the research project documented in this PhD thesis involved providing an approach to make indoor environments more accessible by means of digital technologies. More specifically, the ability of exploiting *virtual tours* for this purpose has been explored. Consisting of simulations of existing locations composed of a sequence of videos, still images or 360-degree images and other multimedia elements such as sound effects, music, narration, text and floor maps, virtual tours can have a huge

potential to improve the accessibility of indoor environments for blind and visually impaired people, yet at the same time bring up many accessibility challenges. These considerations have been essential for the development of A11YVT, as it is a system to support the creation of virtual tours that are both accessible for blind and visually impaired people (providing the features they need, with a particular focus on the ones they need for exploring indoor environments they are not already familiar with) along with the ones that are commonly expected from such tools by their sighted peers. In this section some technical details on how a prototype of this system has been developed will be discussed.

Two essential requirements for the development of the A11YVT prototype influenced many implementation decisions:

- the system couldn't require any specific hardware (such as virtual reality headsets) or software to be installed on the user's device;
- taking tours through A11YVT would not require the user to download any software, or go through any complex configuration process.

Considering these requirements, the obvious decision that immediately came to mind was to leverage web technologies to implement A11YVT, as they easily satisfy the non-functional requirements described before while allowing the creation of a sophisticated system to satisfy the functional ones.

A11YVT is made of two main components:

- the backend, which is implemented in JavaScript on top of node.JS. It is responsible for handling the *building* data representation used to extract all the information needed for the tour when they are necessary; and
- the frontend, undoubtedly the most interesting component from an accessibility standpoint. It is responsible for actually *enabling* the user to take the tour: for blind and visually impaired people, it allows them to explore the indoor environments while relying on the features they

need to explore environments they are not familiar with; at the same time, it should provide an engaging experience for their sighted peers by providing all the features they commonly expect from such a tool. This effectively enables the creation of *accessible virtual tours* in a universal design fashion.

Other than the technologies usually involved in this kind of project (e.g. the Node Package Manager (NPM) [NPM22] for managing dependencies, HTML and CSS), noteworthy web technologies leveraged for implementing the A11YVT prototype include the JavaScript Object Notation (JSON) data format [Bra18] to store all information about indoor environments that is necessary for the creation of accessible virtual tours (i.e. content and the structure of the experience), and the ThreeJS [Mrd22] library, which is used to create an interactive 3D web environment based on the WebGL API to actually implement the tour.

More specifically, the A11YVT prototype consists of an HTML web page in which a `canvas` tag dynamically occupies the entire space of the window. Within the tag, a three-dimensional environment is rendered in real-time using the ThreeJS library and custom utilities. The environment consists of a camera placed inside a cube; each inner face of the cube is a fragment of the represented indoor environment, so that their combination make up a 360-degree image of it. The camera can rotate around a central point to which it is anchored, allowing the view to be changed using the mouse or a keyboard: this is the key to the *interactivity* of the tour, and one of the factors that allow both blind and visually impaired people and their sighted peers to get exactly the features they need. Figure 5.11 shows how a user can see the virtual tour of an indoor environment (on the left), and highlights the cubes and the related HTML code (on the right) to better illustrate how the described mechanism works.

In the model, points of interest are located in the positions corresponding to the ones that the corresponding numbers would occupy in a clock-face observed from the focus point. Each point can contain nothing (i.e. no



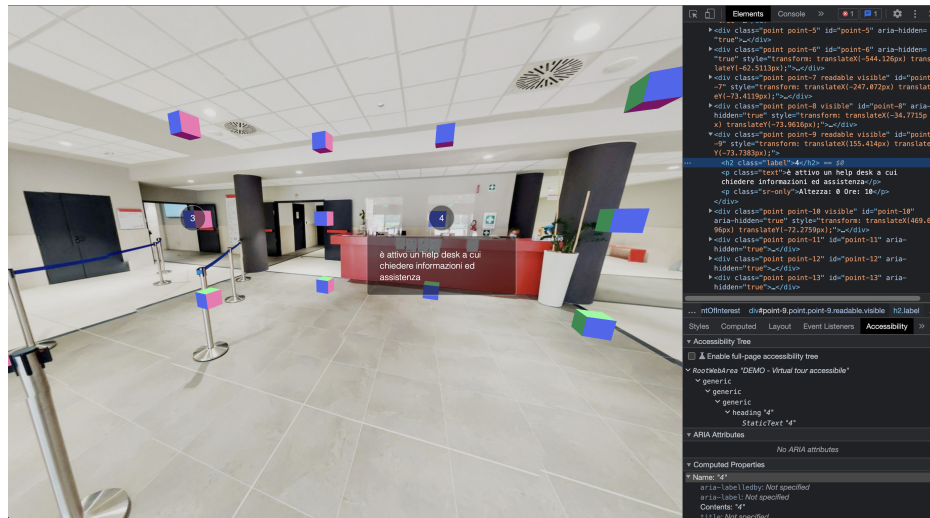


Figure 5.11: A screenshot capturing a virtual tour frame with two points of interest in A11YVT with cubes highlighted for clarity (on the left) and the related HTML code (on the right).

relevant information), specific information (with no assumptions on its type, as it depends on the environment being represented), or links to other points. In the example shown in Fig. 5.11, the focal point at 1 describes some details about the environment. By clicking on the arrow on the right, the camera moves to the cube at two o'clock. This focus point is not linked to any information, so nothing is shown on the screen. Clicking the right arrow again, the camera moves to frame the cube at three o'clock. As for the focus point 1, this focal point contains information that is displayed on the screen.

Consequently, each point is related to each other in the form of a circular container within the web page and designed to appear on a specific point of the 360-degrees image, so that it gets split in twelve parts (the maximum number of points in a clock). A trigonometric function is used to calculate the angle of the camera view of the displayed environment (and the size and ratio of the web page) in order to render the container's perspective at the required point (or to make it invisible if it is not displayed by the camera).

Given these characteristics, making virtual tours accessible for blind and

visually impaired people only becomes a matter of providing appropriate keyboard navigation making A11YVT automatically announce point changes (reading the associated information) to any screen reader. The user can press the right or left arrow to move right or left in the clock, respectively. When this happens, A11YVT makes the screen reader announce the landmark (by updating an WAI-ARIA live region) that the user could reach moving forward in the direction indicated by clock navigation. Pressing the up arrow will then *change* the focus point as if the user moved to that point in the real environment being represented. From that point, by pressing the down arrow, the user can go back to the previous landmark; otherwise, the whole process can be iterated to keep exploring the environment. Other keyboard shortcuts are provided to explore the environment along the third dimension (currently limited to three possible levels, as described in Section 4.5.1), to get back to the previous point in the clock direction, or to get an overview of all points (with their corresponding location in the clock navigation system) around the current focus point. It is argued that this process captures the essence of the way in which blind and visually impaired people explore indoor environments they are not already familiar with.

Virtual tours in A11YVT can be customized in various aspects, including:

- The number and position of information points;
- The contents of the information points and their types;
- The possibility of inserting two-dimensional or three-dimensional objects (deciding whether they occupy two or all three height layers);
- The movement speed of the camera, the Field of View (FOV), brightness and contrast of the environment.

The data model has been carefully crafted to support all these customizations.

### 5.6.2 Implementing the *SCAMP* methodology

As part of the research project documented in this PhD thesis, the SCAMP methodology to make scientific, manipulable artifacts with a high historical value *accessible again* both to blind and visually impaired people and the general public has been proposed. While this methodology heavily relies upon digital technologies to achieve its goal, it also puts a very strong focus on *preserving* the physical nature of such object and their interactivity. Such methodology consists in a theoretical and practical *modelling* process involving the usage of multimedia technologies to *enhance* the specificities of modelled objects. In this section, the application of the proposed process to the Brendel model of *Phaseolus vulgaris* Linn. shown in Fig. 4.4 (which as explained in Section 4.5.2 shows the early phases of the germination process of a bean plant) to demonstrate its feasibility will be discussed, illustrating the challenges faced in the process and the lessons learned to make it more scalable for future applications to other Brendel models and in general to other scientific, manipulable artifacts with a high historical value.

The first step in the proposed methodology is the *model recreation phase*, in which a digital 3D model of the physical object that the process is applied to needs to be created. As discussed in Section 4.5.2, this can be done either by means of a *3D-scanner* or *photogrammetry* based techniques. As a 3D scanning system is more expensive, requires highly specialized hardware and having specific knowledge to use it effectively, it has been decided to experiment with the latter approach. In general, to get a 3D-model of an object photogrammetry based techniques rely on specialized camera systems to acquire multiple *photographs* of the object from various angles, so that they can be combined through specialized software to *reconstruct* its three-dimensional representation. However, the properties of these systems can be obtained also by means of *less traditional* approaches that provide the noteworthy advantage of not requiring specialized hardware, thus having reduced costs. In this case, a high resolution camera system equipped on a smartphone (which was put on a tripod to capture stabilized images), has

been used. Several trial and error attempts were necessary to establish the most appropriate light conditions to take pictures in order to get the best possible results.

Unfortunately, evidence showed that a camera system equipped on a smartphone mounted on a tripod cannot yield the same results that a system specifically designed for photogrammetry would; it was clear there were several variables that could not be controlled easily during the images acquisition process. In particular, the most important one that influenced the final result was the *rotation angle* from each picture to the next one. More traditional approaches to photogrammetry require very strict rules to be followed in order to get high quality results out of it, but less traditional ones can only *get close enough* to following those rules due to technical limitations of the tools being leveraged. However, such limitations can be compensated by applying image post-processing techniques, and further refining the final result within a 3D modelling software. As this step was already necessary to apply several optimizations to get the best possible result from the 3D printing process (such as adding the necessary supports), this did not bring up any particular complications to the general process.

3D printing is a critical step in the proposed methodology to make scientific, manipulable artifacts with a high historical value accessible again, as it is responsible for actually *producing* the tangible artifacts that power the whole experience designed around them. Several adjustments to get the best possible 3D prints have been applied; among others, these include:

- choosing the most appropriate print resolution, to find a balance in between being high enough for *physical details* of the original object to be reflected, yet not *too costly in terms of time and money* to be unaffordable on a larger scale basis (the higher the print resolution is, the longer the process takes);
- picking the right materials, so that they are both suitable for 3D-printing while allowing the printed artifact to *produce* as much as possible the *tactile feelings* you would experience by touching the original

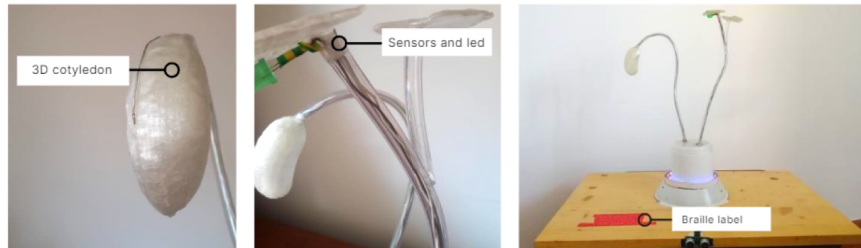


Figure 5.12: Artifact obtained by applying the model recreation phase to the Brendel model of *Phaseolus vulgaris* Linn.

object;

- ensuring the *accuracy* of the 3D-printer used, so that it could reflect *fine details* of the model being printed while minimizing the number of *printing artifacts* introduced in the process.

Special attention has been paid in reproducing tactile feelings; while sometimes fine-tuning the 3D printing process was enough to achieve the goal, reality showed that in some circumstances using more *handcrafting* techniques (such as adding details made by hand using different materials) might be necessary. The final artifact obtained by applying the model recreation phase to the Brendel model of *Phaseolus vulgaris* Linn. can be seen in Fig. 5.12.

After the model recreation phase, the proposed methodology to make scientific, manipulable objects accessible again involves the design of a multimodal user experience that *enhances* the values embodied by those artifacts and exploits the characteristics of the recreated model. In case of the Brendel model of *Phaseolus vulgaris* Linn. the complete *user experience* arose from the consideration that the original Brendel model is made by two different parts: the first one illustrates the cotyledons, while the second one the leaves that germinate a few days after planting the seed. The *cotyledons* play a significant role in the bean germination process, supplying the nutrition a plant embryo needs to germinate<sup>4</sup>.

<sup>4</sup>For further information on the process, see the definition of *cotyledon* available at

A short, one-minute video to illustrate the whole bean germination process has been created. While images show how the seed transforms into a plant, storytelling has been used to narrate the process; this information has been added to the video (by means of a voice-over audio track), along with some background music and audio effects to make it more interesting. In order for it to be fully accessible, though, the text of the voice-over track has been added as an optional captions track available on demand; it was not necessary to provide an audio-description, as in this case the voice-over already fulfilled all the needs of blind and visually impaired people.

In order to preserve the interactivity of the original bean germination model, the multimedia content *encourages* the user to *manipulate* and *touch* the artifact. Therefore, several *calls to actions* have been implemented. The first one *invites* the user to get closer to the model, suggesting that she/he can interact with it; this is achieved by:

***Led pulsing.*** Different light-emitting diode (LED) lamps have been installed, and they pulse faster as the user get closer to the artifact;

***Background sounds.*** Some speakers (that have not been put in the artifact, but close to it) play an audio track containing different pre-recorded nature sounds; sound volume gets louder as the user gets closer to the model.

At this point, the user is expected to *move* the artifact from its position, and tactile and visual cues indicate where it should be put. This ensures the user has had at least a chance to touch and manipulate the artifact; as different sensors have been installed both in the artifact and its surroundings, the system can also provide explicit feedback if the user does not understand what to do from the given cues.

When the user moves the artifact, video playback starts; whenever it comes to points illustrating characteristics represented by the model, it pauses and invites the user to touch and explore the specific part involved at that

---

<https://www.britannica.com/science/cotyledon-plant-anatomy>.

moment of the storytelling. Once again, this call to action is provided by means of audio (using the voice-over in the video), and visual cues: LED lamps indicating the parts to be touched gets turned on, and a *stop-motion* picture in the video shows that exact same point. Capacitive sensors have been installed in these *points of interest* as well; this allows the system to *understand* when the user has explored the indicated point. When this happens, LED lamps get turned off, and video playback resumes until the next call to action. This loop gets repeated until the video eventually finishes; the user is then invited to reposition the artifact where he found it initially, and accompanied in moving away from the installation by the same initial led pulsing and background music fading out.

In order to implement the described user experience, different capacitive sensors and a microcontroller have been installed within the artifact; while evaluating such products available on the market, it was clear that the requirement for their custom design must be kept into account during any application of the methodology. In case of the Brendel model of *Phaseolus vulgaris* Linn. on which the methodology has been experimented, for instance, it has been chosen to design custom-made capacitive sensors to avoid altering the visual appearance of the artifact and, most of all, leverage only materials that would be extraneous to the tactile feelings you would experience by touching the original object. In such a project it is clear that electronic components size matters as well; therefore, it has been decided to integrate an Arduino Nano, the smallest microcontroller available on the market for do it yourself (DIY) prototyping purposes. To make the experience possible, very small LED lamps and ultrasonic sensors used to measure the user distance from the artifact have also been added: as these sensors often have to be placed within the artifact obtained from the model recreation phase, it is fundamental to account for their presence while creating the 3D-model.

To apply the proposed methodology to the Brendel model of *Phaseolus vulgaris* Linn. an additional central processing unit was necessary; while

this requirement may probably arise in most applications of such methodology, its design and architecture have not been *strongly characterized* from a theoretical point of view as its implementation may be actually different on a case-by-case basis to better suit specific requirements. In case of the experience designed for the Brendel model of *Phaseolus vulgaris* Linn. its main responsibility is to receive the information transmitted by the micro-controller, and play multimedia content (both audio and video, by means of a screen and a speaker system connected to it). Once available, though, such a unit can be used to enrich the experiences created with the proposed methodology: in the described implementation, for instance, it is exploited in order to make the final experience much more immersive. Background nature sounds played at the beginning and the end of the experience were determined according to the local weather conditions: if it is sunny you can hear birds tweeting and the typical sounds of a quiet grass in a spring day, while if it is rainy you get to hear the typical sounds of an area within a storm; intermediate conditions were kept into account as well. Weather information was retrieved by means of additional sensors suitable for that specific purpose and connected to the central unit.

The software application that powers the experience was implemented on top of a client-server architecture organized around a message passing system (with communications happening via sockets on a local network, to achieve the lowest possible latency). Specific tricks (such as the logic to determine whether a distance measurement is accurate) were adopted to make various readings from sensors *stable and precise* (e.g. to avoid considering spurious readings), and make the system resilient to various types of errors.



# Chapter 6

## Evaluating the research project

Having discussed the theoretical foundations of the research project documented in this PhD thesis, as well as the purposes of the created artifacts and the relevant decisions behind their design and implementation, in this chapter the ways in which those artifacts have been evaluated will be discussed. All results obtained in the process will be explained in light of determining whether the created products (the **AX** framework, the **SAHARIAN** browser extension, the **A11A** website, the *methodology to make scientific, manipulable object with a high historical value accessible again*, **MARE**, the **ARIA-Service JavaScript** library, and the **A11YVT** for virtual tours) are effective towards the goal they have been created for, and can be used efficiently by their intended audience.

### 6.1 The evaluation process

As illustrated in Chapter 4, the research project can be divided in two distinct, yet related parts. In the first part, an integrated approach to *make web accessibility more accessible* is proposed; this approach is threefold, as it aims to achieve three goals:

- Making the development of accessible websites and applications more accessible, by providing the **AX** framework described in Section 5.1; be-

ing implemented as a declarative framework of web components, its main purpose is to provide a mechanism in which *accessible markup* is generated by default with no significant efforts required by web developers.

- Making accessibility testing and evaluation more accessible. This is achieved by proposing a *methodology* that relies on concepts that web developers are already familiar with to *communicate* them accessibility issues in their code, and letting them *directly perceive* their impact on people with disabilities. This methodology has been implemented in the SAHARIAN browser extension described in Section 5.2.
- Making accessibility-related resources more accessible. This is achieved by creating the A11A website described in Section 5.3: a categorized, structured repository of accessibility-related resources of various types and covering a wide variety of topics. The main purpose of A11A is to make these resources more easily discoverable and easier to use by letting developers easily understand in which context they should be leveraged, and grasp the *big picture* behind them without having to perform difficult, long-lasting thorough researches.

Having these goals clear in mind has been fundamental in determining how the artifacts proposed as part of the research project documented by this PhD thesis should be evaluated. In particular, analyzing their target and the *ratio* behind them, has been dramatically useful in determining some metrics that could be leveraged to assess whether they *effectively* satisfy the goals for which they have been specifically developed.

First off, all the artifacts produced as part of the research project documented in this PhD thesis have been considered for user evaluation. As clearly illustrated in Section 4.1, this project can be divided in two distinct, yet related parts. In the first one, different artifacts to *make web accessibility more accessible* have been produced; these include the AX framework (a declarative framework that generates accessible markup for elements com-

monly found in websites and applications, thus allowing the development of websites and applications that are *accessible by default* without additional efforts by the developer), the SAHARIAN browser extension (which implements the proposed innovative accessibility testing and evaluation methodology that maps accessibility-related concepts on concepts that developers are already familiar with, thus letting them *directly* perceive accessibility issues in their code and their impact on people with disabilities), and the a11a repository (which consists of a website to collect and classify accessibility-related resources to let stakeholders finding and using them more easily). Instead, the second part focused on applying digital technologies to make indoor environments and scientific, manipulable artifacts with a high historical value more accessible by means of the A11YVT and the SCAMP methodology, respectively; prototype implementations for both to demonstrate their feasibility have been developed as well.

Given the nature, the characteristics and different intended audience of all the produced artifacts, it was clear that a specific test for each of them was necessary; in deed, getting useful insights on all these artifacts required evaluating different metrics, and consequently the design of a specific test for each of them has been deemed necessary. Speaking of metrics, though, a common pattern clearly emerged from the early considerations behind the testing phase and influenced the structure of the evaluation process: artifacts produced *to make web accessibility more accessible* (the AX framework, the SAHARIAN browser extension, the a11a website, the ARIA-Service JavaScript library, and MARE) can be evaluated both from an objective (i.e. in terms of their effectiveness) and subjective (i.e. in terms of user's perceptions about their usage) point of view; instead, evaluating the prototypes implemented to prove the feasibility of A11YVT and the SCAMP methodology from an objective point of view was not possible. Therefore, the first have been evaluated leveraging both objective and subjective metrics; the latter, though, have been evaluated only from a subjective point of view. The repository of accessibility examples described in Section 5.5,

which played a significant role in developing both the SAHARIAN browser extension (and the accessibility testing and evaluation methodology behind it) and ARIA-Service has been intentionally excluded from a thorough testing and evaluation process, as it has not been deemed worthwhile assessing its (in)completeness and effectiveness: in deed, it has been developed only as an artifact *functional* to others, and it has not been thought as a user-facing product since its early design.

Analyzing the purposes of all artifacts for which objective metrics can be evaluated, as well as their nature and characteristics, an interesting pattern can be highlighted; in deed, each artifact can be truly evaluated by considering two different scenarios: one in which only the current *status quo* is considered, and the other in which the evaluated artifact is involved. By evaluating the same metrics in both scenarios and comparing the obtained results, provided that preliminary testing conditions are met (such as involving the same user target segment in the test and comparing the creation of the same website or application both with and without the artifact under evaluation), the usability testing process can also provide quantitative and qualitative measurements to assess whether the artifacts allow to effectively *improve* the current situation with web accessibility by enabling developers to produce content that is more accessible. Applying the same consideration to subjective metrics as well, and therefore *adding* user perceptions to the equation, this comparative analysis made it possible to demonstrate whether the proposed artifacts (thus the approach behind the research project documented in this PhD thesis) can effectively provide a strategy and the necessary tools to *make web accessibility more accessible*.

