Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

DIPLOMARBEIT

A free online Web Accessibility Evaluation Tool

ausgeführt am Institut für

Informationssysteme

Arbeitsgruppe Datenbanken und Artificial Intelligence (DBAI)

der Technischen Universität Wien

unter der Anleitung von

Univ.Prof. Mag.rer.nat. Dr.techn. Reinhard Pichler

Mag.rer.nat. Dipl.-Ing. Katrin Seyr

und

Dipl.-Ing. Dr.techn. Gerald Pfeifer

durch

Johannes Egger
Kriemhildplatz 10/26, 1150 Wien
Matnr. Nr. 9726392

Wien, am 11. Februar 2008

Unterschrift

Kurzfassung

In der zweiten Hälfte des 20. Jahrhunderts verabschiedeten einige Staaten Gesetze zur Gleichstellung behinderter Menschen. Barrieren für Menschen mit Behinderungen haben ihren Ursprung nicht nur in baulichen Anlagen, sondern umfassen auch Produkte und Dienstleistungen sowie Informations- und Kommunikationstechnologie. Barrierefreies Internet (Englisch: Web Accessibility) bezeichnet Internet-Angebote, die von allen Menschen unabhängig von Behinderungen uneingeschränkt genutzt werden können. Das W3C hat 1999 einen Standard zur Förderung eines barrierefreien Internet verabschiedet: Die Zugangsrichtlinien für Web-Inhalte 1.0 (Englisch: Web Content Accessibility Guidelines 1.0, WCAG 1.0). Diese Arbeit befasst sich mit der Implementierung eines freien online Tools zur Überprüfung von Webseiten auf Barrierefreiheit.

Nur ein Teil der Barrieren auf Webseiten kann durch Algorithmen automatisch erkannt werden; einige müssen durch Menschen überprüft werden. Existierende Tools implementieren einen Teil der automatisierbaren Tests. In dieser Arbeit stellen wir eine Testmethodik vor die versucht, die Anzahl der automatisierbaren Tests zu maximieren. Die Testmethodik wurde in einer Programmbibliothek implementiert. Eine Webapplikation stellt eine Benutzerschnittstelle zur Verfügung und ermöglicht verschiedene Repräsentationen der Testresultate.

Eine Evaluierung unseres Tools durch ein Softwarepaket zum Review von Barrierefreiheits-Testern zeigt merkliche Unterschiede in der Interpretation der WCAG 1.0 Richtlinien.

Abstract

In the second half of the 20th century, a number of countries have passed legislation to protect people with disabilities from discrimination. So-called accessibility barriers are not limited to architecture, but extend to products and services, including information and communications technology such as the Web. Web Accessibility is concerned with Web sites that are accessible to everyone regardless of their disability. In 1999 the W3C released its Web Content Accessibility Guidelines 1.0 (WCAG 1.0), a set of guidelines to make Web sites accessible. This thesis is concerned with the implementation of a free online Web Accessibility Evaluation Tool.

Only a subset of accessibility barriers can be determined by software algorithms alone; some need to be determined by human intervention. Existing tools implement a subset of checks that can be automated. We created an Evaluation Methodology that attempts to maximize the checks that can be evaluated automatically, and implemented this methodology in a library. A web application provides a user interface and a number of different representations of the evaluated data.

From an evaluation of our tool using an accessibility tool reviewer we conclude that there are significant differences in how the WCAG 1.0 guidelines are interpreted.

For my parents.

Acknowledgements

I am especially grateful to my advisor Dr. Gerald Pfeifer, who came up with the subject of this thesis. He made a number of suggestions for improvements, and his support and guidance was invaluable. He also allowed me to take an extended hiatus to work for local NGOs in southern Thailand in the aftermath of the December 24th, 2004 tsunami, a debt I hope to repay one day. I also wish to thank Katrin Seyr and Prof. Dr. Reinhard Pichler, my advisors at Vienna University of Technology.

Barbara Klotz and Carly Festa took it upon themselves to proof-read this thesis, busy though they were. Christoph Oberhofer gave me his opinions on disability in architecture.

My friends at Final Five Studios, Timm Jansen and Fabian Jung, provided the EOWA logo. Timm also gave me feedback on an early prototype. I am very grateful.

Tom Insam shared his insights into Character Encoding Scheme handling for Scripting Media Types. Ben Holzman answered my questions concerning his XML-Generator Perl module.

A small host of other people lent me their eyes, minds and opinions, including Hannes Kaiser, Andreas Trenkwalder and John Pierson.

Finally, I would like to thank my parents for their ongoing support, and for the opportunity to attend university.

Table of Contents

List of Figures

1. Introduction

It is widely known that in many countries, new buildings have to comply with principles of Accessible Design so as not to disadvantage persons with a disability. Universal Design, or Design for All, is a philosophy for designing products and services that accommodate the widest possible range of functionality, and can be used by all people without the need for adaptation, including the disabled.

In some countries this requirement applies only to buildings of federal and local Governments, and of public organizations, associations and service providers, such as Health Care and Transportation. In other countries, accessibility requirements apply to any site, facility, or building during design, construction, or alteration, including the private sector [1][1].

Over the past few decades, awareness of accessibility in architecture has been raised due to efforts in communication and legislation on these matters. However, accessibility issues are not limited to architecture. They pertain to any service or product, whether it be used by people able or disabled. This includes products and services in Information and Communications Technology (ICT). At the 2004 International Workshop on Accessibility Requirements for Public Procurement in the ICT Domain [2], it was stated that although the Information Society could potentially create a more inclusive society, without accessibility rules we risk creating new barriers for people with disabilities. "A good example of how this can happen is the case of PCs and blind people. Statistics show that the development of new computer technology has dramatically reduced blind people's ability to use personal computers. In this way, while 99% of PCs could potentially be used by blind users in 1990, only 33% are accessible nowadays", said a member of the EU Economic and Social Committee [2].

Although the Web is only one of many ICT's, it is arguably one of the most important ones, if not the most important one. Unfortunately, accessibility barriers apply even to this technology, thereby effectively disadvantaging disabled citizens. From personal experience, we infer that it is largely unknown whether some within the disabled community, especially the blind, rely on the Web more strongly than the average person, or whether they use it at all. For some disability groups, the Web offers unprecedented opportunities, allowing them to independently pursue activities for the first time. For instance, before the Web made it possible for the blind to read newspapers online, they had to rely on others to read it to them [3].

The Web potentially allows people with disabilities to retrieve information, read newspapers, shop online and communicate with others without having to rely on third parties to help them. But people with disabilities can only access Web content if it is presented in a way accessible to them, according to certain specifications.

[1]All Web pages referenced in this thesis have been accessed between December 2007 and January 2008.

For example, if information on a Web site is only represented within an image, and no equivalent textual description is given, a blind user will be unable to access that information. Ignorance regarding these matters can result in partial or complete inaccessibility of Web content by the disabled community.

This thesis deals with the accessibility of Web sites by people with disabilities, a subject commonly referred to as Web Accessibility.

In the next section of this chapter, Web Accessibility is introduced by example, recounting how in June 1999, Bruce Maguire, an Australian citizen, filed a complaint against the organizing committee of the Sydney Olympic Games (SOCOG) because he could not access their Web site.

1.1. Introducing Web Accessibility: Maguire v. SOCOG

"It is a primary consideration that as far as possible all Australians should have the capacity to share equally in an event of this significance; an alternative source which makes available the same amount or body of information is simply not available." (Hon. William Carter, QC)

Many countries around the world have legislated discrimination laws that require Governments, government institutions, educational institutes, corporations and businesses to provide equal opportunities for people with disabilities. In Australia, the relevant law is the Commonwealth Disability Discrimination Act of 1992 (DDA) [4]. Section 24 of the DDA states that it is unlawful for someone who provides goods, services or facilities, to discriminate against another on the grounds of their disability.

The Australian Human Rights and Equal Opportunities Commission (HREOC) [5] (hereafter referred to as "the Commission") is an independent government body, established in 1986. It has the authority to investigate matters of discrimination on the grounds of disability [6].

1.1.1. The Complaint

On June 7th of 1999, Bruce Lindsay Maguire, who is blind since birth, filed a complaint with the Commission under the DDA, alleging that he was being discriminated against by the Sydney Organising Committee for the Olympic Games (SOCOG) in three respects: the failure to provide Braille copies of ticket books, the failure to provide the souvenir programme in Braille, and the failure to provide a Web site accessible to the blind [7].

Maguire stated that, before lodging his complaint, he spoke to SOCOG personnel, seeking information on the availability of the ticket book in Braille and had been told that "blind people can access information if it is available on the internet". He had replied, “That is not correct. We can only access information if it is presented in accordance with international accessibility guidelines. The SOCOG website does not comply with those guidelines, so a lot of information is not accessible to me”. Maguire was then told that he would have to seek assistance from a sighted person when accessing the Web site [7].

1.1.2. Expert Witness Testimony

In response to the complaint, SOCOG stated that to make its Web site compliant with W3C guidelines[2], and thus accessible to the blind, would cause considerable expense, and that “such expense would be an unjustifiable financial imposition” [7].

To verify this claim, two expert witnesses for Maguire prepared reports on the accessibility of the SOCOG Web site: Tom Worthington, an architect of the Commonwealth Government’s Internet and Web strategy, and Jutta Treviranus, an academic and chair of the W3C Authoring Tool Guidelines Working Group [8].

Maguire’s lawyer requested that SOCOG provide details of the architecture of their Web site, a request that was denied on grounds that it was “highly commercially sensitive information” [7]. The experts had to use the information available on the current Web site to evaluate accessibility compliance and to estimate the costs needed to make the site accessible [8].

By April 2000, SOCOG had made some changes to their site but Maguire contended that some parts were still inaccessible to him. He narrowed his complaint down to three specific issues and asked the Commission to order SOCOG to make the following modifications:

- *to provide ALT text*[3] *for all images and image maps* [7].
- *to provide access to the Index of Sports from the Schedule page* [7].
- *to provide access to the Results Tables during the Olympic Games* [7].

[2]The Reasons for Decision of the Hon. William Carter, QC, refers to the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines 1.0 (WCAG 1.0), an internationally recognized set of standards on how to make Web sites accessible to people with disabilities. See also Section 2.5.

[3]“Alternative text” is placed in the HTML code for an image on a Web site and is displayed if the image is not loaded or not available. It is also used by assistive technologies, such as so-called “screen readers”, software that reads aloud the elements of a web page. Assistive technologies cannot process images that lack a text equivalent; disabled people are at a disadvantage if information is only conveyed through images.

1.1.3. SOCOG's Response

In its response, SOCOG claimed that they had implemented ALT text for images and image maps, and that "access to the Index of Sports from the Schedule was available and had always been available by a different route; namely, by entering the URL for each sport directly into the Web browser" [7]. SOCOG sent Maguire a letter indicating the URLs for 36 nominated sports and suggested he type them in manually. Regarding the matter of the Results Tables, SOCOG stated that compliance would inflict "unjustifiable hardship" [7] under the DDA.

1.1.4. "Unjustifiable Hardship"

A number of anti-discrimination laws throughout the world know the concept of *unjustified*, *unreasonable* or *undue hardship* or *burden* [9]. In Australia, an employer or provider of goods and facilities may not be required to provide equal access in circumstances where this would e.g. impose a significant financial burden on the alleged discriminator's business. Unjustifiable hardship is not explicitly defined; the Commonwealth DDA states that "in determining what constitutes unjustifiable hardship, all relevant circumstances of the particular case are to be taken into account" [4].

In the present case, SOCOG stated that to comply with Maguire's request, additional infrastructure and additional designs to display the Table of Results in a manner accessible to the disabled would have to be added to the site, at a cost of 2.2 million AUD. Maguire's experts contended that the cost involved would be modest and take a small group of Web developers four weeks to implement [7].

The Commission based its decision on competing evidence of the expert witnesses of both sides. The HREOC Commissioner, William Carter QC, preferred the evidence provided by Maguire's experts because their reputation was impressive and convincing, and because the experts of the defense were engaged only a few days before the hearing, and had little knowledge and experience with the Web site. It was concluded that the damage to SOCOG would be modest and, had the matter been addressed from the beginning, negligible. The claim of unjustifiable hardship was therefore rejected [7].

1.1.5. Discrimination under the Commonwealth DDA

Under Section 24(1) of the Commonwealth Disability Discrimination Act, it is "...unlawful for a person who ...provides goods or services ...to discriminate against another person on the ground of the other person's disability ..." by refusing to provide these goods or services, or in the terms or conditions or the manner in which those services are provided. [4]

The Commission found the argument that one could manually type in URLs to access Web pages to be "both unorthodox and cumbersome and need not be resorted to by a sighted person". Maguire said it was an inadequate way of accessing the Index because "that is not the way that people use web pages" [7].

The Commission held that in creating a Web site and providing information about the Olympic Games, SOCOG offered a public service. However, due to the way the information had been made available, it could not be properly accessed by a blind person because of their disability. Since Maguire was unable to use the provided services in a manner equivalent to a sighted person, the Commission ruled that an act of discrimination under the DDA had indeed taken place, and that SOCOG was in violation of Section 24 of the DDA [7].

The Reasons for Decision of the case emphasize the uniqueness and the historical and cultural significance of the Olympic Games; blind and sighted people should equally be able to follow the Games, especially since an equivalent alternative simply does not exist.

1.1.6. Delayed Proceedings

The Commissioner noted the repeated attempts of the defense to delay the proceedings by refusing to provide information sought by the complainant's experts, by attempting to vacate hearing dates, and by failing to provide statements of its expert witnesses a week before the hearing.

1.1.7. The Commission's Decision

The Commission decided that SOCOG had violated the DDA in that it had provided a public Web site which was inaccessible to Maguire because of his blindness. It was ordered that SOCOG provide ALT text for all images and image map links on its Web site; that they provide access to the Index of Sports; and that they provide access to the Results Table on the Web site during the Sydney Olympic Games [7].

SOCOG ignored the Commission's ruling. In November 2000, the Commission held that SOCOG pay Maguire a compensation of 20.000 AUD [10]. SOCOG paid the fine.

1.1.8. Impact

In the case of Maguire v. SOCOG, the Commission repeatedly referenced the W3C WCAG guidelines. It was the first time formal reference was made to the Web Content Accessibility Guidelines in court.

In June 2000, Ministers from the Commonwealth, State and Territory Governments met in Adelaide for a meeting of the Online Council[4]. The Council adopted the W3C WCAG guidelines as the "common best practice standard for all Australian government websites" [13].

In August 2002, the Human Rights and Equal Opportunities Commission released the World Wide Web Access DDA Advisory Notes. This document provides background information on legal issues and contains advice on how to avoid disability discrimination. Section 2.2 specifically states that the "provision of information and online services through the Worldwide Web is a service covered by the DDA. Equal access for people with a disability in this area is required by the DDA where it can reasonably be provided" [14].

1.2. Task and Motivation

The main task of this thesis is the implementation of a Web application that evaluates Web sites for conformance to accessibility guidelines. Existing Web applications in this field typically serve as a demonstration for the commercial products of the respective companies.

Although it may be argued that the accessibility of public Web sites is a moral obligation, in Section 2.7 we will see that it is also a legal requirement in many countries. Consider that with resolution COM(2005) 425, the European Commission has recognized the need for further action in this area, and that the Commission has taken initiatives in the previous eEurope 2002 and eEurope 2005 Action Plans [15]. We hope that this thesis will raise awareness in this area, and that our tool will provide the designers and providers of Web sites as well as third parties and special interest groups with a means to evaluate their sites for Web Accessibility conformance.

Why implement another Accessibility Evaluation Tool? Our tool will provide features that other tools in this area do not provide at all, or not to a sufficient extent: it will implement additional algorithms, offer a number of different views of the evaluated data, save copies of the evaluated Web sites for future reference etc.

1.3. Overview of rest of thesis

The next chapter introduces concepts and terminology essential to Web Accessibility. It gives a possible definition of the term, explores how the disabled use the Web and lists categories of disabilities that can affect the accessibility of Web pages.

[4]The Online Council (OC) was established in 1997 by the Australian Government. States, Territories and local governments agreed that cooperation on online issues was needed to promote consistency at national level [11]. At the twelfth Ministerial Meeting of the Council in August 2005, its name was changed to Online and Communications Council [12].

The reader is then familiarized with accessibility barriers experienced by disabled individuals when using the Web, and presented with the W3C's Web Content Accessibility Guidelines 1.0 (WCAG 1.0), a set of standards that try to address these barriers. We analyze the legislation regarding Web Accessibility in selected countries and address the relevance of Web Accessibility. The last section of Chapter 2 introduces Web Accessibility Evaluation Tools.

We evaluate the state of the art in the area of Web Accessibility Evaluation Tools in Chapter 3, "Related work". The task itself is illustrated in more detail in Chapter 4, "Task and Requirement Analysis", which gives a rationale for implementing a tool that offers more functionality than existing tools, and describes its architecture and environment.

The Web Content Accessibility Guidelines 1.0 are meant to be stable across evolving Web technologies and do not provide implementation details for its checkpoints. Chapter 5 describes an Evaluation Methodology for WCAG 1.0. The actual conformance tests can be found in Appendix A, "Techniques for Evaluation of Conformance to WCAG 1.0".

Chapter 6 describes the implementation of our tool, and Chapter 7 evaluates the implementation. The final chapter gives a summary and suggestions for further work and improvements.

2. Concepts and terminology

"For people without disabilities, technology makes things convenient. For people with disabilities, it makes things possible." (Judith Heumann)

The next sections introduce the concept of Web Accessibility and related issues. In the first section, Web Accessibility will be explored and a possible definition of the term will be given. To understand the relevance of the subject, it is important to know how individuals with disabilities use the Web, what kinds of disabilities exist, and how a disability can affect the access to Information and Communication Technology (ICT). These topics are explored in Section 2.2 and Section 2.3.