Another important consideration about the evaluation of the proposed tools to *make web accessibility more accessible* needs to be made, as it influenced the whole evaluation process. Given the extensive set of features of the proposed tools, as well as their very broad support for many different usage scenarios and wide variety of requirements (possibly by different stakeholders) that they satisfy (for instance, think of all *elements* supported

by the SAHARIAN browser extension, or the components provided by the AX framework), providing full coverage through usability tests was unfeasible. Therefore, it has been decided to employ a *task based approach*: instead of evaluating the artifacts on all possible tasks in which they could be leveraged, a *relevant* one has been selected in order to perform usability testing based on that task. Selecting a *significant* task for each tool among the set of all possible ones required establishing different *selection criteria*. Such selection criteria have been chosen while keeping into account different factors, including:

***The relevancy of the task.*** The chosen task had to be representative both from the point of view of the features provided by the evaluated artifact (i.e. allowing the artifact to *express* its potential) and the possible usage scenarios in which the artifact could be leveraged.

***The time necessary to complete the task.*** It is clear that the proposed tools can be leveraged to complete task which may have a different level of complexity, thus requiring different times to be completed. It has been decided to choose tasks that were *complex enough* to effectively evaluate each artifact, yet not taking too long to be completed to guarantee the feasibility of the test.

Additionally, it should be noted that obtaining qualitative and quantitative measurements for objective metrics for each tool required establishing a specific approach on a case-by-case basis, as the AX framework, the SAHARIAN browser extension, the a11a website, the ARIA-Service JavaScript library and the MARE interactive tool are very different from each one in terms of many characteristics (type of tool, intended purpose, usage scenario, and so on). In contrast, it has been decided to evaluate subjective metrics by relying on *user perceptions* about the task: more specifically, it has been decided to focus on measuring the perceived complexity of the task both without the evaluated tool (the current scenario) and with it (the proposed scenario). Such task complexity evaluation can be done by means of a questionnaire

administered at the end of each test; other than the usual psychometric properties of being reliable, sensitive and valid, though, it was desired such questionnaires to be:

- short, in order to reduce the time required to fill them to the minimum;
- easy to respond to, in order to minimize the complexity of the answering process for users;
- easy to administer, in order to facilitate the evaluation process;
- easy to score, in order to make the data analysis process as easy as possible.

This kind of questionnaires, as well as their impact on the results of usability evaluation tests and the most suitable scenarios in which they should be administered, have been widely discussed in the literature; different alternatives have been considered for usage in the evaluation process of the artifacts produced as part of the research project documented in this PhD thesis, including:

- the *After-Scenario Questionnaire (ASQ)* [Lew91] proposed by Lewis, which can be used to quantify a rating of the ease of a task, the amount of time the task took to complete, and the level of support received by the user throughout the process;
- the *NASA-TLX* (NASA's Task Load Index) [HS88], which is aimed at measuring the mental effort required to perform a task;
- the *System Usability Scale (SUS)* [Bro96] proposed by Brooke, which is aimed at measuring the perceived usability of a system (be it a website, an application, or any other digital product);
- the *Usability Metric for User Experience (UMUX)* [Fin10] by Finstad and its "lite" variant proposed by Lewis et al. [LUM13], aimed at being a short qualitative assessment designed to measure the general usability of a system.

After carefully evaluating various UX questionnaires proposed in the literature, it has been decided to leverage the UMUX one as it respects the four fundamental properties desired for the evaluation questionnaire outlined before in this section and allows to evaluate each artifact by measuring its usability assessing effectiveness (i.e. the accuracy and completeness with which specified users can achieve specified goals in particular environments), efficiency (i.e. the resources expended in relation to the accuracy and completeness of goals achieved) and user's satisfaction (i.e. the comfort and acceptability of the work system to its users and other people affected by its use): while reflecting three characteristics of the current situation with web accessibility that the research project documented in this PhD thesis is willing to improve, these three elements constitute the main components of the definition of usability provided by the *ISO 9241* [ISO18] standard.

As mentioned before, though, it has been decided to leverage independent techniques to evaluate the usability of the artifacts proposed in the second part of the research project documented in this PhD thesis. This decision arose from the consideration that they (the prototype implementation of the SCAMP methodology to the Brendel Model of *Phaseolus vulgaris* Linn. and the A11YVT system that allows the creation of accessible virtual tours of indoor environments) have very specific characteristics, and therefore required an evaluation process crafted with these considerations in mind. However, their evaluation process relied on a questionnaire administered to users after performing *relevant* tasks involving the artifacts, so that they could *experience* them before providing their opinions. Such questionnaires have been designed to focus more on user perception (thus subjective metrics) rather than quantitative, objective measurements of their usability: the former have been deemed more relevant to evaluate these artifacts in the actual stage of their development process.

To conclude the analysis of the premises on which the whole evaluation process of the artifacts produced by the research project documented in this PhD thesis has been based upon, it is worth mentioning two additional fac-

tors. First off, as obvious for all usability tests, a protocol has been established to evaluate each artifact; while each artifact required a specific testing protocol, all of them had to include at least:

***A planning phase.*** In this fundamental phase critical decisions on the test had to be made; these included how the artifact could be effectively evaluated, what to test for, how to conduct the test, which metrics could be leveraged to evaluate the artifact, and how they could be measured. Functional and non-functional requirements that had to be satisfied by each artifact played a significant role in this phase, as well as its high-level goals and the reasons behind its creation. The representative task(s) to be performed by users as part of the actual test had to be established in this phase as well.

***A recruiting phase.*** Due to their nature and characteristics, the artifacts to be evaluated are targeted towards different audiences. Therefore, it was necessary to conduct a careful recruiting process to ensure that test participants could provide valid insights on the actual “usefulness” and usability of each artifact.

***Establishing the execution plan.*** Even if every attempt to minimize the complexity of the tasks to be performed by users during the testing phase have been made, due to their nature some of them still remain complex for the intended audiences of the proposed artifacts to be evaluated: such complexity cannot be reduced under a certain threshold, as evaluating the artifact actually encompasses *measuring* such perceived complexity, as well as determining if it can be reduced (and in case of a positive answer how much) by involving the evaluated artifact in the task. Therefore, it was necessary to pay much attention to the *operative planning*, establishing very practical details such as the way in which the test had to be performed, its timing, the way in which questionnaires had to be administered, and so on.

Secondly, to be sure to collect all possible feedback from users actually



performing the test, it has been decided to include in the questionnaire administered immediately after each test a free-form question in which users could actually provide their observations on anything related to the artifact, its testing and evaluation, and the context in which it should be used. While being extremely easy to implement with minimal efforts, It is argued that this very simple idea could provide useful insights that could not be anticipated in the test planning phase.

## 6.2 Evaluating the AX framework, SAHARIAN and A11A

As outlined in Section 4.3, the first part of the research project documented in this PhD thesis consists of proposing an innovative approach to *make web accessibility more accessible*. In particular, this goal is achieved with a threefold approach:

- providing developers with a tool capable of enforcing the generation of accessible markup (the AX framework);
- providing an innovative methodology for manual accessibility testing and evaluation, in which accessibility issues are mapped to visual appearance and mouse operability, two concepts that developers are already familiar with (implemented in the SAHARIAN browser extension);
- providing a structured, categorized repository of accessibility-related resources to make it easier for stakeholders to find what they need, when they need it, even if they do not know what to look for, how and where (the A11A website).

While these tools can be naturally combined to provide an integrated approach covering many accessibility-related necessities which may arise during the development process of any website or application, these tools have been designed to be used independently one from each other; therefore, for the

sake of obtaining more relevant insights, each one has been evaluated in a different test session. In this section the evaluation process for these three artifacts will be described, discussing the obtained results.

### 6.2.1 Evaluating the AX framework

As illustrated in Section 4.3.2, as part of the proposed approach to *make web accessibility more accessible*, the AX declarative framework has been developed; consisting of a library of *web components*, the AX framework can *enforce* the generation of accessible markup by:

- ensuring that all information necessary to the generation of accessible markup is provided;
- ensuring the *correctness* of hierarchical relationships among the components nested within a page;
- ensuring that the markup generated to implement each component is correct (accessible).

In order to perform the evaluation process of the AX framework, both objective and subjective metrics have been leveraged as explained in Section 6.1. Fifteen different users have been recruited for the test; of these, eight of them classified themselves as “male”, seven of them as being “female”, and nobody in other possible gender classifications or not wanting to communicate their gender at all. All of them had an age in the twenty five-fifty years old range. All participants declared themselves as having an intermediate level of knowledge about web development, but only basic skills and limited experience with regard to web accessibility.

The test session took place in a room prepared for the occasion; all users performed the test at the same time by means of their laptops, having access to the Internet and whatever resource they see fit for the purpose of completing the proposed tasks. Special attention and various technical arrangements have been adopted to ensure that participants could not communicate during

the session to share insights on how to complete the tasks. Users have been asked to complete two different tasks:

1. Provide a minimal working example of an HTML page containing a widget to choose one out of five grades to score a film with; they could only use HTML, CSS, and custom JavaScript code.
2. Provide a minimal working example of an HTML page containing a widget to choose one out of five doneness levels to cook a steak; in this case, they could use HTML, CSS, and custom JavaScript code, but the components provided by the AX framework had to be leveraged.

Before the beginning of the test session, participants have been informed on how the session would take place, illustrating the tasks they were expected to complete and how their feedback would have been collected after the end of each task. It was stressed enough the fact that the visual appearance was not an element for validation, and therefore investing time in that area was strongly discouraged. Immediately after the completion of each task, the UMUX questionnaire was administered along with an additional free-form question in which participants could provide whatever feedback they deemed relevant. During the test process different objective metrics have been measured:

**The *task completion rate*.** Only 80% of users were able to complete the first task; all three participants which did not complete it declared in the administered questionnaire that they did not understand how to make sure that their code was accessible. In contrast, 100% of participants were able to complete the second task.

**The *task completion time*.** On average, users took 32 minutes to complete the first task; observations in the process revealed that most of the time was spent navigating the Internet looking for information how to create an accessible radio group using HTML. In contrast, on average participants took only 18 minutes to complete the second task; even if

they had to learn how to use the AX framework, data show a 43.75% task completion time reduction on average.

**The *accessibility of the produced artifacts*.** At the end of the first task, only four participants were able to produce an artifact conforming to the given specification that did not exhibit any accessibility issue. The remaining ones who completed the task produced an artifact conforming to the given specification, but it contained at least an accessibility issue (on average, though, more than one of them was present). Such issues were related to the HTML markup, as well as its visual appearance (especially to color contrast and font sizing).

**The *UMUX scores*.** The standard UMUX questionnaire was administered immediately after the end of each task. The two tasks reported pretty different scores: in fact, if the first one only got a 35 score on average, the second (involving the AX framework) got an impressive 91 score on average. The most significant difference among answers in the two questionnaires was for question  $q_2$ : users found the creation of accessible markup by means of the AX framework significantly less frustrating than native HTML. Other questions confirmed the same trend even if accounting for minor differences in the data comparison.

Analyzing the obtained data, it can be safely concluded that the AX framework provides a far superior user experience when it comes to creating accessible markup; combining this with the consideration that *by definition* the markup generated by the framework is completely accessible, it can be concluded that it contributes to *making web accessibility more accessible* by enabling developers to build applications and websites that are accessible by default with significantly reduced efforts when compared with the more traditional approach.

### 6.2.2 Evaluating SAHARIAN

As illustrated in Section 4.3.1, as part of the proposed approach to *make web accessibility more accessible*, an innovative methodology to perform manual accessibility testing and evaluation; key to the methodology is the fact that, unlike more traditional ones, it *maps* accessibility issues on concepts that developers are already familiar with. In this way, the visual rendering of the page and its mouse operability *reflect* can be made to *reflect* the accessibility-related information conveyed by that page to assistive technologies and the experience that a user could get if using keyboard navigation, respectively. As described in Section 5.2, this methodology has been implemented in the SAHARIAN browser extension available for Google Chrome.

In order to perform the evaluation process of the SAHARIAN browser extension, both objective and subjective metrics have been leveraged as explained in Section 6.1. Sixteen different users have been recruited for the test; of these, eight of them classified themselves as “male”, eight of them as being “female”, and nobody in other possible gender classifications or not wanting to communicate their gender at all. All of them had an age in the twenty five-fifty years old range. All participants declared themselves as having an intermediate level of knowledge about web development, but only basic skills and limited experience about web accessibility. However, they had basic familiarity with accessibility testing tools, and knew how to use at least one of them.

The test session took place in a room prepared for the occasion; all users performed the test at the same time by means of their laptops, having access to the Internet and whatever resource they would see fit for the purpose of completing the proposed tasks. Special attention and various technical arrangements have been adopted to ensure that participants could not communicate during the session to share insights on how to complete the tasks.

An example web page, with known accessibility issues, has been set up for the testing process. In particular, it is based on the popular *Accessibil-*

*ity University Demo Site*<sup>1</sup> made available by the University of Washington, which has been processed to:

- remove all features currently not supported by SAHARIAN, in particular those related to video playback;
- obtain two separate, orthogonal versions of the page to divide the time required to identify all accessibility issues present in the page.

Let `inaccessible.html` and `accessible.html` be the inaccessible version of the *Accessible University Demo Site* and its accessible counterpart, respectively. Two pages. Let `a.html` and `b.html` be the two pages actually used in the testing process. After performing the described processing, `a.html` contains inaccessible versions of structure and form elements, and accessible versions of carousel and paragraphs. Vice-versa, `b.html` has inaccessible versions of carousel and paragraphs, and accessible versions of structure and form elements. Testers of type *AB* received `a.html` first, and were asked to identify its accessibility issues in the way they would see fit (even using an accessibility evaluation tool they were familiar with). They then received `b.html` and were asked to complete the same task leveraging SAHARIAN. The contrary is true for testers of type *BA*: they received `b.html` first, and were asked to identify its accessibility issues in the way they would see fit (even using an accessibility evaluation tool they were familiar with). They then received `a.html` and were asked to complete the same task leveraging SAHARIAN. Therefore, the problems identified by testers of type *AB* on `a.html` are orthogonal to issues identified by testers of type *BA* on `b.html` without SAHARIAN, and problems identified by testers of type *AB* on `b.html` are orthogonal to issues identified by testers of type *BA* on `a.html` with SAHARIAN.

After splitting randomly the participants in two groups with eight members each, they were assigned to the type *AB* or *BA* and asked to evaluate

---

<sup>1</sup>The inaccessible version of the page is available at <https://www.washington.edu/accesscomputing/AU/before.html>, whereas its accessible counterpart is available at <https://www.washington.edu/accesscomputing/AU/after.html>.

the accessibility of the two pages with and without leveraging the SAHARIAN browser extension, respectively. Before the beginning of the test session, participants have been informed on how the session would take place, explaining the tasks they were expected to complete and how their feedback would have been collected after the end of each task. After the completion of each task, the UMUX questionnaire was administered along with an additional free-form question in which participants could provide whatever feedback they deemed relevant. During the test process different objective metrics have been measured:

**The *task completion rate*.** 100% of users were able to complete both proposed tasks.

**The *task completion time*.** On average, users took 21 minutes to complete the first task; observations in the process revealed that most of the time was spent leveraging the accessibility testing and evaluation tool interface. In contrast, on average participants took only 16 minutes to complete the second task; data show a 23.81% task completion time reduction on average.

**The *identified accessibility issues*.** In all cases, participants were able to successfully identify all accessibility issues present in the proposed pages.

**The *UMUX scores*.** The standard UMUX questionnaire was administered immediately after the end of each task. The two tasks reported pretty different scores: in fact, if the first one only got a 22.6 score on average, the second (involving the SAHARIAN browser extension) got an impressive 87.3 score on average. Similarly to what happened for the AX framework, the most significant difference among answers in the two questionnaires was for question  $q_2$ : users found the SAHARIAN browser extension significantly less frustrating than whatever strategy and tool to perform accessibility testing and evaluation they were already familiar with. Significantly different answers emerge for the question  $q_3$ , as

participants found SAHARIAN easier to use than the strategy/tool they already knew and use for the same purpose. The remaining questions only confirmed the same trend.

Analyzing the obtained data, it can be safely concluded that the SAHARIAN browser extension (thus the methodology behind it) provides a far superior user experience while performing manual accessibility testing and evaluation of web pages and applications. Combining these results with the quantitative measurements, it can be concluded that SAHARIAN contributes to *making web accessibility more accessible* by reducing the complexity of accessibility testing and evaluation.

### 6.2.3 Evaluating the A11A website

As illustrated in Section 4.3.3, part of the proposed approach to *make web accessibility more accessible* documented in this PhD thesis consists of making existing accessibility resources much more accessible for people who are not accessibility experts. This is achieved by creating A11A, a website that can be thought as a sort of *accessibility wiki* to collect all accessibility resources available on the Internet. In order to facilitate their discovery, such resources are classified on the website according to different criteria; specific features to help developers find out additional information (related content suggestions, content taxonomies, an internal search engine, etc.) are also provided.

In order to perform the evaluation process of the A11A website, both objective and subjective metrics have been leveraged as explained in Section 6.1. Twenty-two different users have been recruited for the test; of these, twelve of them classified themselves as “female”, ten of them as being “male”, and nobody in other possible gender classifications or not wanting to communicate their gender at all. All of them had an age in the twenty five-fifty years old range. All participants declared themselves as having an intermediate level of knowledge about web development, but only basic skills and limited experience with regard to web accessibility.



The test session took place in a room prepared for the occasion; all users performed the test at the same time by means of their laptops, having access to the Internet and whatever resource they see fit for the purpose of completing the proposed tasks. Special attention and various technical arrangements have been adopted to ensure that participants could not communicate during the session to share insights on how to complete the tasks.

Given its characteristics and the goals for which the A11A website has been created, the tasks selection process has been pretty difficult. After evaluating various possible options, it has been decided to *assess in advance* the knowledge, experience and competencies level of all participants regarding digital accessibility by means of a brief interview administered a few days before the session took place. The main goal of this pre-assessment process was to ensure that all participants were not familiar at all with the specific tasks in order to avoid any learning-effect due to their past knowledge and experience. This allowed to carefully craft the tasks so that the efficiency, efficacy and effectiveness of A11A could be confidently assessed; in deed, it has been deemed important to evaluate A11A on supporting developers regarding accessibility-related tasks that involve all the following aspects:

- A significant *information retrieval process*, meaning that users are expected to look for accessibility-related information they do not know already.
- A non-negligible *information understanding process*, meaning that participants had to *reason about* the information retrieved in order to complete the task. It would be desirable for such reasoning to *involve* acquiring additional accessibility-related information to be completed.
- An *information-selection* process, meaning that in order to complete the task users had to figure out what they needed among contradictory information sources. While this may seem extreme, it is argued that its one of the main strengths of A11A to mitigate such problems, and therefore evaluating it regarding this aspect has been deemed very

useful.

Users have been asked to complete two tasks:

- determine the most appropriate file format to distribute *long-form documents* in order to maximize their accessibility (task *A*);
- determine a strategy to embed Math formulas within web pages, in order to guarantee their accessibility (task *B*).

All users involved in the test have been split randomly in two groups of eleven members each, so that the first group (*AB*) could complete the former task accessing the Internet (but not leveraging the A11A website) and the second relying on A11A, while the second group (*BA*) would accomplish the two task in the opposite order (i.e. task *B* without relying on A11A, and task *A* having access to it).

Before the beginning of the test session, participants have been informed on how the session would take place, illustrating the tasks they were expected to complete and how their feedback would have been collected after the end of each task. Participants were expected to describe their *findings* in a given period of time (15 minutes) about the asked question, describing their proposed solution and the reasoning behind it in a few lines of text (no minimum or max length have been established in advance for answers to be considered acceptable).

Immediately after the completion of each task, the UMUX questionnaire was administered along with an additional free-form question in which participants could provide whatever feedback they deemed relevant. During the test process different objective metrics have been measured:

**The *task completion rate*.** Only twelve out of twenty-two users (54,5%) were able to complete the assigned task without having access to A11A; all participants which did not complete it declared in the administered questionnaire that they faced one or a combination of these difficulties:

- they were not able to make sense of contradictory information they found;
- they had an idea of the best possible solution, but were not sure if it was correct due to contradictory information they found;
- they found useful information from sources of which they could not determine the authoritative level;
- they had troubles understanding technical information.

While numbers may suggest a relationship between the task completion rate and the gender distribution among testers, further analysis of the gathered data excluded it for certain. In contrast, 100% of participants were able to complete the task in which they could rely on A11A.

**The *task completion time*.** On average, users took 22 minutes to complete the task in which they did not have access to A11A; they spent the same amount of time completing the task in which the A11A website could be used. At a first glance, this demonstrated that leveraging A11A did not reduce the time required to find a solution for the proposed tasks; however, other than confirming the appropriateness of the task selection, detailed observations in the process revealed that users spent more time examining different related resources on A11A in contrast to looking for, examining and comparing unrelated content when they could not use the website. This suggests they could have grasped a deeper understanding of the topic. The conducted evaluation, though, does not provide valuable data to confirm this observation (which remains only a hypothesis): while it is very clear that users spent the same time completing the task both with and without the A11A website, the evaluation process was not designed to capture how users spent that time.

**The *correctness of the proposed solutions*.** Unfortunately, only seven out of the twelve participants (58,33% of users who answered, that

diminishes to a 31,81% of total users) who completed the task without relying on A11A provided correct (acceptable) solutions for the task. Further analysis of the data suggest that all the incorrect solutions arose from one or more of the following:

- difficulties in understanding the *big picture*, i.e. understanding all the potential issues that had to be overcome by the proposed solutions;
- not realizing how technical resources, even if considered, could have helped overcome the accessibility issues to be dealt with by the proposed solutions;
- incompleteness, i.e. not considering one or more aspects critical to the success of the proposed solution.

In contrast, twenty-one out of the twenty-two answers (95,45%) for the task in which users had access to the A11A website were correct. This suggests that A11A is effective towards enabling users to understand how to deal with accessibility issues even if they have no experience in the specific application scenario, are complex, and require an elaborate reasoning process to be solved.

**The *UMUX scores*.** The standard UMUX questionnaire was administered immediately after the end of each task. The two tasks reported pretty different scores: in fact, if the first one only got a 21 score on average, the second (involving the usage of the A11A website) got an impressive 95 score on average. Significant differences can be found for all four answers in the questionnaire:

- regarding question  $q_1$ , participants noted that in both scenarios (both having access to A11A and not) their requirements were satisfied accounting only for a slightly higher score towards A11A;
- question  $q_2$  clearly shows that users found leveraging A11A much

less frustrating than using all other resources available on the Internet;

- interestingly, question  $q_3$  confirmed that using A11A made it much easier for people to find the accessibility resources they needed;
- finally, question  $q_4$  clearly confirmed that A11A made users find the information they needed much faster. Once again, this may confirm the hypothesis made while discussing the *task completion rate*: in fact, there is no substantial difference in the time spent by users completing both tasks, therefore it may be assumed that participant felt much more confident in the solution provided while relying on A11A as they grasped a deeper understanding of the topic (and therefore perceived the time spent as an investment rather than a waste).

Analyzing the obtained data, it can be safely concluded that the A11A website can be considered an extremely useful resource as it *bridges* a large gap for people who are not accessibility experts, but are interested in solving accessibility issues in their projects (even if they are not familiar with them). Therefore, it can be concluded that it contributes to *making web accessibility more accessible* as it makes it easier for developers to find accessibility-related resources in a practical, intuitive way.

### 6.3 Evaluating the *SCAMP* methodology and A11YVT

As illustrated in Section 3.3, the second branch of the research project documented in this PhD thesis involved leveraging digital technologies to make indoor environments and cultural heritage more accessible by means of A11YVT (the prototype of a system that lets blind and visually impaired people, as well as their sighted peers, *explore* indoor environments by means of accessible virtual tours), and the SCAMP methodology (which involves a

combination of digital technologies, scientific and art skills to make scientific decomposable artifacts with a high historical and educational value more accessible), respectively. Prototypes have been created for both proposed solutions, and therefore they have been evaluated; in this section their testing and evaluation process will be described, discussing the obtained results.

### 6.3.1 Evaluating the *SCAMP* methodology

As described in Section 4.5.2, as part of the second branch of the research project documented in this PhD thesis the SCAMP methodology (consisting in a theoretical and practical *modelling* process involving the usage of multimedia technologies to *enhance* the specificities of modelled objects) to make scientific, manipulable artifacts with a high historical value *accessible again* both to blind and visually impaired people and the general public has been proposed. To demonstrate the practical feasibility of the SCAMP methodology, it has been applied to the Brendel model of *Phaseolus vulgaris* Linn. shown in Fig. 4.4 (which as explained in Section 4.5.2 shows the early phases of the germination process of a bean plant); in this section, the evaluation process of the produced experience will be described, along with the discussion of the obtained results.

The SCAMP methodology heavily relies upon digital technologies to achieve its goal while putting a very strong focus on *preserving* the physical nature of such objects and their interactivity. Therefore, during the test planning phase, it has been into account these factors to establish where and how the artifact should be tested. After evaluating several options, it has been established that users would *test* the whole experience as designed and imagined, as if they were enjoying it in a real, interactive art installation. Therefore, the testing session took place in a room specifically prepared for the occasion by:

- installing the required central unit, along with a screen and sound speakers, to make the system actually work;

- adjusting lighting in the room to create optimal visual conditions for the experience to be enjoyed;
- moving furniture to let users *live* an enjoyable experience;
- ensuring no distractions or interruptions would stop any user in the middle of the testing process;
- preparing a space in which an interview could be administered to test participants, if necessary.

In order to perform the evaluation process of the artifact, twelve different users have been recruited; of these, six of them classified themselves as “male”, six of them as being “female”, and nobody in other possible gender classifications or not wanting to communicate their gender at all. One participant was blind, another one was visually impaired, and the other ten were sighted people. Users have been recruited randomly, via a non-judgment sample method, but ensuring that at least obtaining results from at least a blind and a visually impaired person was possible. The testing process has been conducted in a single session in a room specifically prepared for the occasion; participants had to wait outside, so that the testing was performed by a single person at a time. Special attention has been paid to ensure that participants did not talk to each other about the test before the end of the session in order to avoid the development of possibly biased results.

Each participant was briefly introduced to the scope of the testing, providing some details about the context in which the experience has been created. He/she was then invited to start *enjoying* the experience. Results have been collected by means of a user observation process performed by three instructed observers, as well as an interview administered after the test to obtain further insights or clarify doubtful observations emerged in the test. More specifically, the complete interaction has been broken down into various tasks which have been evaluated iteratively. Therefore, results have been collected on a task-by-task basis to better isolate possible perceived

weaknesses of the experience or steps in which users faced more difficulties. Therefore, the obtained results are as follows:

1. *Come closer to the installation.* The rhythmic glowing of the LED strips, together with the sound emanating from the prototype, worked well for all users as a “seductive” mechanism “inviting” them to come closer to the prototype and physically interact with it. While this statement holds true for the performed test, it is worth mentioning that in the testing phase the prototype was not placed in an exhibition context together with other artifacts of possible interest for the user; therefore, it has not been possible to evaluate the relative attractiveness of this model in comparison to other parts of a more complex exhibition.
2. *Grab the artifact and place it on a circular base.* During this phase, almost all users had a natural intuition of the action necessary to start the interaction: while this action was felt by most of them as a proper metaphor for “planting” the artifact and let it grow (aided by visual feedback and isomorphism of the round base and the bottom part of the artifact itself), a sighted user has not been able to complete this task without additional hints. When asked, in retrospect, this user admitted that it would have been clearer to have a more particular shape corresponding between the two pieces, quoting her “like in a puzzle”.
3. *Touch the surface of the bean model.* All the users completed this task efficiently, with negligible time differences. The voice-over invitation to explore the model triggered the desired action (i.e. to touch the model in all of its parts): in particular, the glowing parts constituted an additional feedback for tactile interaction. The capacitive sensor for this task was initially not sensitive enough to the touch, and required one user to touch the bean multiple times in order to make it work properly. Multiple testing iterations were useful to calibrate the sensor’s sensitivity to a definitive value. This highlights that special attention needs to



be paid to this specific aspect, as internal testing and fine-tuning the sensor were not enough; further investigation is required to establish whether specific factors can have an impact in this area, isolate them and introduce appropriate fixes to make the prototype more robust.

4. *Touch the surface of the leaf.* As for the previous task, all users considered this action to be very intuitive. Moreover, the users who took more time in the previous task now felt the interaction as more obvious, thus resulting in a faster completion.

Combining the outcomes of all tests, the arising of a very fast learning curve from the users emerges clearly; on the other hand, this also testified a restriction of the heuristic possibilities of the interaction, resulting in reduced serendipities [MC17]. The invitation made to the user in the first place is to explore the model, in the context of an interaction where there are no possible mistakes. All *call to action* invitations keep an exploratory value in themselves; the ideal balance of interplay lies in this trade-off between discovery and achievement. Therefore, instead of trying to fix possible “interaction mistakes” (or, in other words, unpredictable behaviors) the proposed experience makes users draw inspiration from these original actions to create new, spontaneous activities that can reflect new possibilities for a personal discovery of the artifacts.

No special observations have been made about the interactions performed by the blind or the visually impaired participant; this confirms that, as supposed initially, the characteristics of the conceived experience make it *accessible by default* without any special adaptation required, in a truly universal design fashion. Along with the consideration made before about the interactions, it can be concluded that the evaluation process confirms that the *SCAMP* methodology can successfully make scientific, manipulable artifacts with a high historical value more accessible, both for blind and visually impaired people and the general public.

### 6.3.2 Evaluating A11YVT

As discussed in Section 4.5.1, part of the research project documented in this PhD thesis involved providing an approach to make indoor environments more accessible for blind and visually impaired people by designing and developing a prototype for A11YVT, a system that enables the creation of accessible virtual tours. As already illustrated in Section 5.6.1, virtual tours have a huge potential for improving the accessibility of indoor environments, as they consist of simulations of existing locations composed of a sequence of videos, still images or 360-degree images and other multimedia elements such as sound effects, music, narration, text and floor maps. However, to the knowledge of this author there is no system readily available that satisfies all the requirements described in Section 4.5.1 and Section 5.6.1 which A11YVT has been developed upon. In this section the evaluation process performed on a tour implemented using A11YVT will be described, providing some considerations on the obtained results and insights.

First off, given the characteristics of A11YVT, its target audiences naturally arise from the *ratio* behind its development; in fact, as its purpose is to enable the creation of accessible virtual tours that could be experience by blind and visually impaired people as well as their sighted peers, it is immediately clear that evaluating the usability of such a system should involve both user segments. With regard to the expectations of sighted people, though, A11YVT is very typical in its area; being based upon `ThreeJS` [Mrd22], a very well established `JavaScript` library for creating interactive 3D models within web applications which is often leveraged for the development of virtual tours of 3D environments, it is reasonable to assume that A11YVT is pretty close to its competitors in terms of provided features, look and feel, user interface, and user experience. In deed, these goals have been drivers for the prototype design and development. Therefore, it has been decided to evaluate the prototype considering only its *unique* characteristics involving the ability to make virtual tours accessible for blind and visually impaired people, in turn increasing the accessibility of indoor environments for which

they are created. Additionally, being in an early prototype stage, all the features necessary to the *authoring* such accessible virtual tours have not been developed yet, thus their usability cannot be evaluated.

Given these considerations, assessing the usability of the A11YVT prototype involved performing an evaluation process of its accessibility-related features. Therefore, it has been decided to perform a preliminary evaluation with five blind users and a visually impaired person. In fact, despite the system having been designed and implemented in a team including a blind person, it has been deemed useful to assess whether the assumptions made in the design and development phases would get confirmed with real users testing. Additionally, quantifying the experienced usability and identifying the main engagement levers could provide very useful insights to drive future developments to the system.

Special attention has been paid to the user's recruitment process. In deed, fully understanding a system such as A11YVT and providing useful insights for its future developments requires blind and visually impaired people to have at least basic O&M skills and some experience in walking independently both indoors and outdoors. Therefore, it has been decided to recruit users based on personal contacts in the development team: this choice may have introduced biased results, in the sense that only the perceptions of people with good O&M capabilities have been considered, de facto ignoring people not having such skills. However, it is argued that:

- A11YVT is a pretty advanced tool from the point of view of O&M skills and capabilities, as it allows blind and visually impaired people to *explore* indoor environments that they are not already familiar with; therefore, assuming that blind and visually impaired people interacting with it have at least basic O&M skills is acceptable;
- introducing features to make the system more usable for blind and visually impaired people not having good enough O&M skills can be done in a later stage, once the system is evaluated and its effectiveness is proven by more experienced people with disabilities;

- considering that A11YVT is in the early prototype stage, and no attention as been paid at all towards making it usable for people non having good O&M skills, including these people in the evaluation process would give unreliable results.

To actually evaluate the A11YVT prototype, two very well-known scales have been exploited: the *SUS* [Bro96] and the User Engagement Scale (UES) short version [OCH18]. In both cases test participants are required to grade a set of statements on a five-values Likert scale (with values ranging from one, strongly disagree, to five, strongly agree) based on their accordance with each of them.

As previously mentioned, five blind people and a visually impaired people have been recruited for the evaluation process; of these, five of them declared themselves as being male, one as being female, and nobody in other possible gender classifications or not wanting to communicate their gender at all. All participants declared their age to be in the twenty five-fifty years old range. The evaluation process took place in person in a single session, ensuring that the tests were conducted one at a time and that participants could not talk to each other about it in the process. The session started with a brief introduction of the system, warning the user that participation in the study was purely voluntary and no additional personal data other than their response, gender and age would be stored or otherwise processed in any way.

In order to perform the test, a virtual tour presenting the main building of the Cesena University Campus has been exploited. All participants were not already familiar with the presented building (as they did not know how to walk independently to reach any point of interest within it); they were asked to complete two different, yet related tasks:

- become confident with A11YVT and freely navigate the environment, in order to understand how the system works;
- find the room of the student association contained in the building, determining the path to get to it from the entrance.

At the end of each interaction (which in average lasted twenty-five minutes with a standard deviation of seven minutes), user's feedback has been collected by administering a questionnaire made of SUS (ten questions) and UES short form (twelve questions) immediately after. Data (both qualitative and quantitative) have been collected orally and recorded. Users were encouraged to use their laptops to navigate the virtual tour in order to avoid obtaining biased results due to issues related to the usage of a different device, screen reader or assistive technologies different from their habitual ones.

On average, the system obtained a very high SUS score of 90 out of 100: given the requirements that influenced the design and development of A11YVT, this made it clear that it fully satisfies them, they are effective towards making indoor environments more accessible, and let blind and visually impaired people that are not familiar with them safely explore such spaces. Some qualitative and very useful comments have been collected as well; among others, it is worth mentioning feedback obtained:

- from question  $q_7$  of the SUS questionnaire, stating "I found the system very cumbersome to use": four out of six participants commented that Screen reader output provided by A11YVT could be refined to convey the same information with shorter sentences, but they considered it a minor issue as output was not too verbose to handle.
- from question  $q_6$  of the SUS questionnaire, stating "I would imagine that most people would learn to use this system very quickly": all participants strongly remarked that having some support documentation would be critical to the success of the system; all of them agreed on the fact that the provided keyboard shortcuts for taking the tour are extremely useful, but it is necessary to provide some documentation explaining that they exist, how they are supposed to work, and perhaps indicating users how to take advantage of them with practical examples.

The UES short form allowed to evaluate A11YVT by measuring four different

dimensions: Focused Attention (FA), Perceived usability (PU), Aesthetic appeal (AE), and Reward factor (RW). Results show that the experience had a positive impact on PU for the system; in fact, A11YVT was perceived by all participants as easy to use, not confusing at all and not taxing. Considering the AE dimension, on average A11YVT obtained a score of 3 out of 5; a very similar score was obtained when inquiring about FA as well. Finally, a much higher score of 4.34 out of 5 was obtained on average when investigating RW, as all participants strongly claimed that *using A11YVT was worthwhile* and the provided experience was very rewarding for them. Once again, this confirms that A11YVT is effective towards letting blind and visually impaired people explore indoor environments they are not familiar with, therefore making such spaces more accessible for them.

# Chapter 7

## Conclusions and future developments

Starting from an in depth analysis of the current situation, the research project documented in this PhD thesis dealt with two different, yet related problems regarding digital accessibility. On one hand, realizing the abundance of web content that exhibit critical accessibility issues, all the efforts that have been done over the years to improve the current situation, the presence of technical resources as well as other support materials helping stakeholders (be they designers, managers, content authors, web developers, and others) produce accessible content, and the fact that such resources come in a variety of flavors to satisfy a wide range of possible requirements and expectations, it is argued that a completely different approach is necessary. Therefore, an innovative, threefold approach to *rethink from the ground up* the way in which accessibility is dealt with has been proposed. Many differences in between the proposed approach and the current situation with web accessibility could be mentioned, but perhaps the most significant one is the *paradigm shift* that it brings to the table:

- Instead of leaving to developers the ability to *choose* between different implementations for the same elements, along with the burden of determining their accessibility by understanding complex standards and

techniques, a single solution that *generates* accessible markup by default is provided; this is achieved by means of the declarative **AX** framework, which provides a set of web components providing the UI controls commonly required while developing websites and applications.

- Instead of requiring stakeholders to understand complex accessibility tools, how they work, the way in which they communicate accessibility issues and the related support resources illustrating how to take advantage of them, an innovative accessibility testing and evaluation methodology that maps *accessibility-related concepts* on concepts that stakeholders are already familiar with has been proposed; such methodology has been implemented in the **SAHARIAN** browser extension available for **Google Chrome**. As explained in Section 4.3.1 and Section 5.2, it exploits both the visual rendering and mouse operability of a web page to reflect its accessibility, letting users *directly perceive* (i.e. in first person) the presence of accessibility issues in the code being tested and their impact on people with disabilities.
- Instead of requiring developers to know what to look for, how and where when it comes to searching for accessibility-related resources, a structured, categorized repository to collect and classify such resources according to various criteria has been created; it has been implemented by means of the **A11A** website and allows stakeholders to easily find what they need, whether they know they need it or not. Along with an accurate resource classification according to various criteria, additional features (such as an internal search engine and related content suggestions) have been provided to encourage and promote the discoverability of such resources by their intended audiences.

Evidence shows that designers, web developers, content authors and all stakeholders in general have a hard time with the tools and resources that are available today to deal with digital accessibility; many attempts to improve them have been proposed, but observing the current reality and data



gathered in the literature they did not bring any significant success. Additionally, as evidence shows that accessibility advances when technology actually makes it easier to create accessible websites and applications rather than due to an increased interest among stakeholders, this lead to the conclusion that such a paradigm shift (thus an innovative approach to *rethink from the ground up* and *make web accessibility more accessible*) is necessary. The lack of success among the proposed solutions, in fact, may depend on the fact that they assume the traditional *paradigm* in which many alternatives in between accessible and inaccessible content exist, developers can only *indirectly* (either by relying on external tools or third parties to perform the accessibility testing and evaluation process) perceive accessibility issues and *blindly trust* the documentation (guides and tutorials, guidelines, technical specifications, abstract instructions, code examples that do not closely match their implementation, and so on) to understand their impact on people with disabilities.

The innovative approach to *make web accessibility more accessible* proposed and implemented in the research project documented in this PhD thesis changes the current *status quo* in which developing accessible content or performing accessibility testing and evaluation requires highly specialized knowledge, which stakeholder may or may not be willing to acquire, or simply the required additional efforts (cognitive, technical, financial, ETC) may not be affordable in many circumstances.

It is strongly argued that the proposed approach has a great potential in facilitating a more widespread production of accessible web content, as it offers tools to alleviate some of the most significant difficulties that developers, designers, content authors and stakeholders in general have to face in order to do that with the tools available today. Not only that, but the proposed approach illustrates both the theoretical foundations (the *strategy*) behind it and also provides the *practical* results of its application (the *tools*) to actually *make web accessibility more accessible*: they consist of the **AX** framework (to support the development of accessible websites and applications),

the SAHARIAN browser extension for Google Chrome (to implement the innovative methodology to map accessibility-related information to concepts that developers are already familiar with, especially the visual rendering and mouse operability of page), and the A11A website (to make it easier to find accessibility-related resources for their intended audiences). While different attempts to address one or more of the three branches of the proposed strategy to *make web accessibility more accessible* have been proposed over the years, it is argued that none of them started from the point of *rethinking from the ground up* the current approach to digital accessibility.

Finally, it is worth mentioning that the proposed artifacts can express their maximum potential when used in combination; however, they have been designed to provide the expected benefits even if used independently one from the other. This may speed up the adoption process of such tools, as well as making them available in those contexts in which all three of them cannot be used together for technical reasons.

With regard to the second branch of the research project documented in this PhD thesis, two very challenging issues have been dealt with:

- improving the accessibility of scientific, manipulable artifacts both for blind and visually impaired people and the general public; this is achieved by means of the SCAMP methodology, which exploits digital technologies, multimedia content and 3D-printing to *enhance* the characteristics of these artifacts to make them more accessible both for blind and visually impaired people and the general public.
- Improving the accessibility of indoor environments by means of *accessible virtual tours* that can be experienced both by blind and visually impaired people and their sighted peers, relying on all the features they expect to find (and actually need) in such experiences. The A11YVT system has been developed to demonstrate the feasibility and effectiveness of this idea.

While these problems may seem completely unrelated, they have in com-

mon the fact that they actually leverage *digital technologies* to improve the accessibility of physical “things” at a meta-level: creating accessible digital experiences to solve such problems, in fact, the underlying physical *things* become more accessible as well. Many solutions to the same problems have been already proposed in the literature and thus have been widely discussed; however, it is argued that none of them has the unique characteristics of the SCAMP methodology or the A11YVT system.

Another unique strength of the proposed solutions to the problems tackled in the research project documented in this PhD thesis is that, thanks to a very careful analysis of the current scenario followed by another even more careful design phase, all the ideas (aka the *philosophy*) behind each produced artifact is then confirmed when the actual tool is used “out in the wild”; other than many preliminary considerations (both by the author and third parties), this has been confirmed by the evaluation process performed to assess the main artifacts that have been produced.

This being said, though, this is not the end of the story; of course, such a wide research project has room for further developments, improvements and additional researches to improve it. First off, as clearly outlined in Section 5.6.1 and Section 5.6.2 the produced artifacts for the A11YVT system and the SCAMP methodology have been designed as mere *prototypes* to demonstrate their feasibility. They are neither complete nor ready for free usage out in the wild as their nature would clearly demand. Therefore, additional efforts will be required to promote them to the status of fully fledged products ready for public distribution and consumption.

In case of the SCAMP methodology, for instance, these include making the produced prototypes more robust and *resilient* both to unexpected and unpredictable user interactions and environment conditions (temperature, humidity, visual and audio conditions, and so on); making the implementation process more scalable in terms of implementation time and costs would be desirable as well. Instead, when it comes to the A11YVT system to create accessible virtual tours, the system needs to be developed further in order to

*facilitate* the creation of such tours, make it easier to distribute them, and allow their consumption even in uncontrolled environments; feedback gathered during its evaluation to make its usage easier (i.e. providing detailed instructions to document of to use the system) also needs to be implemented. The feasibility of all these improvements, though, has already been discussed and proven in this PhD thesis: therefore, it can be argued that they can be implemented for certain.

Even if the artifacts proposed to make web accessibility more accessible (the AX framework, the SAHARIAN browser extension for Google Chrome, the A11A website, the MARE interactive tool and the ARIA-Service JavaScript library) are in better shape from the point of view of their maturity, more work is necessary to complete and improve them as well. The AX framework, for example, currently provides only a very limited set of components (described in Appendix C) that can be leveraged; to make it actually useful *in the wild*, it has to be expanded to include many more of them: once again, though, the components that have been implemented prove the feasibility of developing all the missing ones. The official documentation of the framework, along with practical usage descriptions of each component, API references and possibly guides and tutorials, also needs to be created and published.

An ongoing maintenance process is then necessary for the A11A website; even if it currently counts about 500 pages, more and more resources can be added as time passes to make it more complete and keep it up to date with the latest advancements in the field of digital accessibility. An interesting opportunity for the project would also be to create a community that could maintain and improve it over time, so that external contributions can be accepted: this would require careful considerations and infrastructure changes to be made, though. Finally, additional features to increase user engagement would also be desirable: the most effective ones, though, have not yet been determined.