Section 2.4 describes accessibility barriers caused by Web content, by user agents and by environmental constraints. These barriers emphasize the need for a set of guidelines to promote accessibility on the Web. The W3C's Web Content Accessibility Guidelines were developed for that purpose and released in 1999[5]. They are described in Section 2.5. The subsequent section introduces HTML concepts relevant to Web Accessibility.

Section 2.7 deals with Web Accessibility in legislation, which plays an important role in promoting the accessibility of Web sites and Web services. The respective legislation in the United States, the European Union and selected European countries is analyzed in detail. We then address the relevance of Web Accessibility in a moral, commercial and technical context in Section 2.8.

The last section of this chapter introduces Web Accessibility Evaluation Tools.

2.1. Defining Web Accessibility

The term accessibility is usually associated with architecture, rather than Web site design. It means creating spaces that meet the needs of all: people young and old, able and disabled. Legislation determines minimum standards for new (government) buildings. As a result, new buildings often provide wheelchair ramps, accessible lifts, disability parking spaces, special sanitary equipment, tactiles for orientation, blended curbs, and detectable directional warning systems [18], thus allowing anyone and everyone to gain access to a building and use the services provided therein.

Similarly, the subject of Web Accessibility is concerned with Web sites that are accessible to all users who want to access them. Such Web sites are sometimes referred to as "Universal Web Sites". Tim Berners-Lee, director of the W3C and

[5]Version 1.0 of the Web Content Accessibility Guidelines was released in 1999 [16]. The first working draft for WCAG 2.0 was released in January 2001. At the time of writing WCAG 2.0 is a "Last Call Working Draft" and is expected to be completed in 2008 [17].

considered the "inventor" of the World Wide Web, speaks of "access by everyone, regardless of disability".

Web content is said to be accessible when it can be used by someone with a disability. However, what seems like a simple statement is difficult to narrow down to a formal set of rules to which Web content must adhere [19]. There are a multitude of possible definitions for the term Web Accessibility. The following is a very strict definition demanding that a site meet high requirements:

*"**Anyone** using **any kind of Web browsing technology** must be able to visit **any site** and get a **full and complete understanding of the information contained there,** as well as have the **full and complete ability to interact with the site**"* [20].

To evaluate Web sites for accessibility, this definition is not practical. We will hear more about a standard that addresses barriers on Web pages in Section 2.5.

2.2. How people with disabilities use the Web

A number of people who have used the Internet for work or in a personal environment can hardly conceive of life without it. It provides access to information, new means of communication, of conducting business and purchasing goods online. It allows for ways of entertainment that were impossible 15 years ago. With its ability to provide information at any time and about almost any topic conceivable, it has shaped a way of life for a new generation. Arguably the most revolutionary single invention since Gutenberg's printing press around 1450, the world can be at one's fingertips—if one does not have any kind of disability [3].

For people with disabilities, Information and Communications Technology (ICT), and in particular the Internet, provide unprecedented opportunities to gain access to education, employment, information, the purchase of goods, and communication. By way of example, consider how blind people would read the newspaper before they had access to the Internet. For the most part, they could not. They might have asked someone to read it for them, but that made them dependent upon others [3]. Today, many newspapers publish their content online in a form that can be processed by so-called screen readers used by the blind. These special software programs read text out loud, allowing people with visual impairments to read their favorite newspaper as well as other accessible online content [21].

Similarly, people with motor disabilities can use assistive technologies that emulate computer input, such as eye-tracking software, allowing people to use a computer with nothing but eye movements, as well as special keyboards, mouth sticks, head wands, and oversized trackball mouses, and voice recognition software [22].

Though estimates vary, the percentage of people with disabilities around the world is said to be between 10% and 20%[6] [32]. Not all kinds of disabilities affect access to the Internet, and in particular the Web, but many of them do. As the number of people using the Internet increases, so does the number of disabled individuals using the Web. For people with disabilities, access to ICT is more important than for others. People without disabilities have more traditional sources of information to fall back on, such as using libraries, reading newspapers, magazines, etc [32].

2.3. Disabilities that can affect Web Accessibility

At this time, there are no universally accepted categories of disabilities. The terminology in use varies from country to country, and sometimes even within different communities in the same country [33]. Disability should be thought of as a spectrum, with different types and levels of obstacles [34]. The following list is an excerpt from the W3C technical report "How people with disabilities use the Web"[7] [33]:

- ***Visual Impairments:*** *blindness, low vision, color blindness*
- ***Hearing Impairments:*** *deafness, hard of hearing*
- ***Physical Disabilities:*** *motor disabilities*
- ***Speech Disabilities:*** *articulation disorder, expressive and receptive language disorder, aphasia [35]*
- ***Cognitive and Neurological Disabilities:*** *dyslexia, dyscalculia, attention deficit disorder, intellectual impairments, memory impairments, mental health disabilities, seizure disorders*
- ***Multiple Disabilities***
- ***Ageing-related Conditions***

[6]International figures are mostly estimates based on national statistics. However, consider that according to the U.S. Census Bureau [23], in 2000 19.3% of the U.S. population claimed to suffer from "some type of long-lasting condition or disability", and approximately 10% claimed to have a severe disability [24]. The 2007 statistic of the Canadian Premier's Council on the Status of Disabled Persons reports a disability rate of 14.3% [25]. According to Eurostat [26], the percentage of disabled citizen among the population of the European Union was at an average of 12% in the ten Member States covered in 1991 [27]. In a 2002 Eurostat survey, 16.4% of the population of the 15 Member States and 14.3% of the population of the ten acceding countries aged between 16 and 64 claimed to suffer from some "long-standing health problem or disability" [28]. According to the Equal Opportunities for people with disabilities Action Plan, the percentage of people with disabilities in the acceding States amounts to 25% [29]. The Australian Bureau of Statistics [30] found that in 2003, 20% of the population had reported a disability [31].

[7]The section on speech disabilities is translated from a section in Dilling, Mombour et al: Internationale Klassifikation psychischer Störungen, Huber, 2000.

These disabilities affect the use of ICT's, and the use of the Web, in different ways. A number of assistive technologies help to maintain or increase the capabilities of individuals with disabilities [36, Sec.3.(a)(3)]. The next section describes accessibility barriers that the disabled come across when they use the Web.

2.4. Accessibility barriers

As is the case with different kinds of accessibility, there are several types of accessibility barriers. They can be caused by Web content, by a user agent, or by the environmental conditions in which a person operates. Although this thesis focuses on accessibility barriers on Web pages, this section also includes other accessibility barriers.

Consider that many people use the web under conditions unfamiliar to those without disabilities. The following list of contexts under which users may operate is a direct excerpt from the Web Content Accessibility Guidelines 1.0 [16]:

- *They may not be able to see, hear, move, or may not be able to process some information easily or at all.*
- *They may have difficulty reading or comprehending text.*
- *They may not have or be able to use a keyboard or mouse.*
- *They may have a text-only screen, a small screen, or a slow internet connection.*
- *They may not speak or understand fluently the language in which the document is written.*
- *They may be in a situation where their eyes, ears, or hands are busy or interfered with (e.g., driving to work, working in a loud environment, working in over- or under-illuminated rooms, etc.).*
- *They may have an early or different version of a browser, a voice browser, a different user agent entirely, or a different operating system.*

To promote accessibility on the Web, a set of guidelines that address barriers on Web pages is needed. Members of the World Wide Web Consortium's (W3C) Web Accessibility Initiative (WAI) have developed the internationally recognized Web Content Accessibility Guidelines 1.0 (WCAG 1.0), which are introduced in more detail in the following section.

2.5. The Web Content Accessibility Guidelines 1.0

In April 1997, the World Wide Web Consortium (W3C) announced its new Web Accessibility Initiative (WAI) domain to "promote and achieve Web functionality for people with disabilities" [37]. By February 1998, WAI released the first version of the Web Content Accessibility Guidelines as a W3C working draft. In May 1999, the W3C announced the release of the Web Content Accessibility Guidelines 1.0 as a W3C Recommendation[8]. It recommends the WCAG guidelines as a way to make the Web accessible [32].

2.5.1. Guidelines and Checkpoints

The Web Content Accessibility Guidelines outline principles for accessible Web design, such as the need to provide equivalent alternatives to auditory and visual content. The document contains fourteen specific guidelines, each of which consists of several "checkpoints" that explain how accessibility issues apply to specific site features. Each checkpoint is identified by a unique number, its statement and a priority. The checkpoints are intended to be specific enough so that a non-disabled person reviewing a page or site should be able to verify whether or not they have been satisfied [16].

2.5.2. Priority levels and Conformance

The WCAG Working Group has assigned a priority level to every checkpoint. The following list of priority levels is an excerpt from the Web Content Accessibility Guidelines 1.0 [16], with priority 1 being the highest, and priority 3 being the lowest:

Priority 1 *If a Web site does not satisfy this checkpoint, some disability groups will be unable to access information provided by the site.*

Priority 2 *If a Web site does not satisfy this checkpoint, some disability groups will have difficulty accessing information provided by the site.*

Priority 3 *If a Web site does not satisfy this checkpoint, some disability groups will find it somewhat difficult to access information provided by the site.*

The Web Content Accessibility Guidelines define three levels of conformance to its guidelines and checkpoints [16]. The following direct excerpt is a list of conformance levels:

- ***Conformance Level "A":*** *The Web site conforms to all Priority 1 checkpoints.*

[8]A W3C Recommendation is a technical report, usually a specification or a set of guidelines, that is considered stable and has undergone extensive review in- and outside of the W3C. W3C Recommendations are endorsed by the W3C for wide deployment. They are similar to standards published by other organizations.

- ***Conformance Level "Double-A":*** *The Web site conforms to all Priority 1 and 2 checkpoints.*
- ***Conformance Level "Triple-A":*** *The Web site conforms to all Priority 1, 2 and 3 checkpoints.*

2.5.3. Benefits of accessible Web design

The W3C is internationally recognized as a non-profit organization that creates standards for the technologies that make up the Web. The documents of the Web Accessibility Initiative constitute the most widely accepted guidelines to promote Web Accessibility. Following the accessibility guidelines provided by W3C will make Web content, search engines, agents, etc. more accessible to all users, regardless of which user agent they may be using [16]. Checkpoints that broaden Web access for people with visual impairments also help people accessing the Web through mobile phones, hand-held devices, and automobile-based PCs. Providing captions of the audio track of a video presentation helps individuals with hearing impairments, but it also assists those who use the Web in noisy or silent environments, and makes it easy to search through audio content. The use of Cascading Style Sheets for presentation does not only facilitate the accessibility of a Web page, it may also speed up download time and reduce the costs of maintaining and updating the "style" of a site [37].

The mission of the W3C consortium is to bring the Web to its full potential. This includes promoting the interoperability of the Web [32]. The WCAG guidelines promote interoperability by requiring compliance with other W3C standards.

2.6. HTML concepts relevant to Web Accessibility

This section introduces some HTML concepts that are necessary when applying the WCAG guidelines to HTML[9].

2.6.1. SGML and HTML

The Standard Generalization Markup Language (SGML) is "[a] language for document representation that formalizes markup and frees it of system and processing dependencies" [38]. SGML itself is not a document markup language, but rather a specification on how to create one, in other terms, a meta-language for defining document markup languages. Each markup language defined in SGML is called an SGML application [38].

[9]Some of the concepts have little relevance for Web Accessibility, but are important to understand the implementation of the tool.

The Hypertext Markup Language, or HTML, is one example of an SGML application. It is the language used to create the hypertext documents that we call Web sites. The definition of HTML includes a document type definition (DTD) that defines the syntax of HTML markup (see next section) [39].

Everything we consider about HTML also applies to the Extended Hypertext Markup Language, or XHTML [40], which is an evolutionary step beyond HTML 4.01. Note, though, that XHTML imposes somewhat stricter syntactic rules than HTML[10].

2.6.2. Markup and tags

When examining the source code of a Web page, HTML looks like chunks of text surrounded by code that indicates its logical structure, and helps determine how a document should look when rendered on an output device. The pieces of code surrounding the data are called markup. Markup is text that is added to a document in order to convey structural information about it [38]. In HTML, markup constructs are usually (and sometimes incorrectly) referred to as tags[11]. How markup is processed is defined by the semantics of a particular markup language.

Tags describe the structure and other attributes of a document, independent of any processing that may be performed on it. They identify the start and the end of elements [38]. Tags are usually enclosed in angle brackets. A start tag is thus written `<element-name>`, an end tag is written `</element-name>`. The next section will illustrate Markup and tags with an example.

2.6.3. Elements and attributes

In HTML, an element is the primary syntactic language construct. It consists of a pair of start and end tags and their content, or of an "empty" tag that requires no closing tag or content. By way of example, consider the following HTML code:

```
<blockquote lang="en-gb">
  <p>Was he free? Was he happy? The question is absurd:
     Had anything been wrong, we should certainly have heard.</p>
  <p>-- W. H. Auden, <em>The Unknown Citizen</em></p>
</blockquote>
```

The snippet is a `blockquote` element, marking up a quotation. It consists of the start tag, element content, and the end tag. The element content of the `blockquote`

[10]XHTML 1.0 is the formulation of HTML 4 as an XML 1.0 application. XML itself is a subset, or restricted form, of SGML [41].

[11]SGML knows four different markup categories, the predominant being descriptive markup, also called "tags". The other categories are references, markup declarations, and processing instructions [38].

element is made up of two `p` elements, which represent paragraphs. The first `p` element contains only textual content, while the second contains both textual content and an `em` element indicating emphasis.

HTML elements may represent paragraphs, hypertext links, lists, tables, images, etc. Element names are case-insensitive [39].

Characters that occur between the start-tag and the end-tag of an element are referred to as element content. A text element is an element that causes text characters to be part of a rendered document. It may consist of textual data and/or other elements.

HTML elements may have properties associated with them, which are called attributes. Attributes have names and may have values assigned to them. The name/value pairs of attributes are included in the start tag of an HTML element, as can be seen with the `lang` attribute of the `blockquote` element in the example above. Any number of attribute name/value pairs may appear there; the HTML specification indicates which attributes are valid for a particular element. Attribute values are delimited using either double or single quotation marks. As is the case with element names, attribute names are case-insensitive [39].

HTML elements or attributes that have been outdated are called "deprecated". Deprecated elements and attributes may become obsolete in future versions of HTML [39].

2.6.4. Omitting start and end tags

HTML 4 is a form of SGML that supports "implication", meaning that some HTML elements, such as `p` and `li`, allow the author to omit the closing tags. The idea is that opening a subsequent tag "implicitly" closes the current tag because the HTML code would otherwise not be correct according to the HTML document type definition (DTD) [39, Ch.3].

```
<p>Bunnies, it must be bunnies!
<ul>
  <li>item 1
  <li>item 2
</ul>
```

In the above example, the `ul` start-tag implicitly closes the `p` element, because an `ul` element may not be enclosed in a `p` element. The second `li` element implicitly closes the first `li`, because `li` elements may not be nested. The `ul` end-tag implicitly closes the second `li` element because the HTML would otherwise not be well-formed.

In addition, HTML allows some elements to be omitted altogether, so if an element is not valid in the context it seems to be used in, the parser may be able to determine what the correct context is. For example:

```
<title>A room of state in a castle</title>
He was a man, take him for all in all
I shall not look upon his like again.
```

The above is valid HTML 4.01 Transitional. A HTML parser will determine that the `title` element can only occur in a `head` element, and open a `head` element implicitly, while the text can only occur within a `body` element, and is implicitly wrapped in a `p` element.

XHTML does not allow the omission of start- or end tags.

2.6.5. Client-side and Server-side

In computer science, the term "client-side" usually refers to operations that are performed on the client in a client-server environment, whereas "server-side" refers to operations that are being handled by the server.

2.6.6. Image maps

An image map assigns geometric regions to an image and associates each region with a specific action. When a region is activated, the corresponding action is executed. There are two types of image maps: client-side and server-side image maps.

When a user activates a client-side image map region, the coordinates are interpreted by the user agent on the local machine. The user agent follows the link target of the activated image map region.

In contrast, when a user activates a region of a server-side image map, the coordinates of the pointing device are sent to the server, which interprets the coordinates and takes some action [39].

2.6.7. Device independence

Some WCAG checkpoints require device independence: "Users must be able to interact with a user agent (and the document it renders) using the supported input and output devices of their choice and according to their needs" [16]. This does not mean that every user agent must support every input or output device, but rather that a user agent should provide redundant means for input and output for the supported devices [16].

2.6.8. Intrinsic Events

HTML 4.01 allows to associate an action with pre-defined events that occur when a user interacts with a user agent. These “intrinsic events” are HTML attributes that take a script as their value, which is executed whenever the corresponding event occurs. This scripting code is often referred to as the “event handler” [39].

2.6.9. Programmatic objects

Programmatic objects are pieces of program code that accompany a HTML document or are embedded in it. The `object` element allows authors to embed program code written in other languages, such as Java [42] or Macromedia Flash [43]. The `applet` element can be used to include Java-based applets[12], but it has been deprecated in favor of the generic `object` element [39].

The `script` element allows authors to include client-side scripts that are executed when the document is loaded or when a specific event occurs, via intrinsic event attributes. Although the most common scripting language is probably Javascript, any scripting language can be used in HTML, however the code will only be executed if the user agent supports that language [39].

2.6.10. White space

In computer science, the term “white space” commonly refers to characters that will not be rendered on an output device. The HTML specification defines the following characters as white space characters [39]:

designation	short form	HTML entity decimal	HTML entity hexadecimal
ASCII space	space	&#0032;	&#x0020;
ASCII tab	tab	&#0009;	&#x0020;
ASCII form feed	form feed (ff)	&#0012;	&#x000C;
zero-width space	zwsp	&#8203;	&#x200B;

2.6.11. Data and Layout Tables

In WCAG vocabulary, tables that are used to mark up truly tabular information are called data tables. Tables used merely for layout purposes are called layout tables. See also Section A.4.