Finally, the SAHARIAN browser extension (which implements the proposed innovative methodology for manual accessibility testing and evaluation) could

also benefit from further improvements. As described in Appendix B it currently provides a limited set of features: the first improvement would therefore be to make it complete. As for other artifacts, though, the developed features prove the feasibility of all the missing ones. Making it more robust for usage *out in the wild* could also be possible, yet this needs to be investigated by gathering more user feedback: this suggests that a public beta testing phase after completing the extension could be extremely useful.

With regard to MARE, instead, a more robust pipeline to extract the necessary information from various resources describing the WAI-ARIA specification could be developed; while this would make maintaining up to date the tool over time much easier, the fact that the WAI-ARIA specification is not updated so frequently, though, does not make this a critical priority when compared to other opportunities for future developments. Similar considerations can be done for the **ARIA-Service JavaScript** library, which needs further development to become more robust and support all possible usage scenarios (e.g. make it more robust to detect unexpected usage scenarios or possible race conditions with other libraries that result in unexpected behaviors during event handling).

In general, though, a larger scale evaluation of all the produced artifacts could also be useful, as it would provide more useful insights for further improvements to them and even suggests additional features to increase their impact.

But as they say, unfortunately “all good things must come to an end”. And this is the end for this PhD thesis; the hope, though, is that this does not coincide with the end of the research project itself, and more specifically with the End of Life (EOL) of the produced artifacts. The hope is that they can be improved even in the future, perhaps one day leading to the so much necessary (and wanted) paradigm shift to actually *make web accessibility more accessible* even out in the wild, at a much larger scale. While this process has been proven to be feasible in this PhD thesis, though, the day in which it gets completed and all the documented possible improvements

become a reality has still to come. The ultimate hope of this author, though, is that one day or another the described (and so much necessary) “accessibility revolution” can actually start in the real world.

# Appendices





# Appendix A

## WebAIM one million 2022 - Main WCAG violations

Table A.1 contains the data represented in Fig. 3.1, illustrating the most common WCAG violation types detected while performing the Homepage analysis of the most popular one million websites by WebAIM in 2022.

WCAG failure type	% of Homepages affected
Low contrast text	83.9%
Missing alternative text for images	55.4%
Empty links	50.1%
Missing form input labels	46.1%
Empty buttons	27.2%
Missing document language	22.3%

Table A.1: WCAG violation types detected in the most popular 1 million Homepages by WebAIM

Source: [Web22a]

# Appendix B

## Features of the SAHARIAN browser extension

As described in Section 4.3.1 part of the research project described in this PhD thesis involved developing an innovative manual accessibility testing and evaluation methodology to map accessibility-related concepts to ones that web developers are already familiar with: in particular, the proposed methodology communicates the accessibility of a web page by exploiting its visual rendering (to reflect the information that is conveyed to assistive technologies) and mouse operability (to reproduce the experience that keyboard-only users can have). As illustrated in Section 5.2 this methodology has been then implemented in the SAHARIAN browser extension available for Google Chrome.

In order to make SAHARIAN truly effective and ready to use at large, its features should include:

- coverage for all possible HTML elements;
- coverage for any custom widget (e.g. implemented by means of the WAI-ARIA specification);
- coverage for all possible implementation strategies to implement and develop each possible element that can appear within a web page;

- all accessibility requirements arising from the *WCAG 2.1*, *WAI-ARIA 1.1* and the *ATAG 2.0* specifications;
- additional accessibility requirements arising both from other specifications (e.g. for more specific technologies);
- any possible variation in the actual implementations of the previous elements (e.g. incorrect markup, different technical implementation strategies, specific frameworks, and so on);
- any possible combination of the elements mentioned before.

It is clear that completing such an implementation is unfeasible for a research project; however, it was important to determine a *significant* set of requirements to be implemented in the SAHARIAN browser extension. Such requirements have been chosen so that they could:

- demonstrate the *feasibility* of the proposed methodology, proving that it could be effectively implemented;
- be *diverse enough* one from each other, so that the technical feasibility of all features necessary by a complete implementation of the proposed strategy could be proven;
- allow evaluating the effectiveness of the proposed strategy with actual users in realistic scenarios (i.e. closely matching the ones the tools would be used out in the wild);
- include requirements which may result in *troublesome* situations arising from the actual implementations (i.e. due to known limitations of APIs exposed by web browsers).

It has therefore been decided to follow a different strategy. Instead of analyzing various accessibility requirements, the most natural approach arising from the proposed methodology itself has been adopted. In fact, in order to implement SAHARIAN various *design patterns* have been selected. For each of

those, support for various possible accessibility-related variations has been developed in the extension. It was also critical to ensure that the extension could work as expected no matter what the implementation strategy used for that design pattern in a web page (which cannot be controlled, as it depends on the actual code whose accessibility needs to be evaluated) was. The implemented design patterns include:

- support for all the so-called *landmark elements* (HTML tags like `<nav>`, `<header>`, `<main>`, `<footer>`, and so on) that can be used to define the structure of a page;
- support for HTML elements that can be used to implement form controls (text fields, radio buttons and radio groups, checkboxes, listboxes, and so on);
- support for HTML elements used to represent headings;
- support for HTML elements used to implement images and links;
- support for visual characteristics that can have an impact on the page accessibility (especially font size and color contrast);
- support for all roles matching the previously mentioned elements provided by the WAI-ARIA specification, including all keyboard interactions (if necessary);
- support for specific roles from the WAI-ARIA specification for which HTML does not provide equivalent elements yet (.e.g. tree-views and menubars), including support for the necessary keyboard interactions.

The process to effectively implement the support for each design pattern in the SAHARIAN browser extension has been illustrated in Section 4.3.1 and Section 5.2.



# Appendix C

## Elements in the AX framework

As illustrated in Section 4.3.2 and Section 5.1 part of the research project documented in this PhD thesis to *make web accessibility more accessible* involved the creation of **AX**, a framework capable of enforcing the generation of accessible markup as much as possible. As explained in Section 5.1 the framework has been implemented leveraging *web components*, a suite of technologies that allows the creation of custom **HTML** elements (encapsulating the **HTML**, **JavaScript** and **CSS** code necessary to implement their functionalities) and reuse them across different web apps. Introduced by various popular **JavaScript** libraries (like **React** [Met22], **Vue** [Vue22] and **Angular** [Ang22]), the idea has proven to be so successful to deserve the standardization in a set of different specifications by the W3C.

To be considered complete, the **AX** framework should provide all components necessary to implement each possible widget a developer may ever need during the development of any website or application. In addition, as **HTML** does not enforce the generation of accessible markup (but rather introduces various problems as already illustrated in this PhD thesis, see Section 4.2), makes it clear that the **AX** framework should provide a component to replace each **HTML** element available. Unfortunately, such a complete implementation was not feasible for the scope of a research project; therefore, it was necessary to determine which components to implement to demonstrate the feasibility

of the proposed solution.

The components implemented in the **AX** framework have been selected in order to satisfy the following principles:

- Be *diverse enough*, in order to unveil all the technical challenges that may arise during the complete implementation; in this way, overcoming them would clearly indicate that the complete implementation will be possible in the future.
- Be *representative* of the elements commonly used in the development of websites and applications; while official data on the popularity of web components or specific HTML elements across the Web is not available, evidence shows that the major (i.e. most popular) JavaScript frameworks providing UI controls tend to share a set of essential components. The presence in such a set influenced the decision to implement it or not.
- Be *complex enough* to represent all the challenges that would be necessary to face from an accessibility point of view; in this way, making the **AX** framework capable of enforcing the generation of accessible markup in such scenarios will ensure that even its complete implementation will be able to achieve the goal for which it was designed.
- Allow evaluating the effectiveness of the framework with actual users in realistic scenarios; in this way, the effectiveness of the **AX** framework could be evaluated with actual users representing its target in real-world scenarios closely matching the ones it would be used out in the wild.

As described in Section 4.3.2 and Section 5.1, though, in order to achieve its goal the **AX** framework has to perform some analysis at runtime (i.e. after the page has finished rendering), as some checks necessary to enforce the generation of accessible markup cannot be performed statically (think of color contrast, that can be influenced by factors that cannot be controlled



by the framework such as styling). Some components integrating this kind of analysis have been included in the set as well.

In light of these considerations, the components that have been developed to prove the feasibility of implementing the AX framework include:

- elements to represent the page structure (i.e. the so-called page landmarks);
- HTML elements critical to ensure the accessibility of the structure of a page (e.g. headings);
- form controls (text fields, radio buttons and radio groups, checkboxes, list boxes, sliders, and more);
- widgets commonly used in websites and applications, yet not represented by *native* HTML elements (e.g. date pickers, tabs, tab lists and tab panels, treeviews);
- elements that should be provided by HTML natively, but for various reasons do not work as expected (e.g. to implement modal dialogs and autocomplete fields);
- elements that allow linking a page to another, or a point of a page to another point within the same (e.g. buttons and links);
- elements to represent *data tables* (which unlike their native HTML counterparts can prevent their usage for layout purposes)<sup>1</sup>;
- elements for embedding images (both for decorative purposes and not).

The implementation process of the framework, as well as the technical challenges faced and the strategies adopted to overcome them, have been thoroughly discussed in Section 5.1.

---

<sup>1</sup>A trend originating from the early 90s, an era in which CSS was very different from what we know today, involved *abusing* tables in order to define the layout of a page; this problem is widely known as *table-based layouts* within the accessibility community and can cause a lot of troubles for assistive technology users.



# Appendix D

## A class diagram for the AX framework

In order to illustrate some architectural decisions behind the implementation of the **AX** framework proposed in the research project documented in this PhD thesis, an UML class diagram to show some of its components has been provided in Fig. 5.1. In this appendix you can find the textual representation from which it has been generated by means of `plantUML` [Roq22].

Listing D.1: the `PlantUML` code for the class diagram showing some components of the **AX** framework

```
@startuml
class HTMLElement << Base class for DOM elements >>

package AX {
    class AXAttribute {
        + constructor()
        + string name
        + bool validator
    }

    abstract class AXElement {
```

```

    {static} + abstract string [] allowedParents
        .. Initialization ..
    + void constructor ()
    #node: HTMLInputElement
        .. Life cycle callbacks ..
    + void connectedCallback ()
    + void adoptedCallback ()
    + void disconnectedCallback ()
        .. Attributes callbacks ..
    {static} + string [] getObservedAttributes ()
    + void attributeChangedCallback ()
    {static} + abstract AXAttribute [] attrs
}
HTMLInputElement <|-- AXElement

abstract class FormField {
    {static} + abstract string [] allowedParents
    {static} + abstract inputType
    + HTMLInputElement render ()
}
AXElement <|-- AXElement

class TextField {
    {static} + string inputType
}
TextField <|-- FormField

class Slider {
    {static} + string inputType
    {static} + AXAttribute [] attrs
}

```

```
FormField <|-- Slider

FormField <|-- Radio
AXElement <|-- Radiogroup
Radiogroup *-- Radio
AXElement <|-- Form
Form *-- Slider
Form *-- TextField
Form *-- Radiogroup

class Table {
    {static} + abstract string [] allowedParents
    {static} + AXAttribute [] attrs
    + HTMLElement render()
    .. Cell state support ..
    - int currentRowIndex
    - int currentColIndex
    ~ bool isHeader()
}
AXElement <|-- Table
AXElement <|-- TableCell
AXElement <|-- TableRow
Table *-- TableRow
Table *-- TableCell
TableRow *-- TableCell

TextField <|-- Combobox
Form *-- Combobox

    AXElement <|-- Treeview
}
```

@enduml

## Appendix E

# A stylesheet to make different checkbox implementations look the same

In order to pursue the goals of the research project documented in this PhD thesis, and more specifically to develop the proposed innovative methodology for accessibility testing and evaluation (implemented in the SAHARIAN browser extension) and the ARIA-Service JavaScript library, A11YExamples has been developed: it consists of a repository of *accessibility examples* that could provide a wide set of *examples* of elements commonly available in websites and applications implemented in different ways, with different accessibility degrees, but *knowing* the accessibility issues affecting each example in advance. One of the design goals influencing the development of these examples was to make them *look the same* even if implemented with different HTML elements, as that would make developing both SAHARIAN and ARIA-Service much easier.

Listing E.1 shows an example of such stylesheets developed to make very diverse *checkbox* implementations look the same; the result of applying this stylesheet can be seen in in Fig. 5.9 and Fig. 5.10 in Section 5.5, which shows the visual rendering of various checkbox implementations with different

accessibility degrees (on the left) and the HTML code that implements them (on the right).

Listing E.1: CSS stylesheet applied to make them look the same even if implemented with different HTML elements

```
input [ type=checkbox ]::before {
    width: 1.1em;
    height: 1.1em;
    border: 1px solid hsl(0, 0%, 66%);
    border-radius: 0.2em;
    background-image: linear-gradient(to bottom, hsl(0,
        0%, 93%), #fff 30%);
}

.checkbox::before {
    width: 0.875em;
    height: 0.875em;
    border: 1px solid hsl(0, 0%, 66%);
    border-radius: 0.2em;
    background-image: linear-gradient(to bottom, hsl(0,
        0%, 93%), #fff 30%);
}

input [ type=checkbox ],
.checkbox {
    display: inline-block;
    position: relative;
    padding-left: 1.4em !important;
    cursor: pointer;
    white-space: nowrap;
    outline: none;
}
```



```
input [ type=checkbox ] :: before ,
input [ type=checkbox ] :: after ,
.checkbox :: before ,
.checkbox :: after {
  position: absolute;
  left: 0.4375em;
  top: 50%;
  transform: translate(-50%, -50%);
  content: ' ';
}

input [ type=checkbox ] : checked :: after ,
.checked :: after {
  display: block;
  width: 0.25em;
  height: 0.4em;
  border: solid #fff;
  border-width: 0 0.125em 0.125em 0;
  transform: translateY(-65%) translateX(-50%)
    rotate(45deg);
}

input [ type=checkbox ] : checked :: before ,
.checked :: before {
  display: block;
  border-color: #DDA374;
  background: hsl(217, 95%, 68%);
  background-image: linear-gradient(to bottom,
    #f6b681, #f18d3a);
}
```

```
input [ type=checkbox ]: focus :: before ,
.checkbox: focus :: before ,
input [ type=checkbox ]: hover :: before ,
.checkbox: hover :: before {
  width: 16px;
  height: 16px;
  box-sizing: content-box;
  border-color: #f4a869;
  border-width: 0.1875em;
  border-radius: calc(0.2em + 0.1875em);
  box-shadow: inset 0 0 0 1px #f6b681;
}
```

# Appendix F

## My publications

In this chapter you can find all research publications authored by myself, or were I have been involved in the authorship process with other people. The majority of them is strictly related to topic of web accessibility, while some others can be related to accessibility in a broader scope (such as including accessibility to indoor environments or cultural heritage). All publications are sorted in chronological order.

- Mirri, S., Peroni, S., Salomoni, P., Vitali, F., & Rubano, V. (2017, January). Towards accessible graphs in HTML-based scientific articles. In 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 1067-1072). IEEE .
- Rubano, V., & Vitali, F. (2020, July). Experiences from declarative markup to improve the accessibility of HTML. In Balisage: The Markup Conference.
- Rubano, V., & Vitali, F. (2021, January). Making accessibility accessible: strategy and tools. In 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC) (pp. 1-6). IEEE.
- Lengua, C., Rubano, V., & Vitali, F. (2022, January). Aligning accessibility design to non-disabled people's perceptions. In 2022 IEEE 19th

Annual Consumer Communications & Networking Conference (CCNC) (pp. 1-6). IEEE.

- Rubano, V., Vitali, F., & Lengua, C. (2022). Tools for an Innovative Approach to Web Accessibility. In International Conference on Human-Computer Interaction (pp. 97-115). Springer, Cham.
- Grillo, R., Morelli, C., Rubano, V., & Vitali, F. (2022, September). Social Good and Cultural Heritage: making the Brendel models accessible again. In Proceedings of the 2022 ACM Conference on Information Technology for Social Good (pp. 191-197).
- Bertani, S., Rubano, V., Mirri, S., & Prandi, C. (2023, January). Accessibility for Virtual Tours: on Designing a Prototype for People with Visual Impairments. In 2023 IEEE 20th Annual Consumer Communications & Networking Conference (CCNC) (pp. 1-6). IEEE.

# Bibliography

- [AAK<sup>+</sup>16] Giorgos Anagnostakis, Michalis Antoniou, Elena Kardamitsi, Thodoris Sachinidis, Panayiotis Koutsabasis, Modestos Stavrakis, Spyros Vosinakis, and Dimitris Zissis. Accessible museum collections for the visually impaired: combining tactile exploration, audio descriptions and mobile gestures. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 1021–1025, 2016.
- [AAM20] Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. Accessibility issues in android apps: State of affairs, sentiments, and ways forward. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1323–1334. IEEE, 2020.
- [AAS22] Muhammad Akram, Ghassan Ahmed Ali, and Mahmood Sulaiman, Adel ul Hassan. Accessibility evaluation of arabic university websites for compliance with success criteria of WCAG 1.0 and WCAG 2.0. *Universal Access in the Information Society*, pages 1–16, 2022. doi:<https://doi.org/10.1108/14678040710760603>.
- [AAV19] Julio Abascal, Myriam Arrue, and Xabier Valencia. Tools for web accessibility evaluation. In *Web Accessibility*, pages 479–503. Springer, 2019.

- [AAVLM18] Tania Acosta, Patricia Acosta-Vargas, and Sergio Lujan-Mora. Accessibility of egovernment services in latin america. In *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 67–74. IEEE, 2018.
- [AAVZMLM20] Tania Acosta, Patricia Acosta-Vargas, Jose Zambrano-Miranda, and Sergio Lujan-Mora. Web accessibility evaluation of videos published on youtube by worldwide top-ranking universities. *IEEE Access*, 8:110994–111011, 2020.
- [AAW<sup>+</sup>18] W Arasid, AG Abdullah, D Wahyudin, CU Abdullah, I Widiaty, D Zakaria, N Amelia, and A Juhana. An analysis of website accessibility in higher education in indonesia based on wcag 2.0 guidelines. In *IOP Conference Series: Materials Science and Engineering*, volume 306-1, page 012130. IOP Publishing, 2018.
- [ABAQ21] Zainab AlMeraj, Fatima Boujarwah, Dari Alhuwail, and Rumana Qadri. Evaluating the accessibility of higher education institution websites in the state of kuwait: empirical evidence. *Universal Access in the Information Society*, 20(1):121–138, 2021.
- [AC21] Sarah Alismail and Wallace Chipidza. Accessibility evaluation of covid-19 vaccine registration websites across the united states. *Journal of the American Medical Informatics Association*, 28(9):1990–1995, 2021.
- [Acc22a] Accessibility Guidelines Working Group (AGWG). Essential components of web accessibility. Technical report, World Wide Web Consortium (W3C), 2022. Last accessed on December 07, 2022. URL: <https://www.w3.org/WAI/fundamentals/components/>.

- [Acc22b] Accessibility Guidelines Working Group (AGWG). Techniques for wcag 2.1. Updated 01 December 2022, World Wide Web Consortium (W3C), December 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/WCAG21/techniques/>.
- [Acc22c] Accessibility Guidelines Working Group (AGWG). Understanding web content accessibility guidelines (WCAG) 2.1. Updated 1 December 2022, World Wide Web Consortium (W3C), December 2022. Last accessed on December 07, 2022. URL: <https://www.w3.org/WAI/WCAG21/Understanding/>.
- [ACD<sup>+</sup>18] Glenn Adams, Cyril Concolato, Mike Dolan, Sean Hayes, Frans de Jong, Dae Kim, Pierre-Anthony Lemieux, Nigel Megitt, Dave Singer, Jerry Smith, and Andreas Tai. Timed text markup language 2 (TTML2). W3C recommendation 08 november 2018, World Wide Web Consortium (W3C), November 2018. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2018/REC-ttml2-20181108/>.
- [ACM<sup>+</sup>22] Chuck Adams, Alastair Campbell, Rachael Montgomery, Michael Cooper, and Andrew Kirkpatrick. Web content accessibility guidelines (WCAG) 2.2. W3C candidate recommendation snapshot 06 september 2022, World Wide Web Consortium (W3C), September 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2022/CR-WCAG22-20220906/>.
- [AD17] Tahani Alahmadi and Steve Drew. Accessibility evaluation of top-ranking university websites in world, oceania, and arab categories for home, admission, and course descrip-

- tion webpages. *Journal of Open, Flexible and Distance Learning*, 21(1):7–24, 2017.
- [Ada22] Paul J. Adam. A11YTools extension for safari, 2022. Last accessed on December 06, 2022. URL: <https://pauljadam.com/extension.html>.
- [ADBMAAB13] Iyad Abu-Doush, Ashraf Bany-Mohammed, Emad Ali, and Mohammed Azmi Al-Betar. Towards a more accessible e-government in jordan: an evaluation study of visually impaired users and web developers. *Behaviour & Information Technology*, 32(3):273–293, 2013.
- [AFAKARAD13] Auhood Al-Faries, Hend S Al-Khalifa, Muna S Al-Razgan, and Mashael Al-Duwais. Evaluating the accessibility and usability of top saudi e-government services. In *Proceedings of the 7th International Conference on Theory and Practice of Electronic Governance*, pages 60–63, 2013.
- [AFGM10] Fernando Alonso, José Luis Fuertes, Ángel Lucas González, and Loïc Martínez. On the testability of wcag 2.0 for beginners. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, pages 1–9, 2010.
- [AGR+16] Dragan Ahmetovic, Cole Gleason, Chengxiong Ruan, Kris Kitani, Hironobu Takagi, and Chieko Asakawa. Navcog: a navigational cognitive assistant for the blind. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 90–99. ACM, 2016.
- [AHV16] Amaia Aizpurua, Simon Harper, and Markel Vigo. Exploring the relationship between web accessibility and user ex-



- perience. *International Journal of Human-Computer Studies*, 91:13–23, 2016.
- [AIN10] Maslina Abdul Aziz, Wan Abdul Rahim Wan Mohd Isa, and Norzainuriah Nordin. Assessing the accessibility and usability of malaysia higher education website. In *2010 International Conference on User Science and Engineering (i-USEr)*, pages 203–208. IEEE, 2010.
- [AKBA17] Hend S Al-Khalifa, Ibtehal Baazeem, and Reem Alamer. Revisiting the accessibility of saudi arabia government websites. *Universal Access in the Information Society*, 16(4):1027–1039, 2017.
- [Akg18] Yakup Akgül. Banking websites in turkey: An accessibility, usability and security evaluation. *İşletme Araştırmaları Dergisi*, 10(1):782–796, 2018.
- [AKS22] Gaurav Agrawal, Devendra Kumar, and Mayank Singh. Assessing the usability, accessibility, and mobile readiness of e-government websites: a case study in india. *Universal Access in the Information Society*, 21(3):737–748, 2022.
- [Ala22] Nancy Alajarmeh. The extent of mobile accessibility coverage in wcag 2.1: Sufficiency of success criteria and appropriateness of relevant conformance levels pertaining to accessibility problems encountered by users who are visually impaired. *Universal Access in the Information Society*, 21(2):507–532, 2022.
- [ALPS15] James Allan, Greg Lowney, Kim Patch, and Jeanne Spellman. User agent accessibility guidelines (UAAG) 2.0. W3C working group note 15 december 2015, World Wide Web

- Consortium (W3C), April 2015. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2015/NOTE-UAAG20-20151215/>.
- [AMK13] M Al Mourad and Faouzi Kamoun. Accessibility evaluation of dubai e-government websites: Findings and implications. *Journal of E-Government Studies and Best Practices*, 2013.
- [Ang22] Angular. Angular: the modern web developer’s platform, 2022. Last accessed on December 06, 2022. URL: <https://github.com/angular/angular>.
- [AS17] Muhammad Akram and Rosnafisah Bt Sulaiman. A systematic literature review to determine the web accessibility issues in saudi arabian university and government websites for disable people. *International Journal of Advanced Computer Science and Applications*, 8(6), 2017.
- [ASB16] Solomon Adelowo Adepoju, Ibrahim Shehi Shehu, and Peter Bake. Accessibility evaluation and performance analysis of e-government websites in nigeria. *Journal of Advances in Information Technology (JAIT)*, 2016.
- [ASL22] Jinat Ara and Cecilia Sik-Lanyi. Investigation of covid-19 vaccine information websites across europe and asia using automated accessibility protocols. *International Journal of Environmental Research and Public Health*, 19(5):2867, 2022.
- [ASN10] Anas Ratib Al-Soud and Keiichi Nakata. Evaluating e-government websites in jordan: Accessibility, usability, transparency and responsiveness. In *2010 IEEE International Conference on Progress in Informatics and Computing*, volume 2, pages 761–765. IEEE, 2010.