[12] A Java applet is a program, written in the Java programming language, that can be embedded into a HTML page and downloaded and executed by a Java-compatible browser [42].

2.6.12. ASCII and ASCII art

The American Standard Code for Information Interchange (ASCII) is a character encoding coding that assigns unique numeric values to letters, digits, and special characters. It was first published as a standard by the American Standards Association (now American National Standards Institute) in 1963. Later versions were developed in cooperation with the International Organization for Standardization (ISO), which led to the publication of ISO 646 in 1967 [44]. At the time of this writing, the most recent edition of this standard is ISO/IEC 646:1991 [45].

The term "ASCII art" refers to a series of text characters and symbols that are combined to create an image. By way of example, the character sequence `:-)` stands for a smiling face [16]. The image representations range from short character sequences to entire screen drawings. Although they make sense visually, they are merely a character sequence to computers. ASCII art may not make sense if read by a screen reader.

2.6.13. Character Sets and Character Encodings

In modern computing, a character repertoire is a set of distinct characters that does not assume any internal representation on a computing system. A code point or code position[13] is a non-negative integer value associated with a character. A mapping of all characters in the repertoire to their respective code points is called a character code, a coded character set, or sometimes a character encoding table[14]. This mapping is often represented in tabular form; examples are the ASCII table and what IBM calls "code pages" [46].

A character encoding[15] is a method of representing characters digitally by transforming code points into sequences of bytes (octets), usually for transmission or storage. In the most simple case, such as ASCII, the code points and the octal values used to represent them digitally are the same. ASCII is a fixed-width encoding scheme, meaning that the sequence of octets representing a code point is always of the same length [46]. This is not necessarily the case for more complex character encodings such as UTF-8, which is a variable-width or variable-length encoding scheme. Most variable-width encodings are multi-byte encodings, which use a variable number of bytes to encode characters.

The term character set is sometimes used to refer to the character repertoire, the character code, or the character encoding.

[13]A large number of synonyms are used for "code point", e.g. code number, code value, etc.
[14]The HTML specification calls this a document character set [39].
[15]In practice, this is complicated by byte order ambiguity. The character encoding model distinguishes between a character encoding form, which assigns each code point a sequence of bytes, and a character encoding scheme, which resolves byte-order ambiguity by specifying the byte ordering scheme to be used [47].

2.6.14. Character Encodings and HTML

A HTML document is a series of characters from the character repertoire, which in turn is part of the character code (in HTML: document character set). The HTML specification does not mandate which character encodings must be supported by user agents. Servers are allowed to change (transcode) the character encoding during transmission to meet the needs of user agents [39]. Note that the character encoding used to represent the document may differ from the encoding used to store it on the filesystem or to transfer it over a network.

2.6.15. HTML and Unicode

Although user agents are free to choose a character encoding of their choice to encode HTML documents, the character repertoire for HTML 4.01 documents is the Universal Character Set (UCS), which ISO/IEC 10646 shares with Unicode[16] [39].

To work around encoding limitations, HTML allows Unicode characters to be represented by so-called character references, which are “a character encoding-independent mechanism for entering any character from the document character set” [39]. Character references are available in two forms:

Numeric character references specify a character’s code point in UCS using the syntax “&#D;” where D is a decimal number representing the code point, or the syntax ”&#xH;” or ”&#XH;” where H is a hexadecimal number representing the code point. For example, &#229; and &#xE5; represent the letter “a” with a small circle above it [39].

Character entity references are named character references explicitly defined for some commonly used characters in the document character set. For instance, &lt; represents the “less than” sign [39].

To process a HTML page a user agent must correctly determine the character encoding of the document. The character encoding may be specified in the HTTP response via the `Content-Type` header, but this is not mandatory. The encoding may also be declared in the document using the `meta` declaration with `http-equiv` set to `"content-type"` and a value for `charset` [39]. Unfortunately, many HTML documents are served with no or with an inaccurate character encoding declaration. Many Web browsers allow the user to override the character encoding by selecting an encoding from a list.

[16]Though they share a character repertoire, Unicode and ISO/IEC 10646 differ in a number of ways that are outside the scope of this thesis.

2.6.16. ALT text

ALT text is an abbreviation of “alternative text”. It refers to descriptive text that is the value of the `alt` attribute of an image on a Web site. If the image is not displayed, the ALT text can be presented by the user agent. ALT text is especially useful to users of so-called “screen readers”, software that converts text into synthesized speech.

2.6.17. Description Link (D-link)

A Description Link is a link that provides additional information about a part of the content of a Web page [48].

An invisible D-Link is represented by a small or transparent image. Its `alt` attribute value is usually “D” or “D-Link”. It refers to a text equivalent of the associated image. Invisible D-Links were a temporary solution for Web designers who wanted to avoid visible D-Links, but they are now deprecated in favor of the `longdesc` attribute [49].

2.6.18. Braille and Assistive Technology

Braille is a writing system using six raised dots grouped in different patterns to represent letters and numbers. People read Braille by running their fingertips across the dots [16].

Assistive Technology is “software or hardware specifically designed to assist people with disabilities in carrying out daily activities” [16].

An example of hardware-based assistive technology are dynamic Braille displays, which are capable of raising or lowering Braille dot patterns, thereby creating a line of Braille that can change from one moment to another [16].

Examples of software-based assistive technology are screen magnifiers and screen readers. A screen magnifier is software program that magnifies a part of the screen to increase readability and visibility. It is usually used by people with low vision. A screen reader is a software program designed to read the text printed on a screen aloud to a user [16].

2.6.19. Equivalent information

A number of checkpoints in the Web Content Accessibility Guidelines require that for some types of content, equivalent information be provided by some other means. The information is considered “equivalent” if it provides the same function for a disabled person that the original content does for individuals without a disability [16].

Alternative contents that are provided in textual form are called text equivalents. They are required for graphic and audio information and must "convey all essential content". Non-text equivalents, such as an auditory description of a video or movie, are meant for individuals that cannot access visual information and written text [16].

2.7. Web Accessibility in Legislation

Legislation plays an important role in promoting Web Accessibility and in furthering the integration of people with disabilities into society. This section explores the respective legislations in the United States and the European Union in some detail.

2.7.1. General Principles

Although promoting accessibility for people with disabilities is seen by many as a moral obligation, there are also legal obligations to consider. In Section 1 we mentioned that in many countries discrimination laws require Governments, government institutions, educational institutes and businesses to provide equal opportunities for people with disabilities. In some of those countries, discrimination equality laws also apply to ICT's [50], while in others it is unclear whether existing policies can be applied to these technologies.

There are a number of different approaches among national laws which address the accessibility of ICT, the Internet, and the Web. Some address accessibility issues in civil rights laws; others require that all ICT in federal procurement must meet accessibility standards; still others require that manufacturers of ICT products and service providers make their products and services accessible to people with disabilities [51]. All these approaches can be found in the national laws described in the next few sections, which are sorted by the specific years in which legislation was passed, with the European Union taking precedence over the United Kingdom.

2.7.2. The United States of America

In the United States there are several laws that relate to Web Accessibility:

- Section 508 of the Amendments of 1998 to the Rehabilitation Act [52]

  Section 508's primary purpose is to provide access to federal agencies' electronic and information technology (EIT)[17] for employees and customers with

[17]Information and communications technology (ICTs) are sometimes called EIT in the United States.

disabilities. It requires that any such technology that is developed, procured, purchased or maintained by federal agencies meet defined accessibility standards.

The 1986 version of Section 508 established non-binding guidelines for technology accessibility, while the 1998 version created binding, enforceable standards and guidelines that have been incorporated into federal procurement procedures. These standards are similar to, and based on, the WAI WCAG guidelines 1.0, Level A.

The scope of Section 508 is limited to the Federal sector. It does not apply to private sector technology companies [53].

Impact. The need for legislation to bring down accessibility barriers is well illustrated by Section 508. In order to understand the importance of accessibility provisions in public procurement, it must be said that public procurement has a very significant effect on the economy. By way of example, consider that in 2006 public procurement transactions accounted for 16% of the European Union's Gross Domestic Product [54].

Under Section 508, mainstream technologies in the United States, such as PCs, telephones and photocopying machines are required to comply with accessibility standards if they are to be purchased by federal agencies. Because the US government is the largest buyer of ICT in the United States, manufacturers have addressed the needs of government employees and disabled users of government services [55].

- Section 255 of the Telecommunications Act of 1996 [56]

  The Telecommunications Act of 1996 recognizes the importance of access to telecommunications for people with disabilities. Section 255 of the Act requires that telecommunication products and services be accessible to people with disabilities [56].

  The Telecommunications Act Accessibility Guidelines (TAAG) [57] were issued in February 1998, setting forth criteria for accessibility, compatibility and usability of telecommunications equipment. They became effective on March 5th, 1998 and specified that manufacturers must consider accessibility and usability in the early product design phase. It further states that special information on the use of products and services by the disabled must be provided [58].

  The Federal Communications Commission (FCC) enforces the Telecommunications Act. Accessibility complaints can be filed with the FCC, however, lawsuits are not possible and there is no provision for damages [56].

  Impact. Since lawsuits are not possible and there is no provision for damages, the impact of the Telecommunications Act seems negligible.

- The Americans with Disabilities Act of 1990 [59]

  The Americans with Disabilities Act (ADA) is a civil rights legislation that protects individuals with disabilities from discrimination. Of particular interest concerning Web Accessibility are specific sections of Title II and Title III of the Act. Title II states that communication with people with disabilities must be "as effective as communication with others", while Title III covers public accommodation of disabled citizen [59].

  Whether or not the ADA applies to Internet Web sites is a question that does not have a straightforward answer. The following case studies try to determine whether Title II and Title III apply to Internet Web sites. Under Title II, it may be argued that communication over the Internet must be as effective as communication by other means, while under Title III, it could be argued that the Internet is a place of public accommodation.

  In an answer to an inquiry by Senator Tom Harkin in 1996 [60], the U.S. Department of Justice set forth that it considers the Internet a place of public accommodation under Title II. Because the Department of Justice is the agency given authority to enforce the ADA, many hold this view as persuasive. It will be seen that court decisions do not necessarily reflect this notion.

  Carparts v. Automobile Wholesaler's: In the case of Carparts Distribution Center v. Automotive Wholesaler Association of New England, a case involving insurance coverage, the Court of Appeals stated that "[i]t would be irrational to conclude that persons who enter an office to purchase services are protected by the ADA, but persons who purchase the same services over the telephone or by mail are not. Congress could not have intended such an absurd result" [61]. Although this statement does not specifically mention the Internet, it involves products and services using the telephone. The Court of Appeals also noted that "[t]he plain meaning of the terms do not require 'public accommodations' to have physical structures for persons to enter." [61] In a similar case, the Seventh Circuit also reached the conclusion that public accommodations are not limited to physical places [62].

  Hooks v. OKbridge: In Hooks v. OKbridge, Harold Hooks filed a lawsuit against a commercial Web site, claiming that the company terminated his membership because of his disability. The Western District of Texas dismissed the case because it held that a service provided over the Internet is not a place of public accommodation under the ADA. The Court ruled that public accommodations are limited to physical structures or facilities [63].

  On appeal, the U.S. Department of Justice filed a friend of the court brief[18] [64] with the Fifth Circuit, arguing that public accommodations

[18]amicus curiae (latin for "friend of the court"): the name for a brief that is filed with the court

under Title III of the ADA are not limited to physical locations. The Court of Appeals, however, did not consider the brief of the Department of Justice, rejecting the appeal on grounds that OKbridge was not aware of Hooks' disability when terminating his membership [63].

In Ford v. Schering-Plough Corp, the Third Circuit also held that under Title III of the ADA, places of public accommodation are limited to physical space [62].

NFB v. AOL: In November 1999, the American National Federation of the Blind (NFB) filed a lawsuit against AOL based on the public accommodations requirement under Title III of the ADA, stating that the AOL Web site was inaccessible to the blind. The dispute was settled out of court, and both parties entered an agreement. AOL took a number of measures to make its Web technologies accessible and consented to consult with the disability community on accessibility issues [63].

Martin v. MARTA: In the case of Martin v. Metropolitan Atlanta Rapid Transit Authority (MARTA), a number of disabled individuals claimed that they were unable to look up routes and schedules from the Transportation Authority's Web site. The lawsuit was brought under Title II of the ADA, dealing with the activities of state and government, and more specifically, with the obligation of transit providers. The U.S. District Court found that the MARTA Web site was in violation of Title II of the ADA due to its inaccessibility for the blind. This is the first Federal Court case ruling that Title II of the ADA applies to the Internet [63].

Access Now v. Southwest: A similar case occurred in October 2002. In Access Now v Southwest Airlines, the complainants alleged that the lack of ALT text and other features made the Southwest Web site inaccessible to the visually impaired. They claimed violation under Title III of the ADA, interpreting the Web site as a place of public accommodation. The Court dismissed the case, finding that to fall within the meaning of the ADA, a "public accommodation" must be a physical structure. "To expand the ADA to cover 'virtual' spaces", so the Judge, "would be to create new rights without well-defined standards." [63] This is the only case that directly addresses the applicability of Title III of the ADA to the Internet. The Eleventh Circuit dismissed the appeal because on appeal, the plaintiffs abandoned their old claim and introduced a new claim that was never presented to or considered by the District Court [65].

by someone who is not a party to the case, but who believes that the court's decision may affect its interests.

NFB v. Target: In February 2006, the National Federation of the Blind (NFB) sued Target on behalf of all blind Americans, arguing that target.com is not accessible to the blind and therefore in breach of the ADA, the California Disabled Persons Act and the California Unruh Civil Rights Act. The complaint states that images on the site are missing ALT text, that keyboard controls do not work, and that the site lacks headings necessary for navigation [66]. Judge Patel rejected Target's argument that only physical stores were covered by the provisions of the ADA: "The statute applies to the services of a place of public accommodation, not services in a place of public accommodation. . . . To limit the ADA to discrimination in the provision of services occurring on the premises of a public accommodation would contradict the plain language of the statute". In her September 5th, 2006 order, Judge Patel ruled that to the extent that target.com offered goods and services connected to Target stores, the NFB had a claim under the ADA [67]. In her September 28th, 2007 ruling, Judge Patel granted the case class action status under the ADA [68]. This litigation is still ongoing, and the outcome will likely set a legal precedent for Web accessibility.

Impact. *NFB v Target* has established a claim of Web accessibility under the ADA for Web sites of private businesses insofar as the Web site features are connected to a physical place of public accommodation [68]. The outcome of this litigation will likely set a legal precedent on how the ADA applies to Web sites.

2.7.3. Australia

Legal aspects of Web Accessibility in Australia have been introduced by way of example in Section 1.1.

2.7.4. The European Union

The eEurope Initiative was proposed by the European Commission (EC) in 1999 and adopted by the European Council in Feira in June 2000 [69]. The initiative's objective was "to bring . . . citizen, school and business online and to exploit the potential of the new economy for growth, employment and inclusion" [70].

The eEurope 2002 Action Plan had three main objectives [71]:

1. to make the European Internet infrastructure more secure (secure networks, smart cards), and to provide cheaper and faster Internet access;
2. to invest in the European people and their skills;
3. to encourage the use of the Internet.

Concerning Web Accessibility, the Action Plan states that public sector Web sites of the Member States and European Institutions must be accessible to citizens with disabilities. One of the actions of eEurope 2002 was the adoption of the WAI WCAG guidelines by each Member State by the end of 2001 [71].

In a response to a communication from the European Commission to the Council and the Parliament on the accessibility of public Web sites (EC COM(2001) 529) [72], the European Parliament set forth Resolution (2002) 0325 [73]. Therein, it recognizes the W3C WCAG guidelines as "the global standard for the design of accessible websites", and that compliance with the guidelines will result in little or no cost for Web designers. Furthermore, the communication points out that all public Web sites of EU Institutions and Member States should be fully accessible by the year 2003, and observes that Web sites must be double-A compliant to be deemed as accessible. Finally, the Parliament notes that the promotion of accessibility guidelines for the private sector should start as soon as possible [73].

In January 2003, the eAccessibility Experts Group[19] published its final report on eAccessibility under the eEurope 2002 Action Plan. This progress report pointed out that Member States and EU Institutions had just begun to apply the W3C WCAG guidelines. The guidelines were first to be applied to public sites at the European and national level, with coverage successively progressing to regional and local levels. The experts recognized the need to cover sites which offer commercial and social services to the public, however it remained unclear what role public authorities should have in achieving this. Finally, the report emphasized that the progress of Member States must be monitored in two steps: by addressing efforts made to implement accessibility, and by monitoring the accessibility of key Web sites of the Member States [76].

Several studies have shown that the implementation of previously stated EU policy initiatives on eAccessibility has been going at a slow pace. In December 2002, the Council recognized the need for further action in a Council Resolution on "eAccessibility for People with Disabilities" [77]. Some Member States have gone further in implementing these policies than others, and they have also devised different means of putting them into practice. This leads to similar yet different eAccessibility requirements for products and services in different European countries. The risk for the European industry increased because of market fragmentation, while consumer products risked becoming costlier and incompatible [78].

In a 2005 Communication on eAccessibility (EC COM (2005) 425), the Commission therefore proposed three approaches "not yet widely used in Europe" [15]:

- integrate accessibility requirements in public procurement procedures and policies;

[19]eAccessibility: Accessibility in Information Society, concerned with the integration of older people, people with disabilities and people placed in impairing environments [74]. The eAccessibilty Experts Group was created under eEurope 2002. Its members are experts and representatives from the Member States [75].

- create a certification mechanism to assess the accessibility conformance of products and services;
- explore the potential of existing legislation on eAccessibility.