- [AV16] Yakup Akgül and Kemal Vatansever. Web accessibility evaluation of government websites for people with disabilities in turkey. *Journal of Advanced Management Science*, 4(3), 2016.
- [AVA08] Myriam Arrue, Markel Vigo, and Julio Abascal. Including heterogeneous web accessibility guidelines in the development process. In *IFIP International Conference on Engineering for Human-Computer Interaction*, pages 620–637. Springer, 2008.
- [AVHAV<sup>+</sup>20] Patricia Acosta-Vargas, Paula Hidalgo, Gloria Acosta-Vargas, Mario Gonzalez, Javier Guaña-Moya, and Belén Salvador-Acosta. Challenges and improvements in website accessibility for health services. In *International Conference on Intelligent Human Systems Integration*, pages 875–881. Springer, 2020.
- [AVLMSU17] Patricia Acosta-Vargas, Sergio Luján-Mora, and Luis Salvador-Ullauri. Web accessibility polices of higher education institutions. In *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–7. IEEE, 2017.
- [AVSUPMR19] Patricia Acosta-Vargas, Luis Salvador-Ullauri, Jorge Luis Pérez-Medina, and Yves Rybarczyk. Accessibility evaluation of multimedia resources in selected latin america universities. In *2019 Sixth International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 249–255. IEEE, 2019.
- [AZ06] Shadi Abou-Zahra. A data model to facilitate the automation of web accessibility evaluations. *Electronic Notes in Theoretical Computer Science*, 157(2):3–9, 2006.

- [AZMLM20] Tania Acosta, José Zambrano-Miranda, and Sergio Luján-Mora. Techniques for the publication of accessible multimedia content on the web. *IEEE access*, 8:55300–55322, 2020.
- [AZS17] Shadi Abou-Zahra and Michael Squillace. Evaluation and report language (EARL) 1.0. W3C working group note 2 february 2017, World Wide Web Consortium (W3C), February 2017. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2017/NOTE-EARL10-Schema-20170202/>.
- [Bau20] Steven M Baule. Evaluating the accessibility of special education cooperative websites for individuals with disabilities. *TechTrends*, 64(1):50–56, 2020.
- [BB12] Mrinal Kanti Baowaly and Moniruzzaman Bhuiyan. Accessibility analysis and evaluation of bangladesh government websites. In *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 46–51. IEEE, 2012.
- [BBLM11] Maria Claudia Buzzi, Marina Buzzi, Barbara Leporini, and Loredana Martusciello. Making visual maps accessible to the blind. In *International Conference on Universal Access in Human-Computer Interaction*, pages 271–280. Springer, 2011.
- [BBR19] Maria Claudia Buzzi, Marina Buzzi, and Fiorella Ragni. Accessibility of italian e-government services: The perspective of users with disabilities. In *International Conference on Electronic Governance and Open Society: Challenges in Eurasia*, pages 281–292. Springer, 2019.
- [BBV18] Martina Ballarin, C Balletti, and P Vernier. Replicas in cultural heritage: 3d printing and the museum expe-

- rience. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(2), 2018.
- [BCDR21] Barbara Rita Barricelli, Elena Casiraghi, Antonina Dattolo, and Alessandro Rizzi. 15 years of stanca act: are italian public universities websites accessible? *Universal Access in the Information Society*, 20(1):185–200, 2021.
- [BD14] Ibtissam Boussarhan and Najima Daoudi. The accessibility of moroccan public websites: evaluation of three e-government websites. *Electronic Journal of e-Government*, 12(1):67, 2014.
- [BJC+08] Dick Bulterman, Jack Jansen, Pablo Cesar, Sjoerd Mullender, Eric Hyché, Marisa DeMeglio, Julien Quint, Hiroshi Kawamura, Daniel Weck, Xabiel García Pañeda, David Melendi, Samuel Cruz-Lara, Marcin Hanlik, Daniel F. Zucker, and Thierry Michel. Synchronized multimedia integration language (SMIL) 3.0. W3C recommendation 01 december 2008, World Wide Web Consortium (W3C), December 2008. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2008/REC-SMIL3-20081201/>.
- [BJJ+18] Mars Ballantyne, Archit Jha, Anna Jacobsen, J Scott Hawker, and Yasmine N El-Glaly. Study of accessibility guidelines of mobile applications. In *Proceedings of the 17th international conference on mobile and ubiquitous multimedia*, pages 305–315, 2018.
- [BK22] Alexei Birkun and Yekaterina Kosova. Limited accessibility of free online resuscitation education for people with disabilities. *The American Journal of Emergency Medicine*, 56:100–103, 2022.

- [BKMS10] Ioannis Basdekis, Iosif Klironomos, Ioannis Metaxas, and Constantine Stephanidis. An overview of web accessibility in greece: a comparative study 2004–2008. *Universal Access in the Information Society*, 9(2):185–190, 2010.
- [BM12] Muhammad Bakhsh and Amjad Mehmood. Web accessibility for disabled: a case study of government websites in pakistan. In *2012 10th International Conference on Frontiers of Information Technology*, pages 342–347. IEEE, 2012.
- [BP11] Christopher Bailey and Elaine Pearson. Development and trial of an educational tool to support the accessibility evaluation process. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, pages 1–10, 2011.
- [BPC10] David A Bradbard, Cara Peters, and Yoana Caneva. Web accessibility policies at land-grant universities. *The Internet and Higher Education*, 13(4):258–266, 2010.
- [BPDI22] Andrius Budrionis, Darius Plikynas, Povilas Daniušis, and Audrius Indrulionis. Smartphone-based computer vision travelling aids for blind and visually impaired individuals: A systematic review. *Assistive Technology*, 34(2):178–194, 2022.
- [BPG14] Christopher Bailey, Elaine Pearson, and Voula Gkatzidou. Measuring and comparing the reliability of the structured walkthrough evaluation method with novices and experts. In *Proceedings of the 11th Web for All Conference*, pages 1–10, 2014.
- [Bra18] Tim Bray. RFC 8259: The javascript object notation (json) data interchange format. Technical report, Internet

- Engineering Task Force (IETF), December 2018. URL: <https://datatracker.ietf.org/doc/rfc8259/>.
- [BRDC<sup>+</sup>18a] Amelia Bellamy-Royds, Joanmarie Diggs, Michael Cooper, Fred Esch, Rich Schwerdtfeger, and Doug Schepers. Graphics accessibility API mappings. W3C recommendation 06 october 2018, World Wide Web Consortium (W3C), October 2018. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2018/REC-graphics-aam-1.0-20181002/>.
- [BRDC<sup>+</sup>18b] Amelia Bellamy-Royds, Joanmarie Diggs, Michael Cooper, Fred Esch, Rich Schwerdtfeger, and Doug Schepers. WAI-ARIA graphics module. W3C recommendation 06 october 2018, World Wide Web Consortium (W3C), October 2018. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2018/REC-graphics-aria-1.0-20181002/>.
- [Bro96] John Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.
- [BWW10] Bruce B. Blasch, Richard L. Welsh, and William R. Wiener. *Foundations of orientation and mobility, 3RD edition: volume 1, history and theory*. American Printing House for the Blind (APH), June 2010. ISBN: 978-0-89128-448-2.
- [BYH12] Giorgio Brajnik, Yeliz Yesilada, and Simon Harper. Is accessibility conformance an elusive property? a study of validity and reliability of wcag 2.0. *ACM Transactions on Accessible Computing (TACCESS)*, 4(2):1–28, 2012.
- [CCF<sup>+</sup>22] Sen Chen, Chunyang Chen, Lingling Fan, Mingming Fan, Xian Zhan, and Yang Liu. Accessible or not? an empirical

- investigation of android app accessibility. *IEEE Transactions on Software Engineering*, 48(10):3954–3968, 2022. doi:10.1109/TSE.2021.3108162.
- [CCKC15] Michael Cooper, Alastair Campbell, Andrew Kirkpatrick, and Joshue O Connor. Authoring tool accessibility guidelines (ATAG) 2.0. W3C recommendation 24 september 2015, World Wide Web Consortium (W3C), September 2015. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2015/REC-ATAG20-20150924/>.
- [CCKC18] Michael Cooper, Alastair Campbell, Andrew Kirkpatrick, and Joshue O Connor. Web content accessibility guidelines (WCAG) 2.1. W3C recommendation 05 june 2018, World Wide Web Consortium (W3C), June 2018. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2018/REC-WCAG21-20180605/>.
- [CCKL13] P John Clarkson, Roger Coleman, Simeon Keates, and Cherie Lebbon. *Inclusive design: Design for the whole population*. Springer Science & Business Media, June 2013.
- [CH21] Balázs Csontos and István Heckl. Accessibility, usability, and security evaluation of hungarian government websites. *Universal Access in the Information Society*, 20(1):139–156, 2021.
- [CHCL16] Chen Chen, Yi Han, Yan Chen, and KJ Ray Liu. Indoor global positioning system with centimeter accuracy using wi-fi [applications corner]. *IEEE Signal Processing Magazine*, 33(6):128–134, 2016.
- [CHJ<sup>+</sup>19] Michael Crabb, Michael Heron, Rhianne Jones, Mike Armstrong, Hayley Reid, and Amy Wilson. Developing accessible services: Understanding current knowledge and areas



- for future support. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.
- [CHMP21] Daniela Cisneros, Fernando Huamán Monzón, and Freddy Paz. Accessibility evaluation of e-government web applications: A systematic review. In *International Conference on Human-Computer Interaction*, pages 210–223. Springer, 2021.
- [CIM<sup>+</sup>14] David Carlisle, Patrick Ion, Robert Miner, Ron Ausbrooks, Stephen Buswell, Giorgi Chavchanidze, Stéphane Dalmas, Stan Devitt, Angel Diaz, Sam Dooley, Roger Hunter, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Paul Libbrecht, Bruce Miller, Chris Rowley, Murray Sargent, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical markup language (MathML) version 3.0 2nd edition. W3C recommendation 10 april 2014, World Wide Web Consortium (W3C), April 2014. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2014/REC-MathML3-20140410/>.
- [CJLM16] Tania Calle-Jimenez and Sergio Luján-Mora. Accessible online indoor maps for blind and visually impaired users. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 309–310, 2016.
- [CJSGLM14] Tania Calle-Jimenez, Sandra Sanchez-Gordon, and Sergio Luján-Mora. Web accessibility evaluation of massive open online courses on geographical information systems. In *2014 IEEE Global Engineering Education Conference (EDUCON)*, pages 680–686. IEEE, 2014.

- [CM20] Sambhavi Chandrashekar and Lindsay McCardle. How wcag 2.1 relates to online user experience with switch-based tools. *The Journal on Technology and Persons with Disabilities*, pages 223–236, 2020.
- [CMLMG19] Milton Campoverde-Molina, Sergio Luján-Mora, and L Valverde García. Accessibility analysis of electronic documents published in educational web portals: the ecuadorian case. In *EDULEARN19 Proceedings*, pages 4686–4696. IATED, 2019.
- [CMLMG20] Milton Campoverde-Molina, Sergio Lujan-Mora, and Llorenç Valverde Garcia. Empirical studies on web accessibility of educational websites: A systematic literature review. *IEEE Access*, 8:91676–91700, 2020.
- [CNW17] Seyed Ali Cheraghi, Vinod Namboodiri, and Laura Walker. Guidebeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 121–130. IEEE, 2017.
- [Com10] Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. European Disability Strategy 2010-2020: A Renewed Commitment to a Barrier-Free Europe, November 2010. Last accessed on December 06, 2022. URL: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0636:FIN:en:PDF>.
- [Com21] Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. Union

- of Equality: Strategy for the Rights of Persons with Disabilities 2021-2030, March 2021. Last accessed on December 06, 2022. URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM%3A2021%3A101%3AFIN>.
- [CPM18] Carlos Máñez Carvajal, Rocío Fernández Piqueras, and Jose F Cervera Mérida. Evaluation of web accessibility of higher education institutions in chile. *International Education Studies*, 11(12):140–148, 2018.
- [CSB14] Gavin Carothers, Andy Seaborne, and David Beckett. RDF 1.1. A line-based syntax for an RDF graph n-triples. W3C recommendation 25 february 2014, World Wide Web Consortium (W3C), February 2014. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2014/REC-n-triples-20140225/>.
- [CTK08] Sakmongkon Chumkamon, Peranitti Tuvaphanthaphiphat, and Phongsak Keeratiwintakorn. A blind navigation system using rfid for indoor environments. In *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 2, pages 765–768. IEEE, 2008.
- [CVJ99] Wendy Chisholm, Gregg Vanderheiden, and Ian Jacobs. Web content accessibility guidelines 1.0. W3C recommendation 05 june 2018, World Wide Web Consortium (W3C), May 1999. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505>.
- [CWL<sup>+</sup>14] Richard Cyganiak, David Wood, Markus Lanthaler, Graham Klyne, Jeremy J. Carroll, and Brian McBride. RDF

- 1.1 concepts and abstract syntax. W3C recommendation 25 february 2014, World Wide Web Consortium (W3C), February 2014. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
- [DAI12] DAISY Consortium. Ansi/niso z39.86-2005 (r2012) specifications for the digital talking book. standard, National Information Standards Organization (NISO), 2012. Last accessed on December 06, 2022. URL: <https://www.niso.org/publications/ansiniso-z3986-2005-r2012-specifications-digital-talking-book>.
- [DBGV15] F D’Agnano, Catherina Balletti, Francesco Guerra, and Paolo Vernier. Tooteko: A case study of augmented reality for an accessible cultural heritage. digitization, 3d printing and sensors for an audio-tactile experience. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):207, 2015.
- [DBJ18] Julie Ducasse, Anke M Brock, and Christophe Jouffrais. Accessible interactive maps for visually impaired users. In *Mobility of visually impaired people*, pages 537–584. Springer, 2018.
- [DE10] Aram Dulyan and Ernest Edmonds. Auxie: Initial evaluation of a blind-accessible virtual museum tour. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*, pages 272–275, 2010.
- [Deq20] Deque Systems, INC. Axe: accessibility testing for development teams, October 2020. Last accessed on December 06, 2022. URL: <https://www.deque.com/axe/>.

- [Deq22a] Deque Systems, INC. axe-core: accessibility engine for automated Web UI testing, 2022. Last accessed on December 06, 2022. URL: <https://github.com/dequelabs/axe-core>.
- [Deq22b] Deque Systems, INC. Color palette: color palette color contrast analyzer, 2022. Last accessed on December 06, 2022. URL: <https://github.com/dequelabs/color-palette>.
- [Deq22c] Deque Systems, INC. Deque Systems: Web Accessibility Software, Services & Training, December 2022. Last accessed on December 08, 2022. URL: <https://www.deque.com>.
- [DGC18] Joanmarie Diggs, Bryan Garaventa, and Michael Cooper. Accessible name and description computation 1.1. W3C recommendation 18 december 2018, World Wide Web Consortium (W3C), December 2018. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2018/REC-accname-1.1-20181218/>.
- [DKH<sup>+</sup>15] Matjaž Debevc, Ines Kožuh, Simon Hauptman, Andrej Klembas, Julija Bele Lapuh, and Andreas Holzinger. Using wcag 2.0 and heuristic evaluation to evaluate accessibility in educational web based pages. In *International Workshop on Learning Technology for Education in Cloud*, pages 197–207. Springer, 2015.
- [DMC<sup>+</sup>17] Joanmarie Diggs, Shane McCarron, Michael Cooper, Richard Schwerdtfeger, and James Craig. Accessible rich internet applications (wai-aria) 1.1. W3C recommendation 14 december 2017, World Wide Web Consortium (W3C), June 2017. Last accessed on December

- 06, 2022. URL: <https://www.w3.org/TR/2017/REC-wai-aria-1.1-20171214/>.
- [DMDOFdaAF17] Cícero Joasyo Mateus De Moura, Souza De Oliveira, Kenyo Abadio Crosara Faria, and Eduardo Noronha de Andrade Freitas. A new api for android accessibility testing. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 594–598. IEEE, 2017.
- [DMR<sup>+</sup>20] Giovanni Delnevo, Giacomo Mambelli, Vincenzo Rubano, Catia Prandi, and Silvia Mirri. Almawhere 2.0: a pervasive system to facilitate indoor wayfinding. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4. IEEE, 2020.
- [DSC22] Joanmarie Diggs, Alexander Surkov, and Michael Cooper. Core accessibility API mappings 1.2. W3C candidate recommendation snapshot 22 november 2022, World Wide Web Consortium (W3C), November 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2022/CR-core-aam-1.2-20221122/>.
- [DSDH14] Luchin Doblies, David Stolz, Alireza Darvishy, and Hans-Peter Hutter. Pave: A web application to identify and correct accessibility problems in pdf documents. In *International Conference on Computers for Handicapped Persons*, pages 185–192. Springer, 2014.
- [dSdOAB12] Vagner Figueredo de Santana, Rosimeire de Oliveira, Leonelo Dell Anhol Almeida, and Maria Cecília Calani Baranauskas. Web accessibility and people with dyslexia: a survey on techniques and guidelines. In *Proceedings of*

*the international cross-disciplinary conference on web accessibility*, pages 1–9, 2012.

- [DSS<sup>+</sup>17] Joanmarie Diggs, Joseph Scheuhammer, Richard Schwerdtfeger, Michael Cooper, Andi Snow-Weaver, Aaron Leventhal, Bogdan Brinza, James Craig, and Alexander Surkov. Core accessibility API mappings 1.1. W3C recommendation 14 december 2017, World Wide Web Consortium (W3C), December 2017. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2017/REC-core-aam-1.1-20171214/>.
- [DSVE22] Henrique Neves Da Silva, Silvia Regina Vergilio, and André Takeshi Endo. Accessibility mutation testing of android applications. *Journal of Software Engineering Research and Development*, 10:8–1, 2022.
- [DWA17] I Gusti Bagus Ngurah Eka Darmaputra, Sony Surya Wijaya, and Media Anugerah Ayu. Evaluating the accessibility of provinces’ e-government websites in indonesia. In *2017 5th International Conference on Cyber and IT Service Management (CITSM)*, pages 1–6. IEEE, 2017.
- [EBKdN17] Clark Evans, Oren Ben-Kiki, and I döt Net. Yaml ain’t markup language (yaml<sup>TM</sup>) version 1.2., 2017.
- [ECM99] ECMA. EcmaScript language specification. *ECMA (European Association for Standardizing Information and Communication Systems), pub-ECMA: adr., 1999.*
- [EDK14] Trude Eikebrokk, Tor Arne Dahl, and Siri Kessel. Epub as publication format in open access journals: tools and workflow. *The Code4Lib Journal*, 2014.

- [Edw08] Alistair DN Edwards. Assistive technologies. In Simon Harper and Yeliz Yesilada, editors, *Web Accessibility: a foundation for research*, Human-Computer Interaction Series, pages 142–162. Springer, London, 1st edition, 2008.
- [Ell14] Elizabeth Ellcessor. < alt="textbooks">: Web accessibility myths as negotiated industrial lore. *Critical Studies in Media Communication*, 31(5):448–463, 2014.
- [EMC+20] Christin Engel, Karin Müller, Angela Constantinescu, Claudia Loitsch, Vanessa Petrausch, Gerhard Weber, and Rainer Stiefelhagen. Travelling more independently: A requirements analysis for accessible journeys to unknown buildings for people with visual impairments. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–11, 2020.
- [ERGF18] Marcelo Medeiros Eler, José Miguel Rojas, Yan Ge, and Gordon Fraser. Automated accessibility testing of mobile apps. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pages 116–126. IEEE, 2018.
- [EU 22] EU Science Hub. What is open education?, December 2022. Last accessed on December 09, 2022. URL: [https://joint-research-centre.ec.europa.eu/what-open-education\\_en](https://joint-research-centre.ec.europa.eu/what-open-education_en).
- [Eur16] European Union (EU). Directive (eu) 2016/2102 of the european parliament and of the council of 26 october 2016 – on the accessibility of the websites and mobile applications of public sector bodies, 2016. Last accessed on December 06, 2022.



- URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016L2102&qid=1644420047022>.
- [Eur19] European Union (EU). Directive (eu) 2019/882 of the european parliament and of the council of 17 april 2019 – on the accessibility requirements for products and services, April 2019. Last accessed on December 06, 2022. URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019L0882>.
- [EW22] Christin Engel and Gerhard Weber. Atim: Automated generation of interactive, audio-tactile indoor maps by means of a digital pen. In *International Conference on Computers Helping People with Special Needs*, pages 123–133. Springer, 2022.
- [EYYH19] Sukru Eraslan, Victoria Yaneva, Yeliz Yesilada, and Simon Harper. Web users with autism: eye tracking evidence for differences. *Behaviour & Information Technology*, 38(7):678–700, 2019.
- [Far11] Glen Farrelly. Practitioner barriers to diffusion and implementation of web accessibility. *Technology and disability*, 23(4):223–232, 2011.
- [FBB20] Kiran Fatima, Narmeen Zakaria Bawany, and Muniba Bukhari. Usability and accessibility evaluation of banking websites. In *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 247–256. IEEE, 2020.
- [FBC<sup>+</sup>19] Jon E Froehlich, Anke M Brock, Anat Caspi, João Guerreiro, Kotaro Hara, Reuben Kirkham, Johannes Schöning, and Benjamin Tannert. Grand challenges in accessible maps. *interactions*, 26(2):78–81, 2019.

- [FDVS20] Agebson Rocha Façanha, Ticianne Darin, Windson Viana, and Jaime Sánchez. O&m indoor virtual environments for people who are blind: A systematic literature review. *ACM Transactions on Accessible Computing (TACCESS)*, 13(2):1–42, 2020.
- [FF18] Delia Ferri and Silvia Favalli. Web accessibility for people with disabilities in the european union: Paving the road to social inclusion. *Societies*, 8(2):40, 2018.
- [FFP13] Louise Fryer, Jonathan Freeman, and Linda Pring. What verbal orientation information do blind and partially sighted people need to find their way around? a study of everyday navigation strategies in people with impaired vision. *British Journal of Visual Impairment*, 31(2):123–138, 2013.
- [FGGM09] José L Fuertes, Ricardo González, Emmanuelle Gutiérrez, and Loïc Martínez. Hera-ffx: a firefox add-on for semi-automatic web accessibility evaluation. In *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibility (W4A)*, pages 26–35, 2009.
- [FGM11] José L Fuertes, Emmanuelle Gutiérrez, and Loïc Martínez. Developing hera-ffx for wcag 2.0. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, pages 1–9, 2011.
- [Fin10] Kraig Finstad. The usability metric for user experience. *Interacting with computers*, 22(5):323–327, 2010.
- [FKMAZ19] Wilco Fiers, Maureen Kraft, Mary Jo Mueller, and Shadi Abou-Zahra. Accessibility conformance testing (ACT) rules format 1.0. W3C recommendation 31 october 2019,

- World Wide Web Consortium (W3C), October 2019. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2019/REC-act-rules-format-1.0-20191031/>.
- [FLF<sup>+</sup>10] José Faria, Sérgio Lopes, Hugo Fernandes, Paulo Martins, and João Barroso. Electronic white cane for blind people navigation assistance. In *2010 World Automation Congress*, pages 1–7. IEEE, 2010.
- [FOL22] Steve Faulkner, Scott O’Hara, and Patrick H. Lauke. ARIA in HTML. W3C recommendation 27 september 2022, World Wide Web Consortium (W3C), September 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2022/REC-html-aria-20220927/>.
- [FPB10] Daniela Fogli, Loredana Parasiliti Provenza, and Cristian Bernareggi. A design pattern language for accessible web sites. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 307–310, 2010.
- [FRF08] André Pimenta Freire, Cibele M Russo, and Renata Pontin de Mattos Fortes. The perception of accessibility in web development by academy, industry and government: a survey of the brazilian scenario. *New Review of Hypermedia and Multimedia*, 14(2):149–175, 2008.
- [GCO18] Nancy E Guerrón, Antonio Cobo, and José Javier Serrano Olmedo. Virtual reality application for blind people in unknown interior spaces. In *ICAT-EGVE*, pages 157–162, 2018.
- [GGL<sup>+</sup>16] Cole Gleason, Anhong Guo, Gierad Laput, Kris Kitani, and Jeffrey P Bigham. Vizmap: Accessible visual information through crowdsourced map reconstruction. In *Proceed-*

- ings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 273–274, 2016.
- [GHC22] Matt Garrish, Ivan Herman, and Dave Cramer. EPUB 3.3. W3C candidate recommendation draft 14 december 2022, World Wide Web Consortium (W3C), December 2022. Last accessed on December 15, 2022. URL: <https://www.w3.org/TR/2022/CRD-epub-33-20221214/>.
- [GI14] Dimitri Glazkov and Hayato Ito. Introduction to web components. W3C note, World Wide Web Consortium (W3C), July 2014. URL: <https://www.w3.org/TR/2014/NOTE-components-intro-20140724/>.
- [GKL<sup>+</sup>22a] Matt Garrish, George Kerscher, Charles LaPierre, Gregorio Pellegrino, and Avneesh Singh. EPUB Accessibility 1.1: Conformance and discoverability requirements for epub publications. W3C candidate recommendation draft 14 december 2022, World Wide Web Consortium (W3C), December 2022. Last accessed on December 15, 2022. URL: <https://www.w3.org/TR/2022/CRD-epub-a11y-11-20221216/>.
- [GKL<sup>+</sup>22b] Matt Garrish, George Kerscher, Charles LaPierre, Gregorio Pellegrino, and Avneesh Singh. EPUB Accessibility Techniques 1.1. W3C group note 13 september 2022, World Wide Web Consortium (W3C), September 2022. Last accessed on December 15, 2022. URL: <https://www.w3.org/TR/2022/NOTE-epub-a11y-tech-11-20220913/>.
- [GL10] Greg Gay and Cindy Qi Li. Achecker: open, interactive, customizable, web accessibility checking. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, pages 1–2, 2010.

- [GMP<sup>+</sup>13] Ramiro Gonçalves, José Martins, Jorge Pereira, Vitor Santos, and Manuel Pérez Cota. Can i access my school website? auditing accessibility of the portuguese teaching institutions websites. *J. Univers. Comput. Sci.*, 19(18):2639–2655, 2013.
- [GMRV22] Remo Grillo, Caterina Morelli, Vincenzo Rubano, and Fabio Vitali. Social good and cultural heritage: making the brendel models accessible again. In *Proceedings of the 2022 ACM Conference on Information Technology for Social Good*, pages 191–197, 2022.
- [GoH] GoHugo.io. Hugo - the world’s fastest framework for building websites. Last accessed on December 06, 2022. URL: <https://github.com/gohugoio/hugo>.
- [Goo20] Google, INC. Lighthouse - Tools for Web Developers, October 2020. Last accessed on December 06, 2022. URL: <https://developers.google.com/web/tools/lighthouse>.
- [Gov05] Government of Ontario. Accessibility for ontarians with disabilities act (AODA), 2005, s.o. 2005, c. 11, 2005. Last accessed on December 06, 2022. URL: <https://www.ontario.ca/laws/statute/05a11>.
- [Gov10] Government of the United Kingdom (UK). Equality act, 2010. Last accessed on December 06, 2022. URL: <https://www.legislation.gov.uk/ukpga/2010/15/contents>.
- [Gov18] Government of Australia. Disability discrimination act, C2018C00125, 2018. Last accessed on December 06, 2022. URL: <https://www.legislation.gov.au/Details/C2018C00125>.

- [Gov19] Government of Canada. Accessible Canada Act, 2019. Last accessed on December 06, 2022. URL: <https://laws-lois.justice.gc.ca/eng/acts/A-0.6/FullText.html>.
- [GP21] Ombretta Gaggi and Veronica Pederiva. Wcag4all, a tool for making web accessibility rules accessible. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.
- [GPG16] Orazio Gambino, Roberto Pirrone, and Fabrizio Di Giorgio. Accessibility of the italian institutional web pages: a survey on the compliance of the italian public administration web pages to the stanca act and its 22 technical requirements for web accessibility. *Universal Access in the Information Society*, 15(2):305–312, 2016.
- [GPM<sup>+</sup>20] Cole Gleason, Amy Pavel, Emma McCamey, Christina Low, Patrick Carrington, Kris M Kitani, and Jeffrey P Bigham. Twitter ally: A browser extension to make twitter images accessible. In *Proceedings of the 2020 chi conference on human factors in computing systems*, pages 1–12, 2020.
- [GRSW09] Loretta Guarino Reid and Andi Snow-Weaver. Wcag 2.0 for designers: Beyond screen readers and captions. In *International Conference on Universal Access in Human-Computer Interaction*, pages 674–682. Springer, 2009.
- [Gru12] John Gruber. Markdown: Syntax. URL <http://daringfireball.net/projects/markdown/syntax>. Retrieved on June, 24:640, 2012.
- [GSB14] Fabien Gandon, Guus Schreiber, and Dave Beckett. RDF 1.1 XML syntax. W3C recommendation 25 february 2014,

- World Wide Web Consortium (W3C), February 2014. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/>.
- [GSEB<sup>+</sup>12] Neveen I Ghali, Omar Soluiman, Nashwa El-Bendary, Tamer M Nassef, Sara A Ahmed, Yomna M Elbarawy, and Aboul Ella Hassanien. Virtual reality technology for blind and visual impaired people: reviews and recent advances. *Advances in Robotics and Virtual Reality*, pages 363–385, 2012.
- [GSGM17] Matt Garrish, Tzviya Siegman, Markus Gylling, and Shane McCarron. Digital publishing WAI-ARIA module 1.0. W3C recommendation 14 december 2017, World Wide Web Consortium (W3C), December 2017. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2017/REC-dpub-aria-1.0-20171214/>.
- [GST<sup>+</sup>14] Aura Ganz, James M Schafer, Yang Tao, Carole Wilson, and Meg Robertson. Percept-ii: Smartphone based indoor navigation system for the blind. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3662–3665. IEEE, 2014.
- [Had21] Samine Hadadi. Adee: Bringing accessibility right inside design tools. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–4, 2021.
- [HC12] Simon Harper and Alex Q Chen. Web accessibility guidelines. *World Wide Web*, 15(1):61–88, 2012.
- [HCM<sup>+</sup>15] Ibrar Hussain, Ling Chen, Hamid Turab Mirza, Gencai Chen, and Saeed-Ul Hassan. Right mix of speech and non-speech: hybrid auditory feedback in mobility assistance of

- the visually impaired. *Universal Access in the Information Society*, 14(4):527–536, 2015.
- [HH21] Dylan H Hewitt and Yingchen He. Internet accessibility for blind and visually-impaired users: An evaluation of official us state and territory covid-19 websites. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 65-1, pages 154–158. SAGE Publications Sage CA: Los Angeles, CA, 2021.
- [HJLY14] Dongsoo Han, Sukhoon Jung, Minkyu Lee, and Giwan Yoon. Building a practical wi-fi-based indoor navigation system. *IEEE Pervasive Computing*, 13(2):72–79, 2014.
- [HMB18] Leona Holloway, Kim Marriott, and Matthew Butler. Accessible maps for the blind: Comparing 3d printed models with tactile graphics. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–13, 2018.
- [HOTS+18] José R Hilera, Salvador Otón, Cristian Timbi-Sisalima, Juan Aguado-Delgado, Francisco J Estrada-Martínez, and Héctor R Amado-Salvatierra. Combining multiple web accessibility evaluation reports using semantic web technologies. In *Advances in Information Systems Development*, pages 65–78. Springer, 2018.
- [HR13] Vicki L Hanson and John T Richards. Progress on website accessibility? *ACM Transactions on the Web (TWEB)*, 7(1):1–30, 2013.
- [HRS08] Vicki L Hanson, John T Richards, and Cal Swart. Browser augmentation. In Simon Harper and Yeliz Yesilada, editors, *Web Accessibility: a foundation for research*, Human-



- Computer Interaction Series, pages 215–229. Springer, London, 1st edition, 2008.
- [HS88] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.
- [HZ14] Alexander Henka and Gottfried Zimmermann. Persona based accessibility testing. In *International Conference on Human-Computer Interaction*, pages 226–231. Springer, 2014.
- [IIU20] Umit Ilhan, Erkut Inan Iseri, and Kaan Uyar. Web accessibility of e-government portals and ministry websites of the cyprus island. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–6. IEEE, 2020.
- [IK19] Abid Ismail and KS Kuppusamy. Web accessibility investigation and identification of major issues of higher education websites with statistical measures: A case study of college websites. *Journal of King Saud University-Computer and Information Sciences*, 2019.
- [IRY19] Yavuz Inal, Kerem Rızvanoğlu, and Yeliz Yesilada. Web accessibility in turkey: awareness, understanding and practices of user experience professionals. *Universal Access in the Information Society*, 18(2):387–398, 2019.
- [ISA<sup>+</sup>16] Wan Abdul Rahim Wan Mohd Isa, Ahmad Iqbal Hakim Suhaimi, Nadhirah Ariffn, Nurul Fatimah Ishak, and Nadilah Mohd Ralim. Accessibility evaluation using web

- content accessibility guidelines (wcag) 2.0. In *2016 4th International Conference on User Science and Engineering (i-USEr)*, pages 1–4. IEEE, 2016.
- [ISO08] ISO. Iso 32000-1:2008 - document management — portable document format — part 1: PDF 1.7. standard, International Standard Organization (ISO), 2008. Reviewed and confirmed in 2020. Last accessed on December 06, 2022. URL: <https://www.iso.org/standard/51502.html>.
- [ISO14] ISO. ISO 14289-1:2014 - document management applications — electronic document file format enhancement for accessibility — part 1: Use of ISO 32000-1 (PDF/UA-1). standard, International Standard Organization (ISO), 2014. Reviewed and confirmed in 2020. Last accessed on December 06, 2022. URL: <https://www.iso.org/standard/64599.html>.
- [ISO18] ISO. ISO 9241-11:2018 ergonomics of human-system interaction — part 11: Usability: Definitions and concepts. standard, International Standard Organization (ISO), March 2018. Last accessed on December 06, 2022. URL: <https://www.iso.org/standard/63500.html>.
- [ISS11] WARWM Isa, Muhammad Rashideen Suhami, Noor Ilyani Safie, and Siti Suhada Semsudin. Assessing the usability and accessibility of malaysia e-government website. *American Journal of Economics and Business Administration*, 3(1):40–46, 2011. doi:10.3844/ajebasp.2011.40.46.
- [Ita04] Italian Parliament. Law nr. 4 – 01/09/2004 the stanca act, January 2004. URL: <http://www.camera.it/parlam/leggi/040041.htm>.

- [İÜİ17] Erkut İ İşeri, Kaan Uyar, and Ümit İlhan. The accessibility of cyprus islands' higher education institution websites. *Procedia computer science*, 120:967–974, 2017.
- [Iva10] Rosen Ivanov. Indoor navigation system for visually impaired. In *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, pages 143–149. ACM, 2010.
- [JBA14] Hanen Jabnoun, Faouzi Benzarti, and Hamid Amiri. Object recognition for blind people based on features extraction. In *International Image Processing, Applications and Systems Conference*, pages 1–6. IEEE, 2014.
- [JWW19] Watthanasak Jeamwatthanachai, Mike Wald, and Gary Wills. Indoor navigation by blind people: Behaviors and challenges in unfamiliar spaces and buildings. *British Journal of Visual Impairment*, 37(2):140–153, 2019.
- [KB21] KS Kuppusamy and V Balaji. Evaluating web accessibility of educational institutions websites using a variable magnitude approach. *Universal Access in the Information Society*, pages 1–10, 2021.
- [KD14] Arvinder Kaur and Diksha Dani. Banking websites in india: an accessibility evaluation. *CSI transactions on ICT*, 2(1):23–34, 2014.
- [KEG21] Brian Kelly and Yasmine El-Glaly. Introducing accessibility to high school students. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 163–169, 2021.

- [Kha22] Khan Academy. `tota1ly` – an accessibility visualization toolkit, 2022. Last accessed on December 06, 2022. URL: <https://khan.github.io/tota1ly/>.
- [KK16] Nishtha Kesswani and Sanjay Kumar. Accessibility analysis of websites of educational institutions. *Perspectives in Science*, 8:210–212, 2016.
- [KK22] Nishtha Kesswani and Sanjay Kumar. Government website accessibility: A cross-country analysis of g7 and brics countries. *Universal Access in the Information Society*, 21(3):609–624, 2022.
- [KKN<sup>+</sup>22] Matt King, JaEun Jemma Ku, James Nurthen, Zoë Bijl, and Michael Cooper. Web content accessibility guidelines (WCAG) 2.1. W3C 19 may 2022, World Wide Web Consortium (W3C), May 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2022/NOTE-wai-aria-practices-1.1-20220519/>.
- [KL17] Hyun-Young Kim and Soon-Bum Lim. Accessibility automatic inspector library for epub and its components. *Journal of Korea Multimedia Society*, 20(2):330–335, 2017.
- [KMC10] Hana Kopackova, Karel Michalek, and Karel Cejna. Accessibility and findability of local e-government websites in the czech republic. *Universal access in the information society*, 9(1):51–61, 2010.
- [KOBKA18] Hernisa Kacorri, Eshed Ohn-Bar, Kris M Kitani, and Chieko Asakawa. Environmental factors in indoor navigation based on real-world trajectories of blind users. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 56. ACM, 2018.

- [KPL17] Hyun-Young Kim, Joo-Hyun Park, and Soon-Bum Lim. Automatic and semi-automatic 2-tier check system for epub accessibility. *The Journal on Technology and Persons with Disabilities*, page 29, 2017.
- [KSS22] Bineeth Kuriakose, Raju Shrestha, and Frode Eika Sandnes. Tools and technologies for blind and visually impaired navigation support: a review. *IETE Technical Review*, 39(1):3–18, 2022.
- [KTTA12] Christos Katsanos, Nikolaos Tselios, Athanasios Tsakoumis, and Nikolaos Avouris. Learning about web accessibility: A project based tool-mediated approach. *Education and Information technologies*, 17(1):79–94, 2012.
- [Kuz10] Joanne M Kuzma. Accessibility design issues with uk e-government sites. *Government information quarterly*, 27(2):141–146, 2010.
- [KWAAK16] Israa Wahbi Kamal, Heider A Wahsheh, Izzat M Alsmadi, and Mohammed N Al-Kabi. Evaluating web accessibility metrics for jordanian universities. *International Journal of Advanced Computer Science and Applications*, 7(7), 2016.
- [KY16] Bridgett A King and Norman E Youngblood. E-government in alabama: An analysis of county voting and election website content, usability, accessibility, and mobile readiness. *Government Information Quarterly*, 33(4):715–726, 2016.
- [LDAM16] Pedro Lorca, Javier De Andrés, and Ana Belén Martínez. Does web accessibility differ among banks? *World Wide Web*, 19(3):351–373, 2016.

- [Lew91] James R Lewis. Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the asq. *ACM Sigchi Bulletin*, 23(1):78–81, 1991.
- [LLPK22] Merja Laamanen, Tarja Ladonlahti, Hannu Puupponen, and Tommi Kärkkäinen. Does the law matter? an empirical study on the accessibility of finnish higher education institutions' web pages. *Universal Access in the Information Society*, pages 1–17, 2022.
- [LM10] Mohd Hanapi Abdul Latif and Mohamad Noorman Masrek. Accessibility evaluation on malaysian e-government websites. *Journal of E-Government studies and best practices*, 2010:11, 2010.
- [LMNP14] Sergio Luján-Mora, Rosa Navarrete, and Myriam Peñafiel. Egovernment and web accessibility in south america. In *2014 First International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 77–82. IEEE, 2014.
- [LRV22] Chantal Lengua, Vincenzo Rubano, and Fabio Vitali. Aligning accessibility design to non-disabled people's perceptions. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2022.
- [LS14] Eleanor Leist and Dan Smith. Accessibility issues in e-government. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 15–25. Springer, 2014.
- [LSFE21] Manoel Victor Rodrigues Leite, Lilian Passos Scatalon, André Pimenta Freire, and Marcelo Medeiros Eler. Accessibility in the mobile development industry in brazil:

- Awareness, knowledge, adoption, motivations and barriers. *Journal of Systems and Software*, 177:110942, 2021.
- [LUM13] James R Lewis, Brian S Utesch, and Deborah E Maher. Umux-lite: when there's no time for the sus. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2099–2102, 2013.
- [LW11] J Lazar and B Wentz. Separate but unequal: Web interfaces for people with disabilities. *User Experience*, 10(3):2011–3, 2011.
- [LXX+22] Junyu Lu, Xiao Xiao, Zixuan Xu, Chenqi Wang, Meixuan Zhang, and Yang Zhou. The potential of virtual tourism in the recovery of tourism industry during the covid-19 pandemic. *Current Issues in Tourism*, 25(3):441–457, 2022.
- [Mac97] Ron Mace. What is universal design. *The Center for Universal Design at North Carolina State University*, 19:2004, 1997.
- [MBK22] Jerzy Montusiewicz, Marcin Barszcz, and Sylwester Korga. Preparation of 3d models of cultural heritage objects to be recognised by touch by the blind—case studies. *Applied Sciences*, 12(23):11910, 2022.
- [MBMS22] Giuseppe Melfi, Jean Baumgarten, Karin Müller, and Rainer Stiefelhagen. An audio-tactile system for visually impaired people to explore indoor maps. In *International Conference on Computers Helping People with Special Needs*, pages 134–142. Springer, 2022.
- [MC17] RICARDO MELO and MIGUEL CARVALHAIS. Patterns for serendipity in interaction design. In *Proceedings of the*

- 5th Conference on Computation, Communication, Aesthetics & X (XCoAX)*, 2017.
- [MCB21] Alicia M Mason, Josh Compton, and Sakshi Bhati. Disabilities and the Digital Divide: Assessing Web Accessibility, Readability, and Mobility of Popular Health Websites. *Journal of Health Communication*, pages 1–8, 2021.
- [MCCMFP21] Carlos Máñez-Carvajal, Jose Francisco Cervera-Mérida, and Rocío Fernández-Piqueras. Web accessibility evaluation of top-ranking university web sites in spain, chile and mexico. *Universal Access in the Information Society*, 20(1):179–184, 2021.
- [MDAG14] Ana B Martínez, Javier De Andrés, and Julita García. Determinants of the web accessibility of european banks. *Information Processing & Management*, 50(1):69–86, 2014.
- [Met22] Meta. React: A declarative, efficient, and flexible javascript library for building user interfaces, 2022. Last accessed on December 06, 2022. URL: <https://github.com/facebook/react>.
- [MH11] Grace Mbipom and Simon Harper. The interplay between web aesthetics and accessibility. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, pages 147–154, 2011.
- [Mic22] Microsoft Corporation, INC. Accessibility insights, 2022. Last accessed on December 06, 2022. URL: <https://github.com/microsoft/accessibility-insights-web>.
- [MMJS22] Giuseppe Melfi, Karin Müller, Gerhard Jaworek, and Rainer Stiefelhagen. The accessible tactile indoor maps (atim) symbol set: a common symbol set for different



- printing methods. In *International Conference on Computers Helping People with Special Needs*, pages 153–159. Springer, 2022.
- [MP22] Şevval Seray MACAKOĞLU and Serhat Peker. Web accessibility performance analysis using web content accessibility guidelines and automated tools: a systematic literature review. In *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–8. IEEE, 2022.
- [MRCG10] Adriana Martín, Gustavo Rossi, Alejandra Cechich, and Silvia Gordillo. Engineering accessible web applications. an aspect-oriented approach. *World Wide Web*, 13(4):419–440, 2010.
- [Mrd22] Mrdoob. Three.JS: JavaScript 3D library, December 2022. Last accessed on December 06, 2022. URL: <https://github.com/mrdoob/three.js/>.
- [MS18] Apostolos Meliones and Demetrios Sampson. Blind museumtourer: A system for self-guided tours in museums and blind indoor navigation. *Technologies*, 6(1):4, 2018.
- [MSW11] Mei Miao, Martin Spindler, and Gerhard Weber. Requirements of indoor navigation system from blind users. In *Symposium of the Austrian HCI and Usability Engineering Group*, pages 673–679. Springer, 2011.
- [MZ10] Shane May and Qi Zhu. A web accessibility assessment on the texas public school system. *Universal Access in the Information Society*, 9(1):87–96, 2010.

- [Nac21] Radka Nacheva. Digital inclusion through sustainable web accessibility. In *International Conference on Digital Transformation and Global Society*, pages 83–96. Springer, 2021.
- [Ng17] Cynthia Ng. A practical guide to improving web accessibility. *Weave: Journal of Library User Experience*, 1(7), 2017.
- [NL22] Telcia Niom and Frank Lin. Accessibility of covid-19 websites of asian countries: An evaluation using automated tools. *SN Computer Science*, 3(6):1–6, 2022.
- [NLM15] R Navarrete and S Luján-Mora. Evaluating accessibility of open educational resources websites with a heuristic method. In *INTED2015 Proceedings*, pages 6402–6412. IATED, 2015.
- [NLM18] Rosa Navarrete and Sergio Luján-Mora. Bridging the accessibility gap in open educational resources. *Universal Access in the Information Society*, 17(4):755–774, 2018.
- [NNKKT19] Joyce Nakatumba-Nabende, Benjamin Kanagwa, Florence Nameere Kivunike, and Michael Tuape. Evaluation of accessibility standards on ugandan e-government websites. *Electronic Government, An International Journal*, 15(4):355–371, 2019.
- [NPM22] NPM, Inc. Npm, December 2022. Last accessed on December 06, 2022. URL: <https://www.npmjs.com/>.
- [NRRK14] Moritz Neumüller, Andreas Reichinger, Florian Rist, and Christian Kern. 3d printing for cultural heritage: Preservation, accessibility, research and education. In *3D research challenges in cultural heritage*, pages 119–134. Springer, 2014.

- [NSS17] Humaira Nazar, M Shahzad Sarfraz, and Umar Shoaib. Web accessibility evaluation of banking website in pakistan. *International Journal of Computer Science and Information Security*, 15(1):642, 2017.
- [OCH18] Heather L. O’Brien, Paul Cairns, and Mark Hall. A practical approach to measuring user engagement with the refined user engagement scale (ues) and new ues short form. *International Journal of Human-Computer Studies*, 112:28–39, 2018. URL: <https://www.sciencedirect.com/science/article/pii/S1071581918300041>, doi:<https://doi.org/10.1016/j.ijhcs.2018.01.004>.
- [OOCA11] Busra Ozdenizci, Kerem Ok, Vedat Coskun, and Mehmet N Aydin. Development of an indoor navigation system using nfc technology. In *2011 Fourth International Conference on Information and Computing*, pages 11–14. IEEE, 2011.
- [OSAB19] Ishaq O Oyefolahan, Aishat A Sule, Solomon A Adepoju, and Faiza Babakano. Keeping with the global trends: An evaluation of accessibility and usability of nigerian banks websites. *International Journal of Information Engineering and Electronic Business*, 2019.
- [PAK<sup>+</sup>17] J Eduardo Pérez, Myriam Arrue, Masatomo Kobayashi, Hironobu Takagi, and Chieko Asakawa. Assessment of semantic taxonomies for blind indoor navigation based on a shopping center use case. In *Proceedings of the 14th Web for All Conference on The Future of Accessible Work*, page 19. ACM, 2017.
- [Pau21] Pamela Paul. *100 Things We’ve Lost to the Internet*. Crown, October 2021. ISBN: 978-0593136775.

- [Pau22] Surjit Paul. Accessibility analysis using wcag 2.1: evidence from indian e-government websites. *Universal Access in the Information Society*, pages 1–7, 2022.
- [PBMZF20] Débora Maria Barroso Paiva, Marisa Helena da Silva Batista, Luciana Aparecida Martinez Zaina, and Renata Pontin de Mattos Fortes. Acc-mobilecheck: a checklist for usability and accessibility evaluation of mobile applications. *CLEI Electronic Journal*, 23(2):1–12, 2020.
- [PBS16] Neha Patil, Dhananjay Bhole, and Prasanna Shete. Enhanced ui automator viewer with improved android accessibility evaluation features. In *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pages 977–983. IEEE, 2016.
- [PC20] Subhajit Panda and Rupak Chakravarty. Evaluating the web accessibility of iit libraries: a study of web content accessibility guidelines. *Performance Measurement and Metrics*, 2020.
- [PDF21] PDF/UA Foundation. PDF Accessibility Checker (PAC) 2021 - the free PDF Accessibility Checker, 2021. Last accessed on December 06, 2022. URL: <https://pdfua.foundation/en/pdf-accessibility-checker-pac/>.
- [PDMJ09] Rémi Parlouar, Florian Dramas, Marc MJ Macé, and Christophe Jouffrais. Assistive device for the blind based on object recognition: an application to identify currency bills. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, pages 227–228, 2009.
- [PE22] Maria Perifanou and Anastasios A Economides. Analyzing repositories of oer using web analytics and accessibility

- tools. *Universal Access in the Information Society*, pages 1–15, 2022.
- [PFPS12] Christopher Power, André Freire, Helen Petrie, and David Swallow. Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 433–442, 2012.
- [PHB<sup>+</sup>19] Eunju Park, Sungjun Han, Hogon Bae, Raekyung Kim, Seungjae Lee, Daejune Lim, and Hankyu Lim. Development of automatic evaluation tool for mobile accessibility for android application. In *2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS)*, pages 1–6. IEEE, 2019.
- [PHK04] Helen Petrie, Fraser Hamilton, and Neil King. Tension, what tension? website accessibility and visual design. In *Proceedings of the 2004 international cross-disciplinary workshop on Web accessibility (W4A)*, pages 13–18, 2004.
- [Pos] PostCSS. PostCSS: Transforming styles with JS plugins. Last accessed on December 06, 2022. URL: <https://github.com/postcss/postcss>.
- [PPFS11] Christopher Power, Helen Petrie, André P Freire, and David Swallow. Remote evaluation of wcag 2.0 techniques by web users with visual disabilities. In *International Conference on Universal Access in Human-Computer Interaction*, pages 285–294. Springer, 2011.
- [PPJH19] Silvia Pfeiffer, Simon Pieters, Philip Jägenstedt, and Ian Hickson. WebVTT: The web video text tracks format. W3C candidate recommendation 04 april 2019, World

- Wide Web Consortium (W3C), April 2019. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2019/CR-webvtt1-20190404/>.
- [RCMCRG<sup>+</sup>19] Víctor Romero-Chacón, Harry Muir-Camacho, Jonnathan Rodríguez-González, Annia Gómez-Blanco, and Mario Chacón-Rivas. Adapting scrum methodology to develop accessible web sites. In *2019 International Conference on Inclusive Technologies and Education (CONTIE)*, pages 112–1124. IEEE, 2019.
- [Rel15] Luz Rello. Dyslexia and web accessibility: synergies and challenges. In *Proceedings of the 12th International Web for All Conference*, pages 1–4, 2015.
- [RFL<sup>+</sup>18] Valentina Rossetti, Francesco Furfari, Barbara Leporini, Susanna Pelagatti, and Andrea Quarta. Enabling access to cultural heritage for the visually impaired: an interactive 3d model of a cultural site. *Procedia computer science*, 130:383–391, 2018.
- [Roq22] Arnaud Roques. PlantUML: an open-source tool that uses simple textual descriptions to draw beautiful UML diagrams, 2022. Last accessed on December 06, 2022. URL: <http://plantuml.com>.
- [RPCT17] Germania Rodríguez, Jennifer Pérez, Samanta Cueva, and Rommel Torres. A framework for improving web accessibility and usability of open course ware sites. *Computers & education*, 109:197–215, 2017.
- [RPD09] Sebastien Rainville-Pitt and Jean-Marie D’amour. Using a cms to create fully accessible web sites. *Journal of access services*, 6(1-2):261–264, 2009.

- [RR15] Jordi Roig and Mireia Ribera. Creation of accessible epub documents by non-technical users: A long way to go. In *Proceedings of the XVI International Conference on Human Computer Interaction*, pages 1–2, 2015.
- [RS12] Dagfinn Rømen and Dag Svanæs. Validating wcag versions 1.0 and 2.0 through usability testing with disabled users. *Universal Access in the Information Society*, 11(4):375–385, 2012.
- [RST15] Jan Richards, Jeanne Spellman, and Jutta Treviranus. Implementing ATAG 2.0, a guide to understanding and implementing authoring tools accessibility guidelines (ATAG) 2.0. W3C working group note 24 september 2015, World Wide Web Consortium (W3C), September 2015. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2015/NOTE-IMPLEMENTING-ATAG20-20150924/>.
- [RV20] Vincenzo Rubano and Fabio Vitali. Experiences from declarative markup to improve the accessibility of HTML. In *Proceedings of Balisage: The Markup Conference*, 2020.
- [RV21] Vincenzo Rubano and Fabio Vitali. Making accessibility accessible: strategy and tools. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.
- [RVFFM14] Rosabel Roig-Vila, Sergio Ferrández, and Imma Ferri-Miralles. Assessment of web content accessibility levels in spanish official online education environments. *International Education Studies*, 7(6):31–45, 2014.
- [RVL22] Vincenzo Rubano, Fabio Vitali, and Chantal Lengua. Tools for an innovative approach to web accessibility. In *Intern-*

- tional Conference on Human-Computer Interaction*, pages 97–115. Springer, 2022.
- [RVRGF15] Johana M Rosas-Villena, Bruno Ramos, Rudinei Goularte, and Renata PM Fortes. Video accessibility on the most accessed websites—a case study regarding visual disabilities. In *International Conference on Universal Access in Human-Computer Interaction*, pages 231–241. Springer, 2015.
- [RZSZ16] Pei-Luen Patrick Rau, Lianhui Zhou, Na Sun, and Runting Zhong. Evaluation of web accessibility in china: changes from 2009 to 2013. *Universal Access in the Information Society*, 15(2):297–303, 2016.
- [SAB<sup>+</sup>20] Mohamad Zaidi Sulaiman, Mohd Nasiruddin Abdul Aziz, Mohd Haidar Abu Bakar, Nur Akma Halili, and Muhammad Asri Azuddin. Matterport: virtual tour as a new marketing approach in real estate business during pandemic covid-19. In *International Conference of Innovation in Media and Visual Design (IMDES 2020)*, pages 221–226. Atlantis Press, 2020.
- [SAL<sup>+</sup>21] Navid Salehnamadi, Abdulaziz Alshayban, Jun-Wei Lin, Iftekhar Ahmed, Stacy Branham, and Sam Malek. Latte: Use-case and assistive-service driven automated accessibility testing framework for android. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2021.
- [San21] Frode Eika Sandnes. Inverse color contrast checker: Automatically suggesting color adjustments that meet contrast requirements on the web. In *The 23rd International ACM*



- SIGACCESS Conference on Computers and Accessibility*, pages 1–4, 2021.
- [SAPV19] Aritz Sala, Myriam Arrue, J Eduardo Pérez, and Xabier Valencia. Accessibility-in-use of public e-services: An exploratory study including users with low vision. In *Proceedings of the XX International Conference on Human Computer Interaction*, pages 1–4, 2019.
- [SBA19] Irum Naz Sodhar, Hina Bhanbhro, and Zaira Hassan Amur. Evaluation of web accessibility of engineering university websites of pakistan through online tools. *IJCSNS*, 19(12):85–90, 2019.
- [SBB<sup>+</sup>13] Caterina Senette, Maria Claudia Buzzi, Marina Buzzi, Barbara Leporini, and Loredana Martusciello. Enriching graphic maps to enable multimodal interaction by blind people. In *International Conference on Universal Access in Human-Computer Interaction*, pages 576–583. Springer, 2013.
- [SCF<sup>+</sup>15] Leandro Coelho Serra, Lucas Pedroso Carvalho, Lucas Pereira Ferreira, Jorge Belimar Silva Vaz, and André Pimenta Freire. Accessibility evaluation of e-government mobile applications in brazil. *Procedia Computer Science*, 67:348–357, 2015.
- [Sch97] Ken Schwaber. Scrum development process. In *Business object design and implementation*, pages 117–134. Springer, 1997.
- [SCSWL14] Joseph Scheuhammer, Michael Cooper, Andi Snow-Weaver, and Aaron Leventhal. WAI-ARIA 1.0 user agent implementation guide. a user agent developer’s guide to

- understanding and implementing accessible rich internet applications. W3C recommendation 20 march 2014, World Wide Web Consortium (W3C), March 2014. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2014/REC-wai-aria-implementation-20140320/>.
- [SE07] Jaime Sánchez and Miguel Elías. Guidelines for designing mobility and orientation software for blind children. In *IFIP Conference on Human-Computer Interaction*, pages 375–388. Springer, 2007.
- [SE12] Bernhard Schmitz and Thomas Ertl. Interactively displaying maps on a tactile graphics display. *SKALID 2012–Spatial Knowledge Acquisition with Limited Information Displays (2012)*, pages 13–18, 2012.
- [SGLM17] Sandra Sanchez-Gordon and Sergio Luján-Mora. A method for accessibility testing of web applications in agile environments. In *Proceedings of the 7th World Congress for Software Quality (WCSQ). En proceso de publicación. (citado en la página 13, 15, 85)*, 2017.
- [SGLMSG20] Sandra Sanchez-Gordon, Sergio Lujan-Mora, and Mary Sanchez-Gordon. E-government accessibility in ecuador: a preliminary evaluation. In *2020 Seventh International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 50–57. IEEE, 2020.
- [Sha15] Bayan Abu Shawar. Evaluating web accessibility of educational websites. *International Journal of Emerging Technologies in Learning*, 10(4), 2015.
- [SK17] Osama Sohaib and Kyeong Kang. E-commerce web accessibility for people with disabilities. In *Complexity in In-*

- formation Systems Development*, pages 87–100. Springer, 2017.
- [SL18] Natalie L Shaheen and Jonathan Lazar. K–12 technology accessibility: The message from state governments. *Journal of Special Education Technology*, 33(2):83–97, 2018.
- [SLOM19] Cecilia Sik-Lanyi and Éva Orbán-Mihálykó. Accessibility Testing of European Health-Related Websites. *Arabian Journal for Science and Engineering*, 44(11):9171–9190, 2019.
- [SLSSW20] Julian Striegl, Claudia Lotisch, Jan Schmalfuss-Schwarz, and Gerhard Weber. Analysis of indoor maps accounting the needs of people with impairments. In *International Conference on Computers Helping People with Special Needs*, pages 305–314. Springer, 2020.
- [SPCM16] Andreia Santos, Yves Punie, and Jonatan Castaño Muñoz. Opening up education: A support framework for higher education institutions. Technical report, Joint Research Centre (Seville site), 2016.
- [Spi12] Diomidis Spinellis. Git. *IEEE software*, 29(3):100–101, 2012.
- [Spi19] Carli Spina. Wcag 2.1 and the current state of web accessibility in libraries. *Weave: Journal of Library User Experience*, 2(2), 2019.
- [SPN+18] Tetiana Shestakevych, Volodymyr Pasichnyk, Maria Nazaruk, Mykola Medykovskiy, and Natalya Antonyuk. Web-products, actual for inclusive school graduates: evaluating the accessibility. In *Conference on Computer Science*

- and Information Technologies*, pages 350–363. Springer, 2018.
- [Sql22] Sqlite.org. SQLite FTS5 Extension, September 2022. Last accessed on December 06, 2022. URL: <https://www.sqlite.org/fts5.html>.
- [SS11] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 21(19):1, 2011.
- [SSS16] Sven Schmutz, Andreas Sonderegger, and Juergen Sauer. Implementing recommendations from web accessibility guidelines: Would they also provide benefits to nondisabled users. *Human factors*, 58(4):611–629, 2016.
- [SSS17] Sven Schmutz, Andreas Sonderegger, and Juergen Sauer. Implementing recommendations from web accessibility guidelines: a comparative study of nondisabled users and users with visual impairments. *Human factors*, 59(6):956–972, 2017.
- [SSS18] Sven Schmutz, Andreas Sonderegger, and Juergen Sauer. Effects of accessible website design on nondisabled users: age and device as moderating factors. *Ergonomics*, 61(5):697–709, 2018.
- [SSS20] Juergen Sauer, Andreas Sonderegger, and Sven Schmutz. Usability, user experience and accessibility: towards an integrative model. *Ergonomics*, 63(10):1207–1220, 2020.
- [SW16] J Smith and J Whiting. WAVE – web accessibility evaluation tool, 2016. Last accessed on December 06, 2022. URL: <http://wave.webaim.org>.

- [TA17] Morten Tollefsen and Trond Ausland. A practitioner’s approach to using wcag evaluation tools. In *2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA)*, pages 1–5. IEEE, 2017.
- [TAAS16] Yahya M Tashtoush, Darabseh Ala’F, and Huda N Al-Sarhan. The arabian e-government websites accessibility: a case study. In *2016 7th International Conference on Information and Communication Systems (ICICS)*, pages 276–281. IEEE, 2016.
- [Taw22] Tawdist. TAW | W3C and web accessibility standardization services, 2022. Last accessed on December 06, 2022. URL: <https://www.tawdis.net/>.
- [TDR11] Eduardo Hideki Tanaka and Heloísa Vieira Da Rocha. Evaluation of web accessibility tools. In *Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction*, pages 272–279, 2011.
- [Ten17] Tenon LLC. tenon.io, 2017. Last accessed on December 06, 2022. URL: <https://tenon.io>.
- [TFA17] Garreth W Tigwell, David R Flatla, and Neil D Archibald. Ace: a colour palette design tool for balancing aesthetics and accessibility. *ACM Transactions on Accessible Computing (TACCESS)*, 9(2):1–32, 2017.
- [TLZX12] Brandon Taylor, Dah-Jye Lee, Dong Zhang, and Guangming Xiong. Smart phone-based indoor guidance system for the visually impaired. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 871–876. IEEE, 2012.