To support these approaches, the Communication outlined the need for additional measures, e.g. to evaluate and certify the accessibility of public Web sites in the Member States [15].

The purpose of the Communication was to promote the Member States to a harmonization of their accessibility solutions, and to encourage self-regulation of the industry. The Commission announced a follow up on the eAccessibility situation in 2007, looking at the success of this approach, and considering the use of additional measures, including legislation [15].

As part of the follow-up of the 2005 eAccessibility Communication, a study on "Measuring Progress of eAccessibility in Europe" (MeAC) was launched. According to this study, only a very small number of key government Web sites such as national government, parliament and key ministry sites in the Member States met international accessibility standards, and an even smaller number of key commercial sites such as railways, newspapers and TV stations met the guidelines. "Almost all countries have policies in place, in many cases directly triggered by EU-level initiatives such as the Ministerial Resolutions and eEurope", however there are still gaps and there are major differences in the approaches across countries. The study concludes that the impact of EU measures on the accessibility of key Web sites has been very weak, and suggests a number of possible policy options for consideration, including legal and/or regulatory measures [79].

eAccessibility is currently an area of eInclusion policy, defined by the i2010 EU policy framework for the information society and media, which "promotes the positive contribution that information and communication technologies (ICT) can make to the economy, society and personal quality of life" [80].

2.7.5. The United Kingdom

Under the UK Disability Discrimination Act of 1995 (UK DDA), it is illegal to discriminate against disabled persons on grounds of employment, the provision of goods, facilities, and services [81]. The Disability Rights Commission (DRC) was established in 1999 to eliminate discrimination and promote equality of opportunities for disabled persons. It provides legal advice and supports disabled people who bring complaints under the UK DDA [82]. The UK DDA has been amended on several occasions, most recently by the Disability Discrimination Act of 2005 [83]. With the Equality Act of 2006 [84], the Disability Rights Commission (DRC) was merged with other Commissions into the Commission for Equality and Human Rights, which opened on October 1st, 2007 [85].

In December 2006, the DRC published a revised version of its 2002 Code of Practice on Part III of the UK DDA, which deals with the provision of goods, facilities and services. This document states that service providers must not discriminate against the disabled on grounds of their disability, and specifically mentions that it "might be reasonable to provide" accessible Web sites for people with hearing and visual impairments [86].

In March 2003, the DRC announced an investigation into the accessibility compliance of 1,000 Web sites. The selected sites were tested for conformance to the WCAG guidelines using a commercial accessibility evaluation tool. A sample of 100 sites were evaluated by a group of disabled users and accessibility experts. The investigation report, published in April 2004, found that 81% of the sites failed to comply with the "most basic Web Accessibility Initiative category", and only 0.2% of the Web sites were AA-compliant [87].

In its findings, the DRC also states that 45% of the problems encountered by the human evaluators were not covered by the WCAG checkpoints [87]. The Web Accessibility Initiative (WAI) responded stating that the DRC misinterpreted its data, because it failed to take browser and media player accessibility, and the interoperability of assistive technologies into account. The data of the WAI report states that 95% of the barriers reported by the DRC are covered by the checkpoints in the Web Content Accessibility Guidelines and User Agent Accessibility Guidelines [88].

Impact. The DRC did not publish the names of the companies that failed to comply with the WCAG guidelines. That sort of negative publicity could have had some effect and led to public discussion. A year seems a rather long time for an investigation that tested only 100 Web sites and ran 900 through an accessibility evaluation tool [87].

While the DRC may provide legal assistance to the disabled, it may not intervene of its own accord, or present its view of a case to court. The Act and its Code of Practice are designed to incite settlement out of court. The influence of the courts is limited because the Act is a civil law, and companies and organizations might not have enough motivation to make their Web sites accessible [89].

At the time of writing, no case on the accessibility of public Web sites has been brought before court in the UK. It has to be seen whether legal action will force businesses and institutions to take notice of the issue of the accessibility of public Web sites.

2.7.6. Germany

In April 2002, the German Federal Government decreed the Act on Equal Opportunities for Disabled Persons (German: Gesetz zur Gleichstellung Behinderter Menschen und zur Änderung anderer Gesetze). Section 11 of the Act is concerned with “barrier-free information technology” and states that public bodies, such as federal authorities, institutions of the federal administration, federal corporations under public law and public law bodies must ensure that their Web services are accessible to people with disabilities. Paragraph two of Section 11 commits the Federal Government to influence providers of commercial Web sites to make their products accessible [90].

In July 2002, the Ministry of Interior and the Ministry for Labor and Social Affairs issued the Federal Regulation on Barrier-Free Information Technology (German: Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz, BITV) [91], which provides measures for the implementation of Section 11 of the Act. It defines a set of guidelines based on W3C WCAG 1.0 as the technical standard for Web Accessibility[20] [92]. This Regulation is the Federal Republic’s implementation of the Web Accessibility commitments in the eEurope 2002 Action Plan [76, D1.2].

Impact. Existing sites must comply effective from 1st January 2005, while new sites must already conform to the Regulation’s requirements [93]. Legal action may be brought forward by a disabled individual or by acknowledged special interest groups if an organization fails to comply with the regulation [94]. Because no one has yet done anything of the sort, the repercussions of the regulation remain unclear. At the end of 2004, many of the Federal Republic’s public bodies claimed to provide accessibility compliant Web sites which were in fact inaccessible [93].

2.7.7. Austria

In April 2004, the Chief Information Office of the Austrian Federal Chancellery published a report on the implementation of the WAI Web Content Accessibility Guidelines in Austria. According to the report, in 2001, the Federal Chancellery took measures to raise awareness of webmasters and providers of Web content, whereas in 2002, the implementation of the guidelines, conformance level “A”, became a part of the Federal Republic’s E-Government strategy. In April 2002, the Chief Information Office decided that service providers which design and offer Web content for the Federal Government must conform to the WCAG guidelines [95].

In Austria the following laws relate to Web Accessibility:

- The E-Government Act [96]

  The Federal Republic’s E-Government Act (German: E-Government-Gesetz) came into force on March 2004. Part I, Section 3 requires that by January

[20]There are a number of differences between BITV and WCAG 1.0: BITV provides only two priority levels and they differ from the levels provided by WCAG; checkpoints 4.2, 5.3 and 11.1 differ slightly in meaning; checkpoints 11.3 and 11.4 are reversed in order; WCAG checkpoint 2.2 is split into two BITV checkpoints.

2008 at the latest, Internet sites of the public administration must "comply with international standards for access to the worldwide web, including unhindered access for disabled persons." [97].

- Disability Equality Act [98]

  The Disability Equality Act (German: Bundes-Behindertengleichstellungsgesetz) came into force on 1st January 2006. Paragraph 6 Section 5 defines accessibility and states that Information Technology is accessible if it can be used in the generally accepted way, without particular difficulty and especially without help from third parties [98]. The Materials accompanying the Act state that accessibility is to be determined with the current state of technology in mind, and that WCAG Guidelines can be used to determine accessibility of Internet Web sites [99].

  Under Paragraph 6 of the Act, there is no discrimination if the removal of accessibility barriers imposes unreasonable burden (see also Section 1.1.4, "Unjustifiable Hardship"). The legal consequence of discrimination is compensation for damages, however legal action can only be brought to court after an arbitration process [98].

The Chief Information Office monitored the implementation of the WCAG guidelines within the scope of the eEurope 2005 Action Plan. It did not actively check the Web sites of the public administration for accessibility conformance, but requested that the respective authorities provide answers to questions regarding the reorganization of their sites for accessibility conformance [95].

From February to July 2007, the Office of the Chancellor and the Ministries evaluated the accessibility of their Web sites. After an information meeting in March where evaluation methodology was discussed, each Ministry had seven weeks to self-evaluate a sample of its Web site. An average of 11 sample pages were selected by the Ministries for evaluation. All Ministries had to evaluate their sample pages for WAI WCAG 1.0 Level A Conformance; 84% evaluated their pages for AA Conformance, and 68% evaluated their paces for AAA Conformance. According to the final report, 94% of the sample pages are in conformance with WCAG 1.0 Level A [100].

Impact. As of January 2008, The E-Government Act requires Web sites of the public administration to conform to accessibility guidelines. The Disability Equality Act requires Web sites to comply with accessibility guidelines as of January 2006. At the time of writing no legal action has been brought to court under either of these Acts.

2.8. Why should Web Accessibility be important to the average citizen?

We already mentioned some benefits of accessible Web design earlier in this chapter. In Section 2.5.3, we saw that conformance to WCAG guidelines makes Web sites available to all users, may speed up download times and reduce the maintenance costs of a site. In the previous sections we have shown that Web Accessibility is a legal requirement for governmental and commercial sites in some countries. According to the Web Accessibility Initiative, there are other reasons why Web Accessibility should be important to everyone responsible for creating or providing Web content in their daily life:

Moral implications: In Section 2.2 we saw that the percentage of people with disabilities around the world is between 10% and 20%. In Europe, about 15% of the population suffer from some kind of disability, and because populations are aging as we get older, the percentage of disabled individuals is bound to increase. Accessibility to Informations and Communications Technology should be available in the same way ramps, special parking places and disability elevators are provided so that everyone can access public services and facilities. From a moral point of view, it is certainly wrong to exclude those who could potentially benefit the most from this technology.

Business consequences: Web Accessibility is also important in business considerations. According to WAI, by conforming with the WCAG guidelines a site's usability will increase, which in turn will allow more people to use the site, including the disabled [101]. From a commercial point of view, excluding a considerable part of the disabled community from an organization's Web site is a loss of potential customers.

Other benefits, such as a good reputation, are also relevant to business. Demonstrating social responsibility will improve an organization's relationship with the public, which may in turn influence people's buying habits and thus affect profit margins [102].

Technical considerations: By providing metadata, text alternatives, and clear and comprehensive content, search engine listings might be improved [101], while separating structure from presentation and the usage of style sheets will reduce site maintenance, redesign and design changes and thereby improve efficiency. Device-independence will facilitate making Web content available to different devices and operating systems [103].

In general, conformance to accessibility will also increase the usability of a site, but this is not always the case. The most common exception is probably the use of Javascript, which can increase usability (e.g. the validation of a form before submitting it to the server), but may in turn cause problems for users of assistive technologies that do not support it.

A large part of the Web Content Accessibility Guidelines focuses on what is considered "good" Web page design. We believe that sites that already separate structure from presentation and do not use tables for layout may only have to make minor adjustments to their site's structure and add some syntax elements to make them Level "A" compliant. The required changes can be implemented with the next planned upgrades or changes of the site.

If a site does not separate structure from presentation, or uses complex layout tables, the transition would probably require a complete redesign of the existing site to even make it Level "A" compliant. If the site provides non-HTML content, the time and effort required for Level "A" compliance may be substantial. By way of example, consider that WCAG requires synchronized auditory and textual alternatives for audio and multimedia content [16].

2.9. Web Accessibility Evaluation Tools

Web Accessibility Evaluation Tools[21] are programs that help users to determine whether a Web page is accessible. They can assist in identifying and repairing accessibility barriers, and help to improve the overall quality of a page or site. Most tools help users to determine the conformance of Web sites or pages to one or more sets of accessibility guidelines.

Criteria such as cost, deployment and platform, and quality characteristics, such as reliability, flexibility, usability, interoperability, scalability and documentation are common to all software products.

The following criteria apply to all Accessibility Evaluation Tools. They are not restricted to tools available on the Web.

- **Accessibility:** Is the Evaluation Tool accessible for people with disabilities?
- **Guideline coverage:** Which guidelines and national policies does the tool support?
- **Checkpoint coverage:** Which checkpoints of the supported guidelines does the tool implement? [104]
- **Accuracy:** Is the automatic part of the evaluation accurate? Does the tool report false positives, and if so, how many?
- **Completeness:** Is the degree of automation in checkpoint coverage maximized? Does the tool include special algorithms to increase the automatic processing of checkpoints?
- **Customization:** Is it possible to customize specific automated checks, or to suppress them? [104]

[21]Sometimes also referred to as Web Accessibility Evaluation and Repair Tools (AERT).

- **Manual checkpoints:** Does the tool assist the user in manual verification of the checkpoints that cannot be processed automatically? Does the tool allow the user to specify the results of those manual checkpoints, and generate a complete and exhaustive accessibility report?
- **User Interaction:** Does the tool generate an accessibility report with little or no user interaction, or does it guide the user through the process step by step? [104]
- **Repair:** Does the tool assist the user in the repair of accessibility barriers?
- **Page/Site coverage:** Is it possible to scan an entire site, or to group single pages together? Is it possible to create an evaluation report for a group of pages? [104]
- **Report Type:** Does the tool provide a text-only or a graphical accessibility report? Is it possible to generate the report in different formats?
- **Report Presentation:** Is the report well structured? Is it concise and clearly arranged?
- **Character Encoding Support:** Does the tool support only ISO-8859-1, ISO-8859-15 and similar character encodings for western European languages, or does it support encodings for other languages? Does it support variable-width encoding schemes such as UTF-8 and other Unicode character encodings?
- **Special Characteristics:** Does the tool emulate older or text-only browsers, check for readability of page content, or test how a site may look for people with low vision or color blindness?

Summing up, it can be said that Web Accessibility Evaluation Tools can save considerable time and effort for human accessibility evaluators, but they can not replace them.

3. Related work

"The power of the web is in its universality. Access by everyone regardless of disability is an essential aspect." (Tim Berners-Lee)

The main task of this thesis is to implement a Web Accessibility Evaluation Tool. In this chapter, requirements and criteria for the evaluation of other tools in this area are elaborated on, then some of these tools are presented briefly. None of them are Open Source or otherwise make their source code available. At the end of this chapter, evaluation results and an overall impression will be provided.

3.1. Properties common to all tools

All the tools evaluated in this section are free online Web applications. Regarding the criteria outlined in Section 2.9, they share the following common properties:

- **Customization:** It is not possible to customize automated checks or to suppress specific checks.
- **Manual checkpoints:** Although some tools assist the user in the manual verification of checkpoints by presenting the corresponding chunks of HTML code, none of the analyzed tools allow the user to specify the results of those checkpoints. Therefore none of the analyzed tools have the ability to generate a complete accessibility report. This is a feature provided by commercial desktop applications.
- **User Interaction:** All the analyzed tools generate an accessibility report with little or no user interaction.
- **Repair:** None of the tools aid the user in the repair of accessibility barriers. This is a feature provided by commercial desktop applications.
- **Page/Site coverage:** The presented tools scan only single pages. It is not possible to group pages together, or to create an evaluation report for a group of pages.

3.2. Requirements and Criteria for Evaluation

For the evaluation of Web Accessibility Evaluation Tools we use the criteria presented in Section 2.9, except for those already covered in section Section 3.1. In addition, we use the following criteria:

- **Complexity:** How complex is the tool? How much work is required to become familiar with it? How long does it take to understand the accessibility evaluation it provides?

- **Comprehensibility:** Is the evaluation report the tool provides comprehensible? Are the algorithms the tool uses published? Is it possible to reconstruct the tests that are being applied, and thus assess their validity?
- **Documentation:** Does the tool provide documentation? If so, how complete and accurate is it? How comprehensible is it?

3.3. Assessment of Web Accessibility Evaluation Tools

The Web Accessibility Evaluation Tools described in this section were found in an extensive search with the Google Web search engine, the World Wide Web Consortium's (W3C) homepage and various sites on Web Accessibility that were found with Google in the first place. We only considered free online Web Accessibility Evaluation Tools.

The text in the "Description from the Web site" sections are direct excerpts from the Web sites of the respective tools.

3.3.1. Bobby Online Free Portal

Version and License:

URL: Formerly `http://bobby.watchfire.com/`, no longer available

Version: 4.0.1

License: Bobby End User License Agreement

General information: Bobby was the first Web Accessibility evaluation tool, released in August 1996. Created by a non-profit organization named CAST, it later became a commercial product, owned by the Watchfire Corporation [105]. The free online version of Bobby has since been discontinued and replaced by Watchfire WebXACT.

3.3.2. Watchfire WebXACT

Version and License:

URL: `http://webxact.watchfire.com/`

Version: not available

License: free for personal use, see homepage for details

Description from the Web site: *"WebXACT checks one page of web content at a time and reports results immediately through intuitive Web-based reports that help expose website quality, privacy and accessibility defects. By testing* *pages on their site with WebXACT, developers can encourage compliance with industry standards and best practices. Upon entering a URL, WebXACT results are immediate and returned in a summary report that show a snapshot of a page's quality, accessibility, and privacy health. ...*

The Quality Status Report in WebXACT explains the page's quality issues and indicates whether it has defects like broken links or anchors, warnings, or issues with the page. Users benefit as WebXACT identifies common issues that can drive users off a website. The WebXACT Accessibility Report summarizes the accessibility issues on the page to help determine if that page can be accessed by individuals using assistive devices such as screen readers, and facilitates compliance with the U.S. Government's Section 508 and the World Wide Web Consortium's (W3C) Web Content Accessibility Guidelines (WCAG). The WebXACT Privacy Report indicates whether or not the page may have a link to a privacy statement, identifies information collection, visitor-tracking technology like cookies and web beacons, P3P[22] compliance and third party links. WebXACT privacy reports are a good first step to help companies understand some of the potential privacy issues that need to be addressed on websites." [106]

General information: WebXACT is a free online service provided by the Watchfire Corporation. It allows users to test single pages of Web content for quality, accessibility and privacy issues. WebXACT promotes other Watchfire products, such as WebXM, WebQA, and Bobby (commercial Desktop version).

Overall impression: WebXACT's features are not limited to testing for Web Accessibility. A WebXACT scan uses a default set of accessibility options, which can be customized on the "Advanced / Accessibility Options" page. Among other things, the advanced options allow the user to choose between WCAG 1.0 and Section 508 guidelines.