- [TPG22a] TPGi. Arc toolkit, 2022. Last accessed on December 06, 2022. URL: <https://www.tpgi.com/arc-platform/arc-toolkit/>.
- [TPG22b] TPGi. TPGi - Your Digital Accessibility Solutions Partner!, December 2022. Last accessed on December 08, 2022. URL: <https://www.tpgi.com/>.
- [TRW18] Haileslasie Tadele, Helen Roberts, and Rosalind H Whiting. Microfinance institutions' website accessibility. *Pacific-Basin Finance Journal*, 50:279–293, 2018.
- [TTHZ16] Hao Tang, Norbu Tsering, Feng Hu, and Zhigang Zhu. Automatic pre-journey indoor map generation using autocad floor plan. *J Tec Pers Disabil*, 4:176–191, 2016.
- [TWB22] TWBS. Bootstrap: The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web, December 2022. Last accessed on December 06, 2022. URL: <https://github.com/twbs/bootstrap>.
- [Uni06a] United Nations. Convention on the rights of persons with disabilities (CRPD) – articles, December 2006. Last accessed on December 06, 2022. URL: <https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities/convention-on-the-rights-of-persons-with-disabilities-2.html>.
- [Uni06b] United Nations. Convention on the rights of persons with disabilities (CRPD) – optional protocol, December 2006. Last accessed on December 06, 2022. URL: <https://www.un.org/development/>

- desa/disabilities/convention-on-the-rights-of-persons-with-disabilities/optional-protocol-to-the-convention-on-the-rights-of-persons-with-disabilities.html.
- [U.S94] U.S. Congress. Architectural Barriers Act of 1968. Public Law 90-480, as Amended Through P.L. 103-437 of 1994, 1994. Last accessed on December 06, 2022. URL: <https://www.govinfo.gov/content/pkg/COMPS-10651/pdf/COMPS-10651.pdf>.
- [U.S08a] U.S. Congress. ADA Amendments Act of 2008, September 2008. Last accessed on December 06, 2022. URL: <http://www.camera.it/parlam/leggi/040041.htm>.
- [U.S08b] U.S. Congress. Americans with disabilities act (ADA) of 1990, as amended, 2008. Last accessed on December 06, 2022. URL: <https://www.ada.gov/law-and-regs/ada/>.
- [U.S15] U.S. Congress. The Rehabilitation Act of 1973 - as amended through P.L. 114-95 (WIOA), 2015. Last accessed on December 06, 2022. URL: <https://assets.section508.gov/files/rehabilitation-act-of-1973-amended-by-wioa.pdf>.
- [Var18] Various Authors. Game accessibility guidelines (GAG). living document, [gameaccessibilityguidelines.com](http://gameaccessibilityguidelines.com), December 2018. Last accessed on December 06, 2022. URL: <https://gameaccessibilityguidelines.com/full-list/>.
- [VAZ14] Eric Velleman and Shadi Abou-Zahra. Website accessibility conformance evaluation methodology (WCAG-EM) 1.0. W3C working group note 10 july 2014, World Wide Web

- Consortium (W3C), July 2014. Last accessed on December 06, 2022. URL: <http://www.w3.org/TR/2014/NOTE-WCAG-EM-20140710/>.
- [VAZK17] Carlos A Velasco, Shadi Abou-Zahra, and Johannes Koch. Developer guide for evaluation and report language (EARL) 1.0. W3C working group note 2 february 2017, World Wide Web Consortium (W3C), February 2017. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/2017/NOTE-EARL10-Guide-20170202/>.
- [VBC13] Markel Vigo, Justin Brown, and Vivienne Conway. Benchmarking web accessibility evaluation tools: measuring the harm of sole reliance on automated tests. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, pages 1–10, 2013.
- [VDW17] Silas Formunyuy Verkijika and Lizette De Wet. Determining the accessibility of e-government websites in sub-saharan africa against wcag 2.0 standard. *International Journal of Electronic Government Research (IJEGR)*, 13(1):52–68, 2017.
- [VF22] Stefano Valtolina and Daniele Fratus. Local government websites accessibility: Evaluation and finding from italy. *Digital Government: Research and Practice*, 3(3):1–16, 2022.
- [VFC20] Roberto Vaz, Diamantino Freitas, and António Coelho. Blind and visually impaired visitors’ experiences in museums: Increasing accessibility through assistive technologies. *International Journal of the Inclusive Museum*, 13(2), 2020.



- [VH14] Markel Vigo and Simon Harper. A snapshot of the first encounters of visually disabled users with the web. *Computers in Human Behavior*, 34:203–212, 2014.
- [VIB<sup>+</sup>19] Beat Vollenwyder, Glena H Iten, Florian Brühlmann, Klaus Opwis, and Elisa D Mekler. Salient beliefs influencing the intention to consider web accessibility. *Computers in Human Behavior*, 92:352–360, 2019.
- [VNvdG17] Eric M Velleman, Inge Nahuis, and Thea van der Geest. Factors explaining adoption and implementation processes for web accessibility standards within egovernment systems and organizations. *Universal access in the information society*, 16(1):173–190, 2017.
- [VPI<sup>+</sup>23] Beat Vollenwyder, Serge Petralito, Glena H Iten, Florian Brühlmann, Klaus Opwis, and Elisa D Mekler. How compliance with web accessibility standards shapes the experiences of users with and without disabilities. *International Journal of Human-Computer Studies*, 170:102956, 2023.
- [VSLLV19] Christopher Vendome, Diana Solano, Santiago Liñán, and Mario Linares-Vásquez. Can everyone use my app? an empirical study on accessibility in android apps. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 41–52. IEEE, 2019. doi: 10.1109/ICSME.2019.00014.
- [Vue22] Vue.JS. Vue: a progressive, incrementally-adoptable javascript framework for building ui on the web, 2022. Last accessed on December 06, 2022. URL: <https://github.com/vuejs/vue>.
- [Wan12] Ye Diana Wang. A holistic and pragmatic approach to teaching web accessibility in an undergraduate web design

- course. In *Proceedings of the 13th annual conference on Information technology education*, pages 55–60, 2012.
- [Web16] WebAIM (Web Accessibility In Mind). Contrast checker, 2016. Last accessed on December 06, 2022. URL: <https://webaim.org/resources/contrastchecker/>.
- [Web22a] Web AIM - Web Accessibility in Mind. The WebAIM million - the 2022 report on the accessibility of the top 1,000,000 home pages, February 2022. Archived in December 2022. URL: <https://web.archive.org/web/20221226040727/https://webaim.org/projects/million/>.
- [Web22b] Web AIM - Web Accessibility in Mind. WebAIM |Homepage, December 2022. Last accessed on December 08, 2022. URL: <https://webaim.org>.
- [Web22c] Web Hypertext Application Technology Working Group (WHATWG). *Html. living standard* — last updated 8 december 2022, Web Hypertext Application Technology Working Group (WHATWG), December 2022. Last accessed on December 08, 2022. URL: <https://html.spec.whatwg.org/>.
- [Web22d] WebAIM (Web Accessibility In Mind). link contrast checker, 2022. Last accessed on December 06, 2022. URL: <https://webaim.org/resources/linkcontrastchecker/>.
- [WLHH09] Zheshen Wang, Baoxin Li, Terri Hedgpeth, and Teresa Haven. Instant tactile-audio map: enabling access to digital maps for people with visual impairment. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, pages 43–50, 2009.

- [Wora] World Wide Web Consortium (W3C). Documents published at W3C. Last accessed on December 06, 2022. URL: <https://www.w3.org/standards/types>.
- [Worb] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). All accessibility conformance testing (ACT) rules. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/standards-guidelines/act/rules/>.
- [Worc] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). ATAG 2.0 at a glance. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/standards-guidelines/atag/glance/>.
- [Word] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). How to meet WCAG (quick reference). a customizable quick reference to web content accessibility guidelines (WCAG) 2 requirements (success criteria) and techniques. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/WCAG21/quickref/>.
- [Wore] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). WCAG 2.1 at a glance. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/standards-guidelines/wcag/glance/>.
- [Wor22a] World Wide Web Consortium (W3C). ARIA authoring practices guide (APG). guide for developers, World Wide Web Consortium (W3C), 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/ARIA/apg/>.
- [Wor22b] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). ATAG report tool, 2022. Last accessed

- on December 06, 2022. URL: <https://www.w3.org/WAI/atag/report-tool/>.
- [Wor22c] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). Selecting web accessibility evaluation tools, 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/test-evaluate/tools/selecting/>.
- [Wor22d] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). Using ARIA, 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/TR/using-aria/>.
- [Wor22e] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). WCAG-EM report tool, 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/eval/report-tool/>.
- [Wor22f] World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI). Web accessibility evaluation tools list, 2022. Last accessed on December 06, 2022. URL: <https://www.w3.org/WAI/ER/tools/>.
- [WPF<sup>+</sup>19] Brian Wentz, Dung Pham, Erin Feaser, Dylan Smith, James Smith, and Allison Wilson. Documenting the accessibility of 100 us bank and finance websites. *Universal Access in the Information Society*, 18(4):871–880, 2019.
- [YBVH15] Yeliz Yesilada, Giorgio Brajnik, Markel Vigo, and Simon Harper. Exploring perceptions of web accessibility: a survey approach. *Behaviour & Information Technology*, 34(2):119–134, 2015.

- [YC15] Y Tony Yang and Brian Chen. Web accessibility for older adults: A comparative analysis of disability laws. *The Gerontologist*, 55(5):854–864, 2015.
- [YLR<sup>+</sup>19] Chris Yoon, Ryan Louie, Jeremy Ryan, MinhKhang Vu, Hyegi Bang, William Derksen, and Paul Ruvolo. Leveraging augmented reality to create apps for people with visual disabilities: A case study in indoor navigation. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, pages 210–221, 2019.
- [YR19] Shunguo Yan and PG Ramachandran. The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing (TACCESS)*, 12(1):1–31, 2019.
- [Yu21] Sarah Yanyue Yu. A review of the accessibility of act covid-19 information portals. *Technology in Society*, 64:101467, 2021.
- [Zel19] Denys Zelenchuk. Accessibility testing. In *Android Espresso Revealed*, pages 157–164. Springer, 2019.
- [ZTN<sup>+</sup>20] Xiangling Zhang, Ahmed Tlili, Fabio Nascimbeni, Daniel Burgos, Ronghuai Huang, Ting-Wen Chang, Mohamed Jemni, and Mohamed Koutheair Khribi. Accessibility within open educational resources and practices for disabled learners: A systematic literature review. *Smart Learning Environments*, 7(1):1–19, 2020.



# Glossary

**CSS** Cascading StyleSheets.

**HTML** HyperText Markup Language.

**JSON** JavaScript Object Notation.

**NPM** Node Package Manager.

**SAHARIAN** Sighted Architects Helper for Accessibility Notation.

**SVG** Scalar Vector Graphics.

**UML** Unified Modeling Language.

**XML** Extensible Markup Language.

**XSD** XML Schema Definition.

**YAML** Ain't Markup Language.

**A11YVT** Accessibility for Virtual Tours.

**ABA** Architectural Barriers Act.

**ACE** Accessible Colour Evaluator.

**ADA** Americans with Disabilities Act.

**ADHD** attention deficit hyperactivity disorder.

**AE** Aesthetic appeal.

- AEA** Accessibility Evaluation Assistant.
- AJAX** Asynchronous JavaScript and XML.
- AODA** Accessibility for Ontarians with Disabilities Act.
- APG** WAI-ARIA Practices Guide.
- API** Application Programming Interface.
- AR** Augmented Reality.
- ASQ** After-Scenario Questionnaire.
- AT** Assistive Technologies.
- ATAG** Authoring Tools Accessibility Guidelines.
- BLE** Bluetooth Low Energy.
- BLS** Basic Life Support.
- BRICS** Brazil-Russia-India-China-South Africa.
- CI** Continuous Integration.
- CMS** Content Management System.
- CPI** Corruption Perception Index.
- CRPD** Convention on Rights of Persons with Disabilities.
- DAISY** the Digital Accessible Information System.
- DDA** Disability Discrimination Act.
- DIY** do it yourself.
- DOM** Document Object Model.



**E-GOV** Electronic Government.

**EAA** European Accessibility Act.

**EARL** Evaluation and Report Language.

**EDS** European Disability Strategy.

**EOL** End of Life.

**EPUB** Electronic Publication.

**EU** European Union.

**FA** Focused Attention.

**FOV** Field of View.

**GAG** Game Accessibility Guidelines.

**GDPR** General Data Protection Regulation.

**GPS** Global Positioning System.

**HDI** Human Development Index.

**ICT** Information and Communication Technologies.

**IDRC** Inclusive Design Research Centre.

**ISO** International Standard Organization.

**Jamstack** JavaScript APIs and Markup.

**LED** light-emitting diode.

**MARE** the Missing WAI-ARIA Role Explorer.

**MathML** Mathematical Markup Language.

**MOOC** massive open online courses.

**MWE** Minimal Working Examples.

**NFC** Near Field Communication.

**O&M** Orientation and Mobility.

**OCW** Open Course Ware.

**OER** Open Educational Resources.

**PAC** PDF Accessibility Checker.

**PAVE** PDF Accessibility Validation Engine.

**PDF** Portable Document Format.

**POI** Points of Interest.

**PU** Perceived usability.

**RDF** Resource Description Framework.

**RFID** Radio Frequency Identification.

**RSS** Really Simple Syndication.

**RW** Reward factor.

**SCAMP** Scientific Collections Accessibility Making Process.

**SMIL** Synchronized Multimedia Integration Language.

**SPA** Single Page Application.

**SSA** Sub-Saharan Africa.

**SSG** Static Site Generator.

**SSR** Server Side Rendering.

**SUS** System Usability Scale.

**TDD** Test Driven Development.

**TTML2** Timed Text Markup Language 2.

**U.S.** United States.

**UAAG** User Agents Accessibility Guidelines.

**UES** User Engagement Scale.

**UI** User Interface.

**UK** United Kingdom.

**UMUX** Usability Metric for User Experience.

**UNCRPD** United Nations' Convention on Rights of Persons with Disabilities.

**UX** User Experience.

**W3C** World Wide Web Consortium.

**WAI** WWeb Accessibility Initiative.

**WAI-ARIA** Active Rich Internet Applications.

**WAVE** Web Accessibility Evaluation Tools.

**WCAG** Web Content Accessibility Guidelines.

**WCAG-EM** Web Accessibility Conformance Evaluation Methodology.

**WebAIM** Web Accessibility in Mind.

**WebVTT** Web Video Text Tracks Format.

**WWW** World Wide Web.



# Ringraziamenti

Ed eccoci qui, dopo una lunga tesi, a scrivere quella che forse è la pagina più attesa da te, caro lettore o cara lettrice, che avrai avuto la pazienza di leggere questo intero manoscritto da oltre 300 pagine o forse sarai saltato/a direttamente a questa pagina; in ogni caso, comunque, a te il primo sentito ringraziamento per aver consultato questa tesi!

Una tesi, dicevo, piuttosto lunga ed elaborata... Così come lungo ed elaborato è stato questo mio percorso di dottorato. Di cose in questi tre anni ne sono successe davvero tante, tantissime... Alcune anche imprevedibili! Nel 2019, per esempio, chi mai avrebbe immaginato che dal 2020 ci saremmo trovati nel mezzo di una pandemia globale che avrebbe stravolto le nostre vite, ci avrebbe fatto riflettere su ciò che conta davvero, costringendoci a ripensare il nostro modo di vivere per ben due anni? Lo ricordo bene il primo lockdown: da quel gelato che uscii a prendere così per caso, quando là fuori la situazione iniziava a mettersi male ma nessuno ancora si aspettava il peggio, alle corse per fare la spesa prevedendo ciò che stava arrivando, per poi ascoltare in diretta gli annunci; e poi l'isolamento di fatto in studentato, le giornate passate a trovare nuovi modi per lavorare e trascorrere il tempo libero, le surreali sensazioni provate camminando per le strade deserte di Bologna con la consapevolezza che tutti erano in casa e ben udibili dalle loro finestre aperte... E chi lo dimentica? Penso sia stata un'esperienza che mi ha segnato profondamente, tanto che ricordo benissimo tutte le sensazioni provate ogni singolo giorno. Potrei dilungarmi all'infinito raccontando aneddoti, sensazioni, le chiacchierate su ClubHouse, le videoconferenze con gli amici

di vecchia data, ma probabilmente non è questa la sede opportuna; dovrei scrivervi un testo a sé, forse, o forse no... Chi lo sa?

Ma come si suol dire a volte anche nel male si trovano cose positive. E sì perché, per esempio, proprio a causa del ripensamento di tante attività (incluse le conferenze scientifiche) grazie alla pandemia ho potuto partecipare a tantissimi eventi che non avrei potuto frequentare altrimenti; certo, nessuno può dire come si sarebbe sviluppato questo mio percorso di dottorato senza di essa, ma la possibilità di partecipare a molteplici eventi OnLine mi ha permesso di incontrare e conoscere persone con cui difficilmente sarei riuscito a entrare in contatto altrimenti.

E proprio da due di queste persone vorrei partire con i ringraziamenti, Yeliz Yesilada e Mike Paciello: due tra i più stimati professionisti nel campo dell'accessibilità di fama e autorevolezza indubbiamente riconosciute a livello internazionale, i cui feedback mi hanno consentito di migliorare sia questo lavoro di tesi sia i manufatti prodotti durante queste ricerche; a voi dedico un immenso grazie, è stato un onore avervi come revisori esterni di questa tesi e poter ricevere i vostri commenti sul mio lavoro!

Restando in ambito accademico non posso esimermi dal ringraziare il prof. Fabio Vitali, supervisore instancabile che mi ha seguito durante tutte le attività di ricerca in questi tre anni; secondi ma non certo per importanza ringrazio anche il prof. Paolo Ciancarini e la prof.ssa Silvia Mirri, che in qualità di membri della Commissione di dottorato si sono sempre dimostrati pronti a fornire feedback utilissimi durante questi tre anni. E poi l'ex coordinatore del Dottorato prof. Davide Sangiorgi, nonché l'attuale coordinatrice prof.ssa Ilaria Bartolini che si sono sempre dimostrati sensibili a venirmi incontro per superare le difficoltà incontrate lungo il percorso: grazie mille per il vostro supporto! Allo stesso modo ringrazio tutti i docenti incontrati durante le attività di formazione previste dal corso di Dottorato, che non elenco esplicitamente semplicemente perché sono stati davvero tanti, i quali si sono dimostrati sempre estremamente disponibili a rendere più accessibili le lezioni e i corsi quando è stato necessario.

Ma si sa: siamo in Italia. Paese noto per il buon cibo, il vino, i paesaggi, e tante altre cose... Ma soprattutto la burocrazia! E dunque permettetemi anche un sentito ringraziamento a Vito, Alice, Lucia, e tutti gli altri ragazzi degli uffici del dipartimento: senza il vostro apporto tante cose fatte in questi tre anni non sarebbero state possibili!

Per un non vedente, però, muoversi autonomamente tra i vari uffici non è sempre facile; e dunque permettetemi un ringraziamento anche a Bruna, Giuseppe, Alessandra e tutte le altre persone incontrate durante gli spostamenti tra le varie sedi dell'università, sempre pronte ad accompagnarmi nei momenti del bisogno: forse non ci avete pensato, ma anche il vostro supporto è stato essenziale! Grazie mille davvero!

Permettetemi poi un ringraziamento per i nuovi colleghi del Comune di Bologna, presso cui sono entrato in servizio al termine di questo percorso di dottorato; ringrazio il dott. Stefano Mineo, dirigente del Servizio Innovazione Digitale e Dati nonché mio responsabile, ma anche Sonia, Marco, Valentina, Flavio, Tiziana, Anna, Giuseppe e tutti gli altri che non menziono semplicemente perché l'elenco sarebbe davvero troppo lungo: grazie mille per avermi supportato nell'integrarmi nell'Ente nonostante sia arrivato da pochi mesi! Permettetemi solo una menzione speciale per Elena, Elisa, Luca e Paola: non potrò mai dirvi abbastanza quanto sia stato essenziale (e lo sia ancora tutt'ora!) il vostro supporto anche al di fuori delle attività lavorative, dunque lo scrivo qui a imperitura memoria: non potrò mai dirvi grazie abbastanza volte!

Ma una tesi così lunga non potrebbe esistere senza un editor; permettetemi allora di ringraziare anche Caterina, oltre che per la sua amicizia, per il lavoro svolto e le sue correzioni che mi hanno consentito di effettuare tantissimi ritocchi alla forma di questa tesi.

Ma come recita un antico proverbio africano:

Se vuoi andare veloce, corri da solo. Se vuoi andare lontano, cammina insieme agli altri.

Non si può sottovalutare l'importanza di essere circondati da un gruppo di persone su cui poter contare; e allora permettetemi di ringraziare Alessio, Cristina, Elena, Simone, Tommaso e Vainer dello staff di NVApple (rigorosamente in ordine alfabetico) per il loro supporto indispensabile; Sue Ellen, su cui poter sempre fare affidamento; Remo e Caterina, con cui sicuramente continueremo a parlare di fiori e fagioli oltre che a divertirci assieme; e infine permettetemi di ringraziare anche Emma, un'amica un po' più speciale da oramai quattro anni e mezzo.

Ma come ha detto William Faulkner

The past is not dead. It is not even past.

Il nostro passato non è mai davvero passato, ma influenza in modo significativo il nostro presente e ne diventa parte integrante. Perciò non posso esimermi dal ringraziare chi mi ha permesso di arrivare fin qui: soprattutto i miei genitori Carmen e Alfonso e mio fratello Manuel, la mia famiglia su cui ho la certezza di poter contare per qualsiasi cosa e in qualsiasi momento; lo sapete, non sono bravo con le parole, ma stavolta permettetemi di dirvi quanto sono fortunato e di ringraziarvi per tutto ciò che avete sempre fatto, che fate e che farete! Un grazie "specialissimo" anche a chi non c'è più: Merysol, compagna di vita e di mille avventure, il cui supporto in questi anni è stato quantomai fondamentale: mille parole per descrivere quello che provo non possono bastare! E ovviamente anche una carezza a Chico, l'ultimo arrivato in famiglia. E poi non posso esimermi dal ringraziare Raffaella, Eliana, la signora Enza, le maestre, i professori e tutte le altre persone che mi hanno supportato in qualche modo durante il mio percorso di studi.

Lasciatemi poi ringraziare Er-go per avermi ospitato e supportato in questi quasi 10 anni, e in particolare Cristiana dell'ufficio disabili; tutto è certamente migliorabile, ma senza il vostro supporto oggi non sarei qui... Nel vero senso della parola, visto che sto scrivendo questa tesi dalla stanza 311 del Morgagni, rigorosamente con la finestra aperta, come ho lavorato moltissime volte durante questo dottorato.



Infine permettetemi di ringraziare tutti coloro che non hanno esitato a giudicarmi come un pazzo quando circa 10 anni fa ho detto loro di voler studiare informatica all'università pur essendo cieco, sostenendo che volessi fare una cosa semplicemente impossibile. A loro era dedicato il mio lavoro di tesi per la laurea triennale; e dopo aver confermato la mia pazzia durante la tesi magistrale devo dire che “non c'è due senza tre!” E sì, a quanto pare questo pazzo è riuscito a diventare persino dottore di ricerca... In informatica, pur essendo cieco!

Ma una delle cose più belle che mi ha trasmesso lo studio dell'informatica in questi quasi 10 anni è sicuramente il *background*: tutti quei fatti, citazioni, frasi comuni che semplicemente si devono conoscere, senza se e senza ma, per integrarsi nell'ambiente. Ecco: forse sarà una delle frasi più abusate di sempre, e per uno strano scherzo del destino è stata proprio una delle prime che ho conosciuto durante i primissimi giorni di quel settembre 2013; ne è passato di tempo, ma non saprei come altro concludere questo lavoro di tesi se non con il più sentito:

So long, and thanks for all the fish!