The WebXACT report contains a header section, followed by four tabs to organize the test results: a general tab, a quality tab, an accessibility tab, and a privacy tab.

The header includes the URL of the page, time and date, a link to the "Advanced / Accessibility Options", and an input field to quickly scan another page. Our evaluation focuses on the Accessibility Tab.

The Accessibility Report: The Accessibility Tab shows the accessibility issues on the current page. During a scan, WebXACT gathers information about conformance to WCAG 1.0 or Section 508 guidelines. The report lists accessibility errors and warnings found during the scan. The top of the accessibility report contains a summary on compliance with automatic and manual checkpoints. The rest of the report consists of a detailed list of checkpoints, organized by priority level when testing for WCAG conformance, and category.

[22]Platform for Privacy Preferences (P3P) Project, `http://www.w3.org/P3P/`.

An automatic checkpoint is considered an “error”, a manual checkpoint a “warning”. The report includes the checkpoint number, its description, the number of instances on the site and the line numbers in the HTML source code. “Expand code fragments” links, which are provided for each priority level and category, allow the user to display code fragments in the report. Each checkpoint links to WebXACT online documentation.

WebXACT does not provide syntax or error highlighting. It splits code fragments into element name, attribute name, and attribute value.

The WebXACT accessibility report is very well structured and clearly arranged. Expanding code fragments makes it easy to associate a checkpoint with the HTML code that caused an error or warning.

Character Encoding Support: WebXACT serves its reports as UTF-8, which is also specified as the report's character encoding via the HTTP Content-Type header. The HTML code of the report however specifies windows-1252 as the document character encoding via a `meta` declaration with `http-equiv` set to `Content-Type` and a value for `charset`. Although this is not entirely correct and may confuse some user agents, according to the HTML specification the HTTP Content-Type header takes precedence over the `meta` declaration [39].

By performing tests with a number of character encodings[23], we infer that WebXACT transcodes Web pages to UTF-8 to be able to include content from the evaluated page in the report.

Extreme test cases, such as when an invalid encoding is specified via the HTTP Content-Type header, or when no encoding is specified via either HTTP Content-Type or the `meta` element, are decoded and displayed successfully; this leads us to conclude that in such cases, WebXACT uses a number of heuristics to determine the character encoding of a document.

Documentation: The documentation is very extensive. WebXACT provides a summary of tests that are performed for each Guideline. The online documentation also specifies whether a test belongs to the “error” or “warning” category.

Completeness: Some issues that could be processed automatically are not, for example WCAG 1.0 checkpoint 3.2 [16], which requires documents to validate to published formal grammars. There are free HTML validation services that could be used to automatically detect conformance to this checkpoint.

WebXACT raises some manual checkpoints even when they do not to apply to the page in question. For instance, checkpoint 13.1 requires the target of each link to be clearly identified. WebXACT may raise this issue even if there are no links on a Web page.

[23]Among the character sets tested were gb2312 (Simplified Chinese), Windows-1255 (Hebrew), UTF-16, KSC_5601 and EUC-KR (Korean), Shift_JIS (Japanese), and Windows-874 (Thai).

Summary: WebXACT is a free online service. It is not limited to accessibility evaluation and offers other tests that are important for Web developers. The accessibility report is well structured, and the documentation is very extensive. WebXACT is the only evaluated tool written with non-western languages and character encoding issues in mind. It is very well implemented in this regard, and handles even extreme test cases without problems.

3.3.3. Cynthia Says Online Portal

Version and License:

URL: `http://www.cynthiasays.com/`

Version: n/a

License: free for personal use, see homepage for details

Description from the Web site: *"The HiSoftware Cynthia Says portal educates users in the concepts behind Web site accessibility. The simple, well-designed interface puts accessibility compliant code within the reach of all users, even those with little or no knowledge of Web design. The Cynthia Says portal provides feedback to users in a reporting format that is clear and easy to understand. Accessibility issues are detected within web sites from Web-based applications, dynamic pages, or static HTML pages. Users get an immediate "status" of their Web site accessibility. To find these, Cynthia Says utilizes HiSoftware's AccMonitor Server technology, through which USER Agents (crawlers/scanners) collect individual page or dynamic page accessibility data. This information is then sent to the central server where actual accessibility verification is performed. The output is returned immediately to the user's browser.*

The Cynthia Agent tests your page against programmatic test groups for Section 508 or W3C WCAG 1.0 Accessibility Guidelines. Please remember that in addition to these checks there are checks that must be manually performed. These manual tests are listed in your report!" [107]

General information: The Cynthia Says portal is a joint Education and Outreach project of the Internet Center for Disability Resources on the Internet (ICDRI), the Internet Society Disability and Special Needs Chapter, and HiSoftware [107]. The free Online Portal validates one page at a time. HiSoftware provides commercial products with more functionality [108].

Overall impression: The Cynthia Says Online Portal checks at most one page per minute from a particular client. It can evaluate pages for compliance with WCAG 1.0 or Section 508 guidelines. For WCAG 1.0 guidelines, it allows to test for compliance with conformance levels A, AA and AAA. The "full options" page features advanced settings of page evaluation, among them a report on potential blinking or moving page content.

The Accessibility Report: Upon submission of a Web page, Cynthia will generate an extensive report, consisting of a header, the actual accessibility report organized in tabular form, and a small footer.

The header of the report includes the URL of the verified site, date and time when it was tested, a summary on automatic verification ("passed" or "failed") and, if applicable, the specific emulated Web browser[24], e.g. "Netscape 6.0".

The main part of the accessibility report consists of at most three sections, depending on the selected guideline set. For Section 508 guidelines, the report consists of one section; for WCAG 1.0 guidelines, the report is made up of one section for each priority level.

Each section of the accessibility report is displayed as one large table with four columns. The leftmost column of the table describes the checkpoints in that section, and the remaining columns show the result of the conformance test. Values of "Yes" and "No" apply to checkpoints that can be automatically verified. A value of "N/A" means that the checkpoint does not apply to the page because no related HTML elements or attributes were found in its source code. The checkpoints that do not show any results are manual checks, whose conformance must be verified by the user. "N/V" indicates checkpoints that have not been selected for verification.

Within a section or table, checkpoints are grouped together by subject, e.g. "Basic", "Image Maps", "Tables", and "Frames". Cynthia splits checkpoints into several rules, each of which may or may not apply, depending on page content. The descriptions of the checkpoints in the table cells include links to WAI WCAG 1.0 or U.S. Section 508 for more information.

Cynthia Says provides line and column numbers for each instance of automatically detected errors. The HTML source code of the page can be included at the bottom of the report. Cynthia does not provide any syntax or error highlighting, nor does it link the line and column numbers in the table cells to the corresponding HTML code on the bottom of the report.

Character Encoding Support: Cynthia Says serves its reports as Windows-1252, specified via `meta` declaration. When the user choses to include the source code of the evaluated page in the accessibility report, it becomes apparent that Cynthia Says has been written without non-western character encodings in mind: everything is interpreted—and displayed—as if it were Windows-1252.

[24]Some Web pages return different content depending on the name and version of the user agent. Because all returned Web pages must be accessible, some evaluation tools allow the user to specify how they should identify themselves to the server.

Documentation: The documentation for Cynthia includes a variety of help pages, tutorials and even instructional videos. It provides a chart for how WCAG 1.0 and Section 508 checkpoints are implemented as well as a comparison of Cynthia with the now discontinued Bobby Online Portal.

Completeness: The Cynthia Says Online Portal provides a description of the algorithms and tests that are made for each checkpoint. There are some issues that could be determined automatically, but are not, e.g. WCAG checkpoint 3.2.

Summary: Cynthia provides a text-only accessibility report, there is no option for graphic visualization. The report always includes all checkpoints, whether or not they apply to a page. The documentation is adequate. Cynthia does not provide its own online reference for the checkpoints, but links to the appropriate section of the WCAG Web site instead. Cynthia does not support character encodings other than Windows-1252[25].

3.3.4. Torquemada

Version and License:

URL: `http://www.webxtutti.it/testa_en.htm`

Version: 2.0

License: not available.

Description from the Web site: *"Torquemada offers to website developers a complete methodology for accessibility analysis which uses a tool for page checking that makes it possible to quickly identify which parts of a page are in error and the HTML code corresponding to these parts."* [109]

General information: Torquemada is under development by Andrea Bernardini.

Overall impression: Torquemada is a free online accessibility evaluation tool that checks a given Web site for compliance with the WCAG 1.0 guidelines. It does not provide advanced customization. Torquemada offers three different types of report: a text-only version and two graphic reports that require recent browsers and Javascript support.

The tool is available in both English and Italian, however at the time of writing only the Italian version is operational; the English version returns HTTP Status 500 Internal Errors.

[25]In practice, a number of similar character encodings will work unless the Web page uses any one of the code points in which they differ. For instance, Windows-1252 maps the Euro sign to code point 0x80. Windows-1252 uses code points 0x80 to 0x9F for letters and punctuation, while ISO-8859-1 uses those code points for control codes.

The Accessibility Report: The text-only report consists of three sections: the header, the test results, and the HTML source with one HTML element per line. The header consists of the page URL, time and date when the page was tested, and the Torquemada version. The test results are grouped into checks. Each check description includes the HTML element or attribute the check refers to, a comment, the number of instances on the page, and a list of those instances. Each instance consists of a line number and is a reference to the corresponding HTML code in the third part of the report. All Torquemada checks fit into either of the "error" and "warning" categories. The comment describes the error or warning and is in itself a link to a WAI technical document with more information.

Both graphical reports consist of three frames: the upper frame contains the same header and test results sections as the text-only report. The lower right frame contains the HTML source code of the Web site under verification, the lower left frame displays the Web site itself. The line instances in the upper frame are Javascript links to the respective areas in the lower frames. When a line instance link is activated, the corresponding parts of the Web site and its source code are shown in the lower frames. The difference between the "light" and "full" graphical reports is that the light report only jumps to the corresponding areas, while the full report highlights those areas with rectangles and bounding boxes.

When displaying the HTML source code, Torquemada strips the original HTML code of comments and some elements, and displays one HTML tag per line. Text content between the opening and closing tag of an HTML element is displayed on the same line as the opening tag. There is a bug where text content between tags is sometimes displayed on a line by itself without a line number. The line numbers in Torquemada's report do not correspond to the line numbers of the original HTML source code.

Character Encoding Support: Torquemada serves its reports as ISO-8859-1, specified via `meta` declaration. The HTML source code of the evaluated page is included in all three report types, and is interpreted as ISO-8859-1, regardless of its character encoding. Torquemada seems to have been developed without character encoding issues in mind.

Documentation: Although the Torquemada interface and the generated accessibility report is available in both Italian and English, usage instructions are available only in Italian. Other than that, there is no documentation. Although the text-only report does not specify which WCAG 1.0 checkpoint a check refers to, the comments supply links to a corresponding WAI document with more information.

Completeness: The algorithms and tests Torquemada applies are neither described nor listed on its Web site. This makes it difficult to determine how complete the accessibility report is. Furthermore, Torquemada offers little aid in visual and manual verification of WCAG checkpoints, because it only lists automatically detectable errors and warnings.

The software seems to be unfinished and incomplete in other respects as well. In some cases, the current version provides a warning about no `tabindex` being specified for `a`, `area`, `button`, `input`, `label`, `legend` or `textarea` elements, and provides the HTML element on line 1 as the only occurrence in the instance list. This is confusing at best, because one would expect Torquemada to list all line instances of these elements. Torquemada produces this warning even if the page contains none of these elements. There are instances of errors that should be classified as warnings, and constitute false positives.

Summary: Torquemada can be used as a complement to other accessibility evaluation tools. The documentation and the software are unfinished and incomplete, and there is room for improvement for some of its features. Torquemada provides graphical as well as text-only accessibility reports. Torquemada does not support character encodings other than ISO-8859-1[26].

3.3.5. The WAVE 3.0

Version and License:

URL: `http://www.wave.webaim.org/`

Version: 3.0

License: free for personal, non-commercial use, see homepage for details

Description from the Web site: Only an explanation of the WAVE icons is available [110].

General information: The WAVE is a free online service provided by WebAIM (Web Accessibility In Mind), which is a non-profit organization within the Center for Persons with Disabilities at Utah State University.

Overall impression: The tool tests single Web pages for compliance with WCAG 1.0 and Section 508. The WAVE's accessibility report is graphic/icon-based. A "Change Preferences" page allows the user to choose between WCAG and Section 508 guidelines, to select which elements of the original page to display, and to customize the WAVE features to include in a scan. The preferences are locally stored in a cookie, the tool provides no user management.

The development version of the WAVE can simulate a text-only view and display an outline of the page based on header levels. At the time of writing,

[26]ISO-8859-1 uses code points between 0x80 and 0x9F for control codes, which cannot be displayed, while Windows-1252 uses the same code points for letters and punctuation. Most modern browsers will therefore decode ISO-8859-1 using Windows-1252.

version 3.5 of the WAVE has been under development for over 2.5 years.

The Accessibility Report: The WAVE is a graphical/icon-based evaluation tool that integrates report icons within the Web page it evaluates. The icons are color-coded; red denotes errors that should be fixed while yellow denotes possible errors. The ALT text of an icon displays additional information about the problem that needs to be addressed. In some cases, textual information is included next to the icon. The WAVE modifies the links on the page so that the activation of a link will result in a WAVE scan of the link target.

The problem with this approach is that the WAVE employs a high number of icons. Even on a small or medium-sized page, the amount of information presented to the user can be overwhelming, and icons may overlap. The information provided by the ALT text of some of the report icons is rather short and not self-explanatory, while other icons do not provide any ALT text at all. These issues make it difficult to use the WAVE for accessibility evaluation without investing a considerable amount of time and effort to learn the meaning of icons and ALT text.

Character Encoding Support: The WAVE serves its reports as ISO-8859-1, specified via the HTTP Content-Type header. The WAVE report is a modification of the HTML code of the Web page being evaluated, with additional text and images added in the appropriate sections. An existing `meta` declaration with `http-equiv` set to `Content-Type` and a value for `charset` is not modified; if the value of `charset` differs from “ISO-8859-1”, this introduces an inconsistency with the value of the HTTP Content-Type header.

Since the accessibility report is always served as ISO-8859-1, the HTML source code is also interpreted as such. If the user evaluates a Web page not encoded in ISO-8859-1 (or a superset thereof), he is likely to see a lot of placeholders, question marks, or random characters instead of the original content.

Documentation: Other than an explanation of the icons that are used in the Accessibility Report, there is no documentation. This tool would benefit from usage instructions, a how-to, and a description of the features on the “Preferences” page and how to use them.

Completeness: The algorithms and tests the WAVE applies are neither described nor listed on its Web site. This makes it difficult to determine how complete the accessibility report is. Moreover, the WAVE offers little aid in visual and manual verification of WCAG checkpoints, because it only lists automatically detectable errors and warnings.

Summary: The WAVE uses an original concept to present accessibility issues on a Web page. However, the icon-based accessibility report seems rather overwhelming at first. The tool provides no text accessibility report. The documentation and the ALT text of the icons could be improved substantially. The WAVE does not support character encodings other than ISO-8859-1.

3.3.6. Other tools

The presented tools are a subset of free online evaluation tools. A number of similar tools such as TAW (CTIC Foundation), Web Accessibility Checker (ATRC, University of Toronto), Imergo online (FIT) and Hera (Sidar Foundation) are also available[27].

3.4. Evaluation results

The most powerful tools are definitely WebXACT and Cynthia Says. Both offer a clearly laid out accessibility report, provide usage instructions as well as documentation on reviewing the generated reports, and explain their more advanced features. Both Cynthia and WebXACT provide a text-only accessibility report. The WebXACT report is visually elaborate and well structured, whereas the Cynthia report presents its results in a tabular form. WebXACT has a better visual presentation of results and better documentation, and it supports character encodings.

Torquemada is unique in that it offers three types of accessibility reports, both graphical and textual. Unfortunately, it seems to be unfinished in some respects and lacks in documentation and usability. The WAVE employs an icon-based accessibility report, which takes some getting used to, and provides no textual report. Neither Torquemada nor the WAVE aid users in the visual and manual verification of WCAG checkpoints and only list automatically detectable errors and warnings. This is a drawback because some manual tests depend on site content, and human evaluators would benefit from knowing whether specific manual tests have to be performed for a given Web page. Neither tool supports character encodings.

WebXACT and Cynthia are more thorough than the other tools in that they list *all* checkpoints in their report. According to the Web sites of the Watchfire Corporation [111] and HiSoftware [112], the commercial versions of the respective products assist the user in the verification of manual checkpoints.

[27]A list of evaluation tools can be found at `http://www.w3.org/WAI/ER/tools/complete`. Only a subset of the tools listed are free online evaluation tools.

4. Task and Requirement Analysis

In Chapter 3 we have seen the functionality provided by some free online Web Accessibility Evaluation Tools. This chapter outlines the rationale for the implementation of another tool and describes the design principles and architecture of the software.

4.1. Task outline

4.1.1. The Integration Approach

The original task description for this thesis was based on an an integration approach. It consisted of the following basic steps:

- to implement a framework designed to integrate the results of existing tools into another application;
- to write an application using that framework to include the results of several existing tools;
- to implement some additional accessibility conformance checks to add new functionality that existing tools did not have.

In reviewing existing tools such as Bobby[28], WebXACT [113] and Cynthia Says [108], this approach of creating a new evaluation tool was found to be unfeasible:

- The Bobby license[29] specifically prohibits using Bobby in relation to third party Web sites, to provide third parties with scan results, or to create "derivative products" based on Bobby.
- The WebXACT Terms of Use contain similar provisions [114];
- The Cynthia Says Terms of Use state that the portal must be used via the Cynthia Says homepage, or by using the form provided to add the Cynthia Tester to another Web site. They specifically forbid automatic processing for other web services [115].

It was reasonable to assume that if the integration framework approach was pursued, the operating entities of the respective tools would eventually restrict access to clients processing their results. Apart from the legal aspects and potential countermeasures, it would not be efficient to include tools like Cynthia Says [108], which allows only one evaluation per minute from a particular client. These tools are self-contained and do not allow integration with other tools.

[28]Now discontinued; see Section Section 3.3.1
[29]This license is no longer available online.

4.1.2. The Reimplementation Approach

We abandoned the idea to implement a framework to integrate other tools and decided to re-implement the accessibility evaluation functionality of existing tools instead. Although various tools also implement Section 508, our application focuses on implementing the Web Content Accessibility Guidelines. Because the WCAG document [16] is meant to be stable across evolving Web technologies, it does not provide information on how to implement its guidelines and checkpoints. The accompanying technical documents are not exhaustive and merely provide implementation examples for HTML [49] and CSS [116].

4.2. Goals of the Thesis

In the previous chapter we have seen the features that are provided by some free online tools. The reason for implementing another application is to offer new features that existing tools in the same area do not provide at all, or not to a sufficient extent.

After deciding that we would have to re-implement existing functionality, our first priority was to create an evaluation methodology. The practical part of our work would be to implement that methodology as a library, and to create a rich web application on top of that library.

4.2.1. Creating an Evaluation Methodology

To evaluate Web sites for conformance with the Web Content Accessibility Guidelines 1.0, it was first necessary to create an evaluation methodology, which is described in the next chapter,"An Evaluation Methodology for the Web Content Accessibility Guidelines 1.0". Please note that although conformance to some WCAG checkpoints can be evaluated automatically using software algorithms, many require human judgement by evaluators who have a thorough understanding of Web Accessibility, Web technologies, evaluation tools, assistive technologies, etc.

4.2.2. Implementation

The practical part of our work includes the implementation of the evaluation methodology. Our focus is to implement the methodology as a library with a public API that may be used by other applications, with special emphasis on character encoding support so that it will support non-Western character encodings. The library must be designed to support multiple evaluation guidelines, and the conformance checks should be implemented as part of a data structure so they may be easily iterated, and new checks may be plugged in without the need to substantially alter existing code. The library must also provide a means to export its result in XML.

The second part of our practical work is the implementation of a Web application on top of the evaluation library. The Web application will provide the following features:

User Interface: The application provides the user interface to evaluate Web sites for conformance with the evaluation methodology.

Generate Accessibility Reports: The application must process the results provided by the library and display them as an initial HTML Accessibility Report. This report only covers issues that can be checked automatically.

Provide different 'Views': The application should allow the user to display the evaluation data in several different ways.

Store Evaluation Results: Evaluation Results must be stored for reference and verifiability.

Save evaluated Web pages: The application must store the Web sites it evaluates for future reference, in case the correctness or the accuracy of the generated accessibility reports are challenged.

Character Encoding Support: The application must be implemented with character encoding support in mind. It should serve its pages in a Unicode character encoding and must support a variety of character encodings other than ISO-8859-1, ISO-8859-15 and Windows-1252.

User management: The application should allow unregistered users to perform accessibility evaluations, similar to other online tools. However, some of the features described in this section might be reserved for registered users.

User documentation: The application should provide static pages that explain the evaluation methodology and all its conformance checks.

4.3. Architecture

In this section, we outline the basic basic architecture and design of our application. We use free / open source software (FOSS) for our implementation because it is freely available, extensible, and allows third parties to reproduce and verify the implementation.

The architecture of the application is shown in Figure 1.

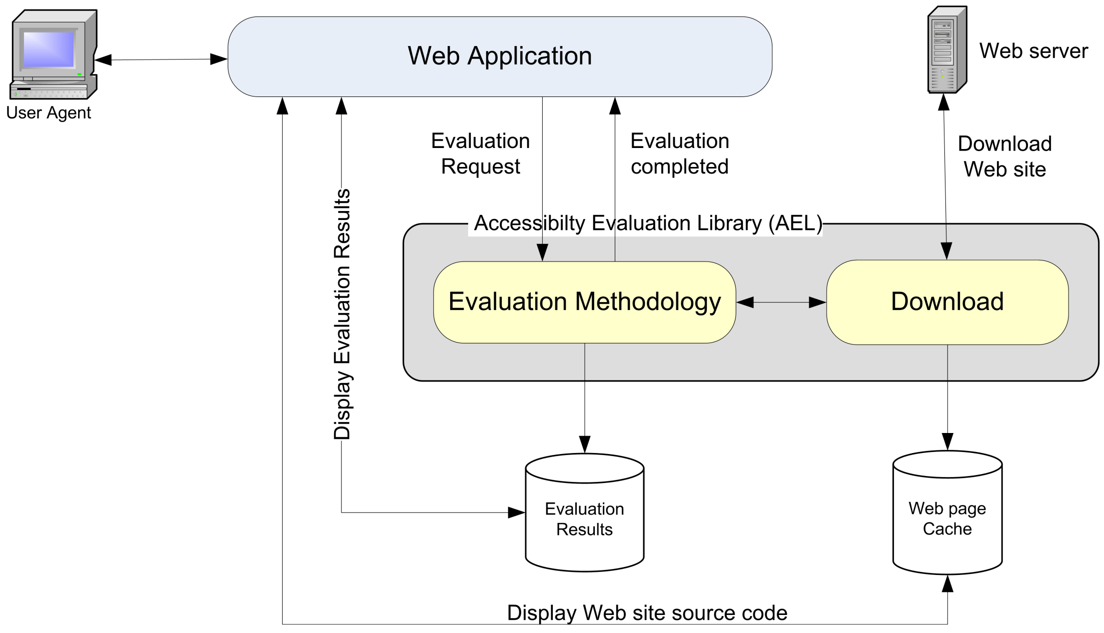

Figure 1: The architecture of the evaluation tool

4.3.1. Typical Workflow

The following is a typical workflow for an evaluation request:

1. The user accesses the Web application via his user agent and requests an evaluation of *http://www.example.com* by submitting a form.
2. The Web application sends an evaluation request to the Accessibility Evaluation Library (AEL).
3. AEL's Download subsystem will fetch *http://www.example.com* and associated external files (style sheets, scripts, frames, etc) and store a local copy of the Web site in its Web page cache.
4. AEL's Evaluation Methodology subsystem will apply evaluation criteria to the retrieved page and store the results.
5. AEL signals the Web application that evaluation is complete.
6. The Web application retrieves the results and generates an HTML accessibility report. From there, the user may chose to display the downloaded source to verify the evaluation results.

Should an error occur during evaluation, AEL would return an appropriate error code and/or message to the web application, which would react accordingly (log the error, display a suitable error message to the user and possibly notify the administrator).

4.3.2. Software components

For each component, we must determine how it should be implemented and whether existing software packages can be used to build upon to avoid duplicating existing work and reduce development time. The application will be developed for a UNIX-like environment.

Accessibility Evaluation Library (AEL): AEL is the implementation of the Evaluation Methodology and its Techniques. The programming language used to implement AEL must provide methods to fetch and store remote Web sites, parse HTML and CSS, and perform numerous text processing operations. Text processing is best done with a robust regular expression-engine, therefore the programming language must support regular expressions.

AEL will be invoked by the Web application, fetch and locally store the Web page to be evaluated, check for conformance with the Evaluation Methodology, store the results in XML in the Evaluation Results storage, and notify the Web application that the evaluation has been completed.

Web application: We will use an existing Web application / content management framework to build our Web application. The framework must be able to generate and publish Web content online, manage users (authentication, authorization), support a templating system, support workflows for reviewing and publishing documents, be extensible (by loading plugins, modules, or add-ons) and most importantly provide a powerful and well-documented API. The programming language the framework is written in must be able to execute other processes and communicate with them and provide a robust XML parser.

Web page cache: The evaluation tool will save a local copy of all evaluated Web pages in a Web page cache so that evaluation results may be checked for correctness, allowing us to ensure correct operation of the evaluation tool. The user will be able to access the cache for an evaluated page from the HTML accessibility report for that page.

The Web page cache is stored on the file system. To minimize file system access and increase performance it is important that the web application maintains a cache of pages displayed to the user.

Evaluation Results storage: Evaluation results are stored on the file system. In practice this storage will probably be combined with the web page cache. Access to the storage should likewise be minimized by implementing a page cache at the Web application level.

The Accessibility Evaluation Library and the Web application are software components.

5. An Evaluation Methodology for the Web Content Accessibility Guidelines 1.0

In Section 4.2.1 we have seen that to implement our tool, we first need to create an evaluation methodology based on the W3C Web Content Accessibility Guidelines 1.0 [16].

The WAI Web Content Accessibility Guidelines 1.0 document is meant to be stable across evolving Web technologies and therefore does not provide information about an implementation of its checkpoints. A separate set of technical documents discusses each checkpoint in more detail and provides implementation examples for HTML [49] and CSS [116].

Even so, the WAI documents merely provide implementation examples. There is at this time no WAI resource that provides complete and accurate information about the implementation of the WCAG 1.0 guidelines. WAI does, however, provide a W3C working draft from April 2000 on "Techniques for Accessibility Evaluation and Repair Tools" [117] which "describes techniques that Web Accessibility validation tools may use to evaluate the conformance of HTML documents for conformance to the Web Content Accessibility Guidelines 1.0 (WCAG 1.0)". Unfortunately, the document is an early draft, and incomplete in regards to techniques to check Web pages for WCAG 1.0 conformance.

5.1. Methodology definition

Our evaluation methodology provides a series of "techniques" for automatic and manual verification of the conformance of a Web page to WCAG 1.0. Our interpretation of the guidelines meets the following requirements:

- Conformance to WCAG 1.0 and related Techniques documents as well as other applicable W3C Recommendations.
- The conformance evaluation algorithms offer only one possible interpretation.
- The conformance evaluation algorithms target specific technologies (HTML, CSS) but are independent from specific programming languages.
- The degree of automatic evaluation is maximized.

The methodology is designed to evaluate Web pages based on HTML 4.01, XHTML 1.0, XHTML 1.1 and CSS Level 2.

5.2. Limitations

5.2.1. Limitations of accessibility guidelines

Accessibility guidelines are an attempt to define accessibility barriers, however no single set of guidelines can cover all possible accessibility barrier scenarios experienced by a disabled person. Moreover, the WCAG checkpoints defined in each guideline "explain how the guideline applies in typical ... scenarios", which implies that a guideline may not be fully covered by its checkpoints. We acknowledge that although our methodology is a best effort to maximize coverage of WCAG 1.0, it too will only cover a subset of all possible Web Accessibility barriers.

5.2.2. Limitations of automatic testing

The accessibility of a Web page cannot be determined by algorithms alone. Human review is necessary to determine conformance to a number of checkpoints. Automatic testing alone can identify some specific issues on a Web page, however is not a reliable indicator of a Web page's overall conformance to accessibility guidelines.

Automatic testing of CSS is more problematic than automatic testing of HTML:

- The user may have turned off style sheets;
- Different subsets of CSS are supported by different user agents[30];
- Some CSS rules in a style sheet may not apply to a HTML page;
- The user may override some CSS rules using a user-defined style sheet;
- Style sheets may apply to different media types.

5.2.3. Other limitations

A number of issues are not addressed by this methodology.

- It is limited to the evaluation of a single Web page for WCAG 1.0. Evaluation results from multiple pages of a Web site can be combined to provide a better overall result, however this is outside the scope of our methodology.
- Each conformance test either applies to specific page content, or it does not, meaning that all conformance test results are boolean. The methodology does not address to which degree specific conformance tests constitute an accessibility barrier or indeed whether they constitute an accessibility barrier at all: it merely interprets WCAG 1.0 with regards to (X)HTML and CSS.

[30]For example, CSS attribute selectors are not supported by Microsoft Internet Explorer up to and including version 6.

5.3. Evaluation techniques for WCAG 1.0

Creating a list of conformance tests for WCAG 1.0 was one of our first tasks when writing this thesis. The list is extensive and can be found in Appendix A, "Techniques for Evaluation of Conformance to WCAG 1.0".

5.3.1. Source of conformance tests

The conformance tests are based on the "Techniques for Accessibility Evaluation and Repair Tools" draft [117] and include information from the following additional sources:

- the "Web Content Accessibility Guidelines 1.0" [16];
- the "Techniques for Web Content Accessibility Guidelines 1.0" [49] and associated documents;
- the HTML 4.01 [39] and CSS Level 2 [118] specifications;
- the WCAG 1.0 curriculum [119];
- an analysis of the tools described in Chapter 3;
- a number of other online resources.

5.3.2. Legal considerations

According to the W3C Intellectual Property FAQ [120], part of our "Techniques for Evaluation of Conformance to WCAG 1.0" (see Appendix A) is an annotation of W3C documents that does "not require the copying and modification of the document being annotated".

Disclaimer: We are not associated with the W3C or any other body governing accessibility guidelines. Our work has not been endorsed or sponsored by the W3C or any other third party.

Our "Techniques for Evaluation of Conformance to WCAG 1.0" (see Appendix A) therefore clearly references W3C documents and other sources, including a link to the original document. The W3C document license is included in Appendix C.

5.4. Other Methodologies

At the time we started our work, we were unable to identify any person or organization who had created an evaluation methodology for WCAG 1.0 that was publicly available (in English), and therefore we set out to create our own methodology and started implementing it (for implementation details, see next chapter, "Implementation of Web Accessibility Evaluation Tool").

Examples of organizations that have developed their own evaluation methodologies include Accessiweb in France, Technosite in Spain, the Bartimeus Accessibility Foundation in the Netherlands and AnySurfer in Belgium[31]. In a joint effort of 23 European organizations participating in three projects, efforts were made to develop a Unified Web Evaluation Methodology (UWEM).

5.4.1. Unified Web Evaluation Methodology (UWEM)

UWEM is a methodology to evaluate conformance of Web sites with WCAG 1.0 developed by the EU Web Accessibility Benchmarking Cluster (WAB Cluster). Its aim is "to increase the value of evaluations by basing them on a shared interpretation of WCAG 1.0 and a set of tests that are sufficiently robust to give stakeholders confidence in results". UWEM 1.0 was released in July 2006. [121].

UWEM is developed by three European projects under the WAB Cluster [122]:

The EIAO project: *The EIAO project will establish the technical basis for a European Internet Accessibility Observatory. Frequently updated assessment data will be available online from a data warehouse providing a basis for benchmarking, policymaking, research and actions to develop accessibility to Internet.* [123]

Support-EAM: *Support-EAM is an IST funded project ... under the Sixth Framework Programme of the European Commission. It started on the 1st October 2004 for 22 months and ended on 31st July 2006.*

The objective of Support-EAM (Supporting the creation of a e-Accessibility Quality Mark) was to propose a strategy for creating a Web Accessibility Quality Mark for Web services, as part of the Action Plan eEurope 2005: An information society for all. [124]

BenToWeb: *BenToWeb was a project within the Web Accessibility Benchmarking (WAB) Cluster aimed to support the European public and private sector to implement the recommendations of the eEurope 2005 Action Plan by providing new software modules and methodologies that satisfy some of the accessibility recommendations of the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C), which were not analysed by existing tools due to their inherent complexity.*

BenToWeb supported the objectives of the Cluster in regard to the creation of a validated methodology to test Web sites. [125]

[31]Some of these methodologies are available on the respective Web sites, but not in English.

Unfortunately our evaluation methodology was already implemented at the time UWEM 1.0 was released.

6. Implementation of Web Accessibility Evaluation Tool

In this chapter we describe the major components of our implementation.

6.1. Requirements

The goal of the practical part of this thesis is to implement a free online Web Accessibility Evaluation Tool. As we saw in Chapter 4, this task can be divided into two subtasks represented by two major software components:

1. The Web application: Provides users with user documentation, a Web interface to request evaluation of Web pages, sends evaluation requests to AEL and provides results in the form of an HTML accessibility report;
2. The Accessibility Evaluation Library (AEL): Evaluates Web pages according to the evaluation methodology presented in Chapter 5 and returns the results to the Web application.

6.2. Technologies used for the implementation

We used the following technologies to implement our tool, and to write this thesis:

Accessibility Evaluation Library: To implement AEL, we chose Perl 5.8 [126] as well as a number of Perl modules to parse HTML and CSS etc. Perl is ideally suited for text processing and provides a large number of modules through the Comprehensive Perl Archive Network (CPAN).

Web application: We chose the Drupal content management system [127] to implement our Web application. We use the Apache HTTP Server 2.2.x [128] as the Web server and MySQL 5.x [129] as the Database server for Drupal, as well as PHP version 5.2.x [130].

Source Code Management We initially used CVS [131] and later switched over to Subversion [132] to manage our source code.

Project Management: To keep track of issues, bugs, features, releases etc we use the Trac Project [133], which provides Subversion integration.

Project Development: We chose the Eclipse IDE [134] in conjunction with a number of plugins to write Perl, PHP and LaTeX, edit HTML and CSS files, as well as Subversion integration. This document is typeset in LaTeX.

Note: Our choice of Drupal is unrelated to it being written in PHP[32]. We evaluated a number of open source content management systems, including Mambo / Joomla, Typo3, Wordpress, and Plone, and chose Drupal for its ease of use, robust API, user and session management.

6.3. System Design

In Section 4.3 we have seen a proposed basic architecture for our implementation. In this chapter we expand on our initial analysis and describe the specific system design.

We named the Web application "EOWA" for "Evaluation Of Web sites for Accessibility"[33]. A simplified system design is outlined in Figure 2.

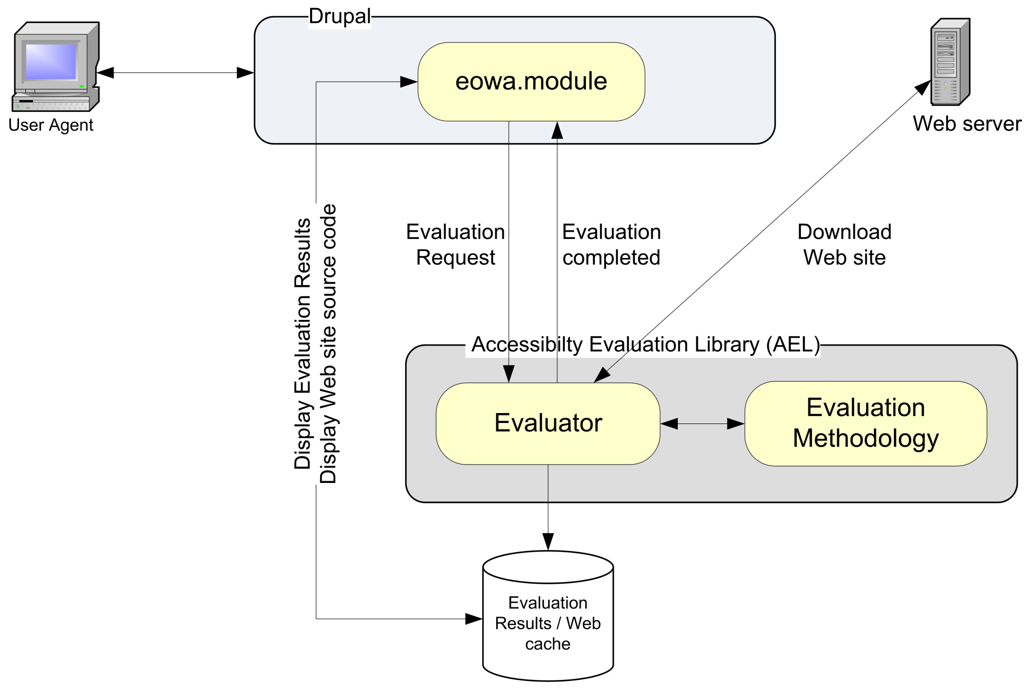

Figure 2: Simplified System Design

The system design differs somewhat from our proposed architecture in Section 4.3: The proposed "Download" component is integrated into a main AEL component called "Evaluator" which provides an interface to the caller. The evaluation conformance tests are implemented in a separate component. We also combined the

[32]We find PHP 5 lacks important language features such as support for namespaces, closures, and dynamic scoping. It also provides way too many functions that perform similar yet different operations, which are inconsistently named. However, we also find the Drupal CMS provides features that outweigh our reservations against the PHP programming language.

[33]The name was inspired by the availability of the domain name *eowa.org*.

storage mechanisms for the Web cache and the Evaluation Results. We describe the two major components in more detail in the next sections.

6.4. The Web application

EOWA has been implemented as a Drupal module called eowa.module. At its core, a Drupal module is a set of PHP functions. Since PHP does not provide namespaces, a function can access all other functions in the system. The idea of a module, or plug-in, or add-on is generally to extend the functionality of a core software package. To achieve this, a module must be able to influence the processing workflow of the main application and possibly other modules. Drupal uses a set of defined hooks to "allow modules to interact with Drupal core" [135]. Hooks are places in the code where the Drupal core calls a module's exported function if a module provides that hook. To implement a hook, e.g. *hook_menu*, a module implements a function called *modulename_menu*.

6.4.1. Third party components

The Web application uses a number of Drupal modules to provide site features such as a Site Map, generation of printer-friendly versions of Web pages, and advanced contact forms.

The EOWA module uses Nigel McNie's syntax highlighting library GeSHi [136] to highlight (X)HTML, CSS and Javascript files.

6.4.2. Module components

The module code has been split into several files to provide another layer of modularity but also to increase performance by loading specific code only if it is needed. A block schematic of the main components of the EOWA module is shown in Figure 3.

1. **eowa.module:** The main file includes all hooks the module implements.
   hook_perm defines permissions for our module; *hook_cron* is used to periodically check the Web cache / Evaluation Results repository for orphaned data, which can happen if an anonymous user has performed evaluations and their user session expires.

   hook_menu enables a module to register URL paths to be handled by one of its functions. Depending on the type of registration, a menu item is placed in the navigation menu. This hook is used to dynamically load code, depending on the current URL path; e.g. if the URL path starts with */admin*, the admin UI is loaded. Error Handling is described in more detail in Section 6.7.

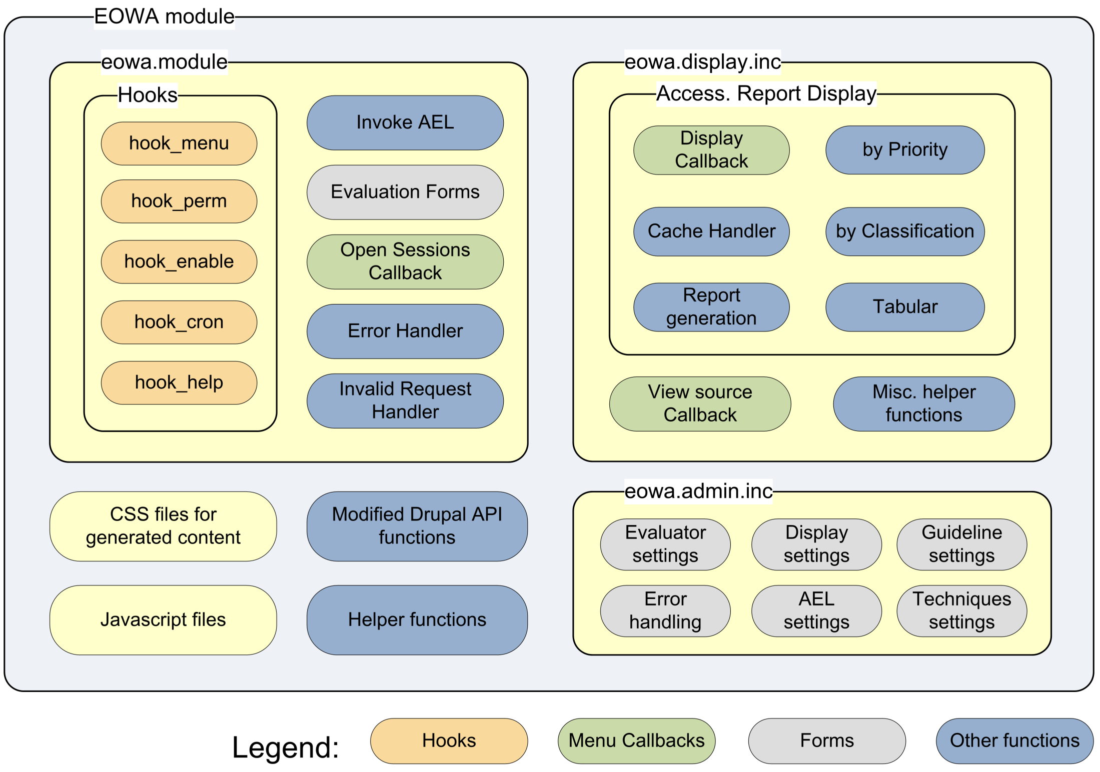

Figure 3: The main components of the EOWA module

eowa.module also defines the functions that invoke the Accessibility Evaluation Library (AEL) and the HTML forms used to submit URLs for evaluation.

2. **Admin UI:** The admin UI allows the administrator to configure the following settings:

Evaluator: The administrator may specify where the evaluation results and cached Web pages will be stored, whether orphaned evaluation data is to be deleted as well as how long accessibility reports should be cached.

Error Handling: The administrator can define the URL path for custom error pages for a number of error cases. See Section 6.7 for more details.

Display settings: This page lets the administrator specify how overly long code lines or string sequences are handled in the HTML accessibility report. It also includes syntax highlighting settings for the display of downloaded code (HTML, CSS, script).

AEL settings: These settings determine the path to the AEL library, the Perl executable, a Perl include path if the library is not installed system-wide, etc.

Guideline / Technique settings: This page of the admin interface lets the administrator activate or deactivate the techniques implemented by AEL, define the short description that is displayed in the HTML accessibility report, and specify a link to a page that provides further information on the technique.

3. **Display functions:** The display functions are the heart of the EOWA module, as they generate the HTML Accessibility Report from the XML data provided by the Accessibility Evaluation Library. They also allow the user to display the HTML source code of the evaluated page, as well as external content such as linked scripts or frames. The Accessibility Report is described in more detail in Section 6.8.

6.4.3. Presentation

Drupal's presentation layer is a Web template system called the theme system. Template rendering is performed by template engines; Drupal's default engine is PHPTemplate. In Drupal each theme can control most of Drupal's output and load its own style sheets. Some Drupal functions provide default HTML output that can be overridden at the theme level by theme functions. The EOWA module implements all functions that generate HTML output as theme functions so that they may be overridden by other modules or themes.

Drupal sub-themes inherit all their files from their parent themes. To create a subtheme is as simple as overriding the parent's style sheets. The EOWA site design is implemented as a sub-theme of the highly customizable Zen theme for Drupal [137].

6.5. The Accessibility Evaluation Library (AEL)

The Accessibility Evaluation Library has been implemented as a set of object-oriented Perl modules. To provide its functionality, a number of third-party modules are used.

6.5.1. Third party modules

HTML-Tree: HTML-Tree is a module suite that creates parse trees out of a HTML source. It mainly consists of the HTML::TreeBuilder and HTML::Element modules.

HTML-Encoding: HTML-Encoding helps determine the character encoding of (X)HTML and XML documents.

LWP-UserAgent: The module implements a simple web user agent that we use to fetch network objects such as HTML, CSS and script files.

XML-Generator: We use XML-Generator to export the evaluation results in an XML document, which is then processed by the Web application.

Encode: Provides the Perl Encoding API, which we use to transcode fetched documents to UTF-8 so that the Web application only needs to process one type of character encoding.

WebService-Validator-HTML-W3C: The module provides access to the W3C online validator[34] (by default) or another validator installation[35]. We use the validator to implement Technique 3.2.1, which requires documents to validate to published formal grammars (see Section A.7).

6.5.2. Module components

A schematic of the major module components can be seen in Figure 4.

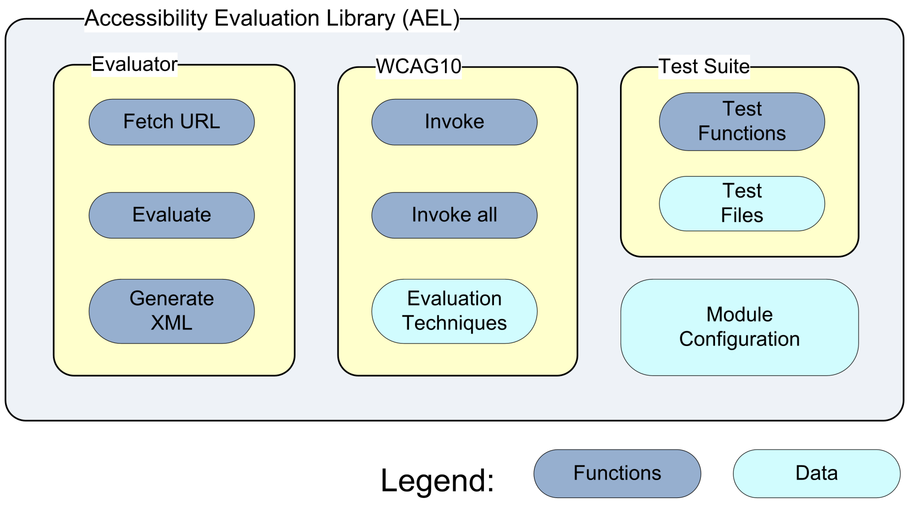

Figure 4: The main components of the Accessibility Evaluation Library

Evaluator: The Evaluator module provides an interface for software using AEL. An Evaluator object initializes a new HTML parser, fetches the URL submitted for evaluation, transcodes it to UTF-8, parses the HTML, initializes a new WCAG10 guideline object and instructs it to perform an evaluation.

WCAG10: AEL is designed to support multiple guideline implementations, although currently only WCAG 1.0 is implemented. WCAG10 provides two public subroutines: one to invoke a specific evaluation technique, and one to invoke all available evaluation techniques.

[34]The W3C Markup Validation Service is located at `http://validator.w3.org/`.
[35]The source code of the W3C Validator is available under the W3C Software License.

Evaluation techniques are implemented as a hash table using the technique id as the key. The value of the hash is a data structure that includes a reference to an anonymous subroutine, which implements the technique. This data structure allows the module to easily iterate through all implemented techniques.

Test Suite: The module includes a test suite that covers all implemented techniques. The test suite is executable, and integrated into the Perl testing framework. The test suite currently includes over 800 unit tests that cover success, failure and special cases for every implemented technique.

6.6. Typical Workflow

This section provides a typical workflow for an evaluation request. The request is divided into two phases: the Processing Phase and the Display Phase.

6.6.1. Processing Phase

The Processing Phase is shown in Figure 5, which shows only the relevant module components. A typical workflow consists of the following steps:

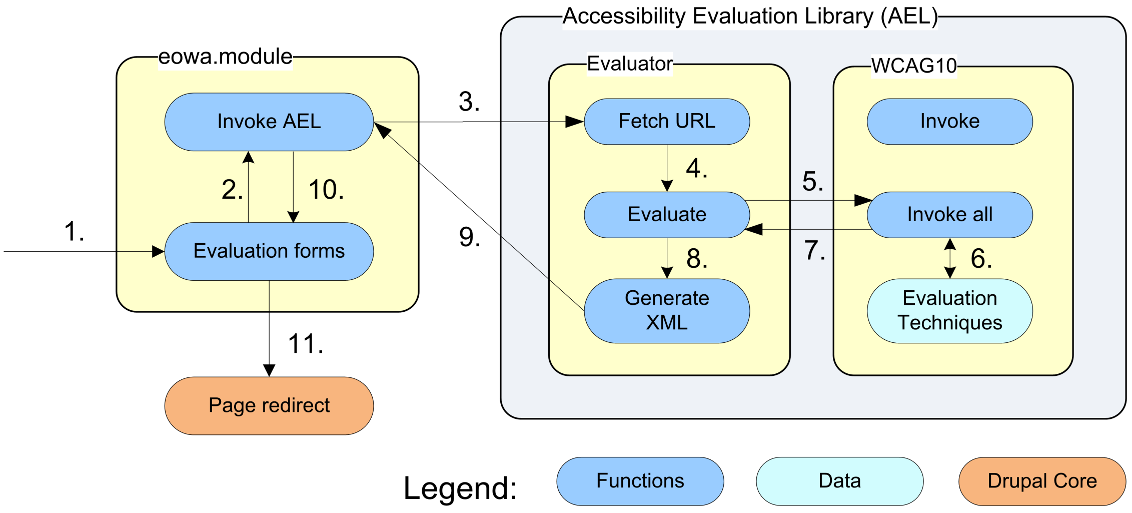

Figure 5: Evaluation Request Workflow, Processing Phase

1. The user submits an URL for evaluation using a Web form. The functions that define the Evaluation Form validate the submitted string ensuring it is a valid *http* or *https* URL.
2. The Web application generates a unique alphanumeric storage ID, which is passed on to AEL along with the submitted URL. The storage ID will be used

to store evaluation results and cached Web pages on the file system. AEL is invoked, and the storage ID and the URL are passed along as parameters.

3. The Evaluator fetches the URL and performs the following checks:
   a) The page must not exceed a defined maximum size;
   b) The fetched content must be an (X)HTML document, as indicated by the HTTP request's Content-Type header, if available;
   c) The character encoding of the document, determined using the HTTP-Encoding module, must be supported.

   If all conditions are satisfied, the Evaluator proceeds to transcode the page into UTF-8 using Perl's Encode module and parses the Web page using the HTML-Tree suite.
4. The Evaluator then invokes the WCAG10 module to evaluate the accessibility conformance of the page.
5. WCAG10 iterates through its evaluation techniques data structure.
6. The results are returned to the Evaluator.
7. The Evaluator invokes XML Generation.
8. An XML representation of the evaluation results is created, and stored at the location determined by the storage ID. AEL exits, control is returned to the Web application.
9. Any errors returned by AEL are handled at this point. EOWA checks the storage to make sure the processed data is available, and saves data associated with this evaluation, such as storage ID and URL, in the user's session. Control is returned to the Evaluation Forms.
10. The Evaluation Form returns the relative URL `eval/view`.
11. The Drupal Forms system uses the return value to construct an absolute URL such as *http://eowa.org/eval/view*, ends the page request and issues a HTTP redirect to that URL[36].

6.6.2. Display Phase

The user agent receives the HTTP Location header from the server, and sends a request for `http://eowa.org/eval/view`. At some point during the page request, the Drupal menu system invokes the EOWA menu hook, which checks the request

[36]Internally, `header()` is used to send a HTTP "Location: *URL*" header with status code "302 Found" to the user agent.

URL, notices this is a request to display evaluation results, loads the display functions from `eowa.display.inc`, and registers a number of URL paths with the menu system.

At a later point in the page request, Drupal invokes the display callback function registered with the menu system for the URL path `eval/view`. This function processes AEL's evaluation results and generates an HTML accessibility report. A typical workflow can be seen in Figure 6, which shows only the relevant components. It consists of the following steps:

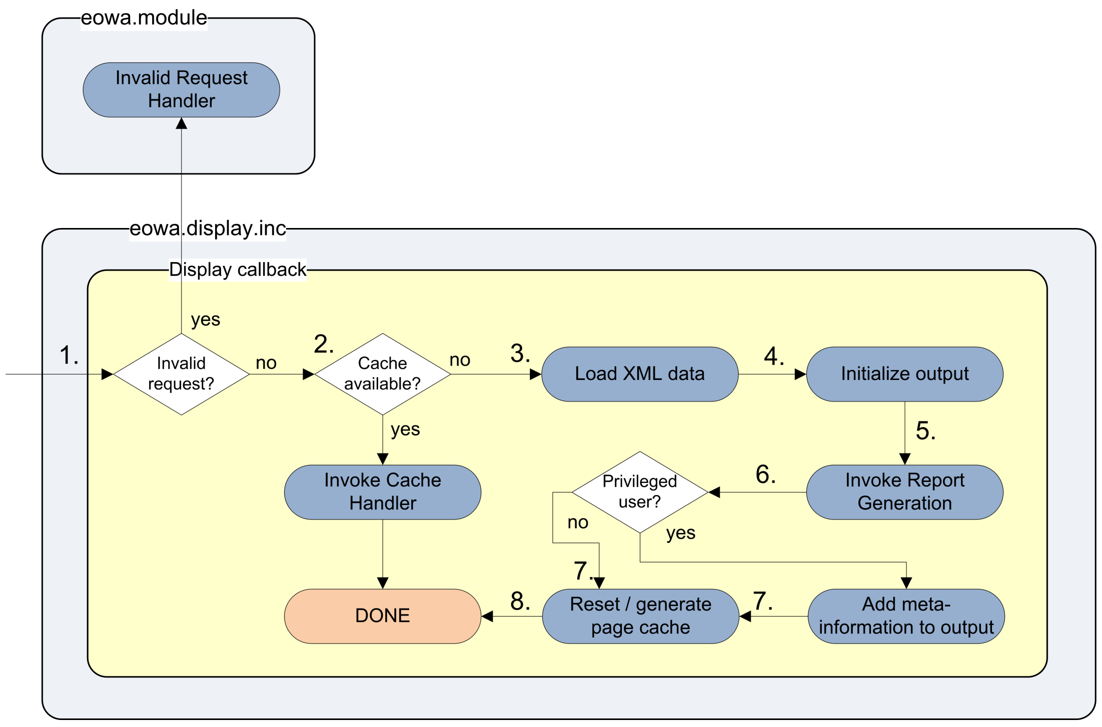

Figure 6: Evaluation Request Workflow, Display Phase

1. Is the page request invalid?

   Yes: Invoke Invalid Request Handler from *eowa.module*.

   No: Continue processing (go to step 2).

2. Is a cached version of the requested page available?

   Yes: Invoke Cache Handler from *eowa.module*, return output. *DONE*.

   No: Continue processing (go to step 3).

3. Load the XML file using the simpleXML PHP extension.

4. Initialize the output: set the page title, add CSS and Javascript files to the page, etc

5. Invoke Report Generation functions.
6. Is this a privileged user?

   Yes: Add meta-information to the page output, then continue to step 7.

   No: Continue processing (go to step 7).
7. Cache the output processed so far using Drupal's Caching API.
8. Return output. *DONE.*

6.7. Error handling

Errors are handled differently at different levels of the application, and a number of different types of errors are recognized. AEL logs Perl warnings and reports the following errors to EOWA, which are handled by EOWA's Error Handler[37] from Figure 3.

Invalid request: This error is reported if the submitted URL was not a valid "http" or "https" URL.

Fetch URI failed: This error is triggered if the document could not be retrieved, e.g. because access is denied, the document was not found, the server was not found, etc.

Maximum content size exceeded: AEL imposes a maximum size on the documents it retrieves. This error is triggered if that limit is exceeded.

Invalid media type: The (X)HTML documents fetched by AEL must be valid HTML or XHTML media types (see [138] for details).

Encoding not supported: This error is triggered if the specified encoding is not supported by the system.

Decoding content failed: This error is triggered if an error occurred when decoding the document, e.g. the document is in the specified encoding but it contains invalid characters.

HTML Parser error: This error is triggered if the HTML parser fails to parse the specified document.

Others: Internal errors are caught and also reported to EOWA.

In addition to the errors reported by AEL, the Error Handler is invoked if cookies are disabled[38], and in various non-recoverable error cases such as failure to write to

[37]The Error Handler logs and optionally redirects the user to a custom error page. Invalid requests are an exception in that they are not logged because they are expected to happen too regularly; instead of a custom error page, errors are displayed inline.

[38]Drupal needs cookies to establish a user session. EOWA needs user sessions in order to operate correctly.

or read from the filesystem.

The administrator can create custom error pages using Drupal, and specify the URL for the error page in the EOWA admin UI.

6.8. The Accessibility Report

This section describes the HTML accessibility report generated by the EOWA display functions.

Our Accessibility Report is made up of several sections: a header that includes the page title and navigation as well as an an evaluation summary, a set of links that present the user with actions she can take[39], and a section with detailed evaluation results.

The Accessibility Report display page allows the user to represent the evaluation data in several different ways: the results can be grouped by Priority, by Guideline number, or not at all, which is called a “Tabular” view. These representations are implemented by applying different transformations to the XML data, which are in turn represented by the “by Priority”, “by Classification” and “Tabular” functions from Figure 3, respectively.

By default, the evaluation results in each of these three views is sub-grouped by category (“Errors”, “Warnings”, and “Need manual verification”, respectively). The user may also chose to display the data without sub-grouping, which is called the “compact” display mode. This makes for a total of six distinct representations of the evaluation data.

In addition, the default view does not display HTML or CSS code fragments, but only the number of times a technique applies to the code (“Instances”). The user may display these code fragments by selecting “Show Code Instances” from the set of action links. Code instances can only be displayed in “category” display mode, e.g. when sub-grouping by category is in effect.

6.9. The Source Code View

From the Accessibility Report’s set of action links (“Available actions”), the user may choose to display the evaluated HTML source code, as well as any external CSS or script files (if any).

By default, the source code is syntax highlighted using Nigel McNie’s GeSHi library [136], however a non-highlighted version is also available.

[39]The availability of some of these actions depend on the user’s permissions.

6.10. Implementation Problems and Limitations

This section outlines some of the problems we came across when implementing our tool, and describes its major limitations. Most of the problems we encountered are related to the limitations or bugs of existing tools, plugins or modules.

6.10.1. Processing HTML

The HTML-Tree Perl suite is very robust and stable, however it has a number of minor drawbacks[40]:

1. The output of the generated parse tree may differ somewhat from the original document. Whitespace and newlines aside, this is mostly due to implicit end tags, omitted tags, and HTML Entities being converted to their corresponding Unicode characters[41].

   The source code fragments AEL reports as a result of WCAG 1.0 conformance tests are generated from the parse tree. This implies that the code fragments may differ from the code in the original document in regards to HTML Entities, omitted and implicit tags[42]. For this very reason, AEL stores both the original and the parsed HTML source code in its Web page cache; the Web application allows advanced users to display both versions.
2. The nodes of the parse tree do not include the line and column numbers where the element occurred in the original document. Line and column numbers could help evaluators to find code instances more quickly; unfortunately such a feature is not provided by the HTML-Tree module suite.
3. A known HTML-Tree bug: processing instructions in the preamble of the document end up under the HTML element. We therefore chose not to store PI's, which is somewhat problematic for XHTML documents.
4. There is a bug in HTML-Tree where an implicit body tag is opened for frameset documents, which causes the generated parse tree to be invalid HTML.

6.10.2. Processing CSS

There are a number of CSS-related modules for Perl, but an analysis has shown that none suit our requirements. For instance, one of the parsers of the "CSS" suite

[40]HTML-Tree uses the Perl HTML-Parser suite to actually parse HTML documents. Some of the following observations are in fact limitations of HTML-Parser.

[41]In a parse tree output, users of the HTML-Tree suite may choose which entities to encode. However, as we do not know which HTML Entities are encoded in the submitted document it is not possible to produce exactly the same output. Similar reasoning applies to omitted and implicit tags.

[42]To "normalize" the code, we configured the suite to generate XHTML-compliant code.

does not support "at-rules" such as @media or @charset, which leads it to ignore @import rules where one style sheet is imported into another. Other parsers in the suite suffer from similar limitations.

To correctly process CSS, cascading and inheritance need to be taken into account, and it must be determined which CSS rules apply to the given HTML document. At the time of writing there is no Perl module with these features[43].

Due to the limitations of existing modules, we implemented our CSS checks using regular expressions. As a result, CSS checks may sometimes be triggered for CSS rules that do not apply to the Web page being verified.

6.10.3. Character Encoding Detection

According to the HTML specification, the value of the "charset" parameter in the HTTP "Content-Type" header takes precedence over that of a `meta` declaration with "http-equiv" set to "Content-Type" [39]. For XHTML documents, the document encoding is specified on the XML declaration using the "encoding" attribute. XHTML documents which set the encoding explicitly (as opposed to using the HTTP "Content-Type" header) must include it both in the XML declaration and in a `meta` element, with the value of the XML declaration taking precedence [40].

Due to differences in HTML and XHTML processing rules, in special cases a certain character encoding would need to be assumed to parse enough of the document to determine whether it is HTML or XHTML, and the results of this detection mechanism would determine the processing rules used to process the document and determine the character encoding in the first place. It is therefore impossible to write a detection algorithm that will always determine the correct character encoding [139]. We use the HTML-Encoding Perl module to determine the document character encoding, using ISO-8859-1 as a fallback encoding in case none is specified.

6.10.4. Character Encoding Support

Our evaluation tool has been developed with character encoding support in mind. To evaluate a document, it has to be converted to UTF-8 for the following reasons:

- Unicode is the only standard that can express most of the world's writing systems;
- The Accessibility Report can optionally include code fragments from the evaluated document. A HTML document may only contain text represented by

[43]The CSS-Tiny-Style module can match selectors to specific HTML-Element objects, but not to the HTML document as a whole. It does not handle inherited properties and is based on CSS-Tiny, which comes with its own set of limitations.

a single character encoding; the code fragments must be represented in the same character encoding as the Web application.

- Drupal uses UTF-8.

This means that all documents that are saved in the Web page cache, including external scripts and style sheets, must be converted to UTF-8. Perl has its own internal format to store text strings in memory and provides an Encoding API to convert strings to and from the internal format into a specific character encoding. PHP does not.

PHP has no native support for Unicode; indeed, a character is the same as a byte [140]. Some Unicode support is provided by functions that convert UTF-8 to and from ISO-8859-1; Windows-1252 and ISO-8859-15 are not natively supported.

The PHP mbstring (multi-byte string) extension provides some support for Unicode-based encodings such as UTF-8 and UCS-2. As most PHP applications are not written to work with multi-byte character encodings, PHP may be configured to overload a number of standard string and regular expression functions.

Since Drupal is meant to run on as many platforms as possible and the mbstring extension is not always available, it is designed to run without it if necessary and has implemented its own wrapper functions to provide UTF-8 support if mbstring is not available. This means that Drupal does not operate correctly if function overloading is enabled.

Moreover, mbstring only provides replacement functions for a small subset of string and regular expression functions. PHP provides the "preg" and "ereg" regular expression engines, and "mb_ereg", which is similar to ereg but with multi-byte support. The preg suite is generally superior both in features and speed, so we could not use ereg.

Although preg does support Unicode, its matching functions only accept or return zero-based *byte* offsets from the beginning of the string being matched. This makes it impossible to use regular expressions to perform some text processing operations with a UTF-8 string without writing special wrapping or helper functions.

Finally, when including code fragments in the Accessibility Report, we were faced with the following problems:

- A code fragment may contain excessively long "words" (long character sequences without white space) that could expand beyond the width of the browser window and disrupt the visual presentation of the Accessibility Report.

- A code fragment may contain too many lines to be displayed in the report.

- A code fragment may contain too many characters to be displayed in the report.

Pre-processing of code fragments was therefore necessary to ensure the Accessibility Report would not be visually disrupted. Since code fragments are strings that may include HTML Entities (which represent a single character) as well als multi-byte characters in UTF-8, PHP's lack of Unicode support was a significant obstacle.

7. Evaluation of our Implementation

This chapter takes a look at our conformance to the goals we set for this thesis in Section 4.2.

7.1. Evaluation Tool Review

We first take a look at our tool using the Access Tool Reviewer (ATR) by Chris Ridpath[44]. ATR is a tool to evaluate accessibility evaluation tools. ATR provides a number of test files the reviewer runs through the accessibility evaluation tool under review. The reviewer then marks each test as either 'passed' or 'failed'.

The tool includes 261 test files, which cover only a part of the techniques implemented by our methodology. The following table gives a list of the 63 tests (24.1%) our tool failed or that are otherwise noteworthy:

Type of failure	Number of test files
Test file / suite error	25
Different interpretation	17
Not (fully) automatable	18
Not applicable	2
Inherent complexity	1
Sum	63

There are a number of errors in the test files. For instance, test file 5-1-2-f5 should trigger the check "Data tables must have at least one complete row of headers or one complete column of headers" (WCAG 1.0 checkpoint 5.1). Our tool implements this as Technique 5.1.1 "Data tables must have row and/or column headers". The problem is that the test file contains a table that has a complete row of headers, so this check is not triggered by our tool. Either the test file is wrong, or the check is meant not to apply.

The other big group of results are checks that are not (fully) automatable. We assume the author included these types of checks to see whether the evaluation tool implements heuristic algorithms. Our tool offers some heuristics, but they may give false positives in some cases, and some of these issues are hard to detect with any degree of confidence. Moreover, the results have to be checked by a human evaluator in any case, so we did not include too many heuristics in our methodology. An example is checkpoint 12.3: "Divide large blocks of information into more manageable groups where natural and appropriate". The test files 12-3-1-f1 and 12-3-1-f2 are virtually identical; yet the first is supposed to trigger, while the second should not. Our tool always includes this technique because this needs to be checked manually by the person evaluating the Web page.

[44]ATR can be downloaded from `http://www.aprompt.ca/ATR/GetAtr.html`.

The third largest group of results are differences between our interpretation of WCAG 1.0 and that of the author of ART. An example of such a test are 1-1-1-f5 and 1-1-1-f6: ART allows empty (NULL) ALT text, or ALT text consisting only of white space, while our methodology does not.

“Not applicable” refers to layout tables; we believe tables need not be used for layout any longer. “Inherent complexity” refers to checkpoint 2.2; see next section.

When Test suite / file errors and not fully automatable checks are not taken into account, it essentially comes down to how WCAG 1.0 are interpreted, and applied to (X)HTML and CSS. This reinforces the notion that human judgment is necessary to evaluate the overall accessibility of a site.

Detailed results of the test can be found in Appendix B.

7.2. Evaluation Methodology

We created and implemented an Evaluation Methodology for WCAG 1.0 according to the goals we set out in Section 4.2. The evaluation methodology includes 114 conformance techniques. With the exception of technique 2.2.1 all techniques were implemented as part of the WCAG10 component of the Accessibility Evaluation Library (AEL) shown in Figure 4 in the previous chapter.

To implement Technique 2.2.1 it is necessary to determine foreground and background color for all elements to determine whether they provide sufficient contrast. Since no Perl module was available to determine which CSS rules apply to the parsed (X)HTML document[45], we did not implement Technique 2.2.1.

A summary of our work in this area can be seen in the following table:

Goal		Status
Create Evaluation Methodology		*completed*
Implement Evaluation Methodology	Structure & API (Evaluator)	*completed*
	Document Retrieval	*completed*
	Conformance Checks	*all except T 2.2.1*
	Character Encoding Support	*completed*
	XML Export	*completed*

[45]See Section 6.10.2.

7.3. Implementation of Web application

All tasks have been completed:

Goal	Status
User Interface	*completed*
Generate Accessibility Reports	*completed*
Provide different 'Views'	*completed*
Store Evaluation Results	*completed*
Store evaluated Web pages	*completed*
Character Encoding support	*completed*
User management	*completed*
User documentation	*completed*

8. Conclusions and Future Work

8.1. Summary

At the beginning of this thesis our goal was to implement a free online Web accessibility evaluation tool by creating an evaluation methodology, implementing it as a library and creating a Web application with a number of features.

In chapter “Concepts and Terminology” we introduced concepts relevant to Web Accessibility, including accessibility barriers, the Web Content Accessibility Guidelines 1.0, legal aspects, HTML concepts, and evaluation tools. The “Related Work” section evaluates existing work in this area.

Our implementation can be summed up in the following steps:

1. Create an Evaluation Methodology.
2. Design and implement a library framework for the Evaluation Methodology.
3. Design and implement the Evaluation Methodology as a dynamic data structure, including a test suite.
4. Design and implement the Web application as a Drupal module.

The Web application can evaluate Web sites for conformance with our Evaluation Methodology for the Web Content Accessibility Guidelines 1.0 (WCAG 1.0).

The application attempts to maximize automatic checks and implements a number of techniques that other tools have not previously implemented. It provides a number of different representations of the evaluation results, allows the user to view the source code of the evaluated document (including all externally referenced files) and supports multiple character encodings.

8.2. Future work

“To strive, to seek, to find, and not to yield.” (Alfred Lord Tennyson)

We have successfully implemented a stable initial version of our tool, and are planning a number of improvements and updates for the next version.

1. AEL:
    - Bring our evaluation methodology closer to the Unified Web Evaluation Methodology (UWEM).
    - Determine if another tool can be integrated to evaluate Technique 2.2.1, *or*

- Determine if an existing Perl CSS Parser can be integrated with the HTML-Tree suite to implement Technique 2.2.1.
- Create a test suite for our Evaluation Methodology using external HTML and CSS test files instead of internal data structures.
- Better character encoding detection: implement heuristic if specified encoding is incorrect or if no encoding is specified.

2. Web application:
   - Allow users to override the results of existing techniques.
   - Allow users to create a complete accessibility report, including the results of “manual” techniques, and publish it online.
   - Allow users to define sets of techniques they wish to override or leave out (evaluation profiles).
   - Allow direct input of HTML code.

References

[1] The Access Board, "ADA-ABA Accessibility Guidelines," Aug. 2005. http://www.access-board.gov/ada-aba/final.htm.

[2] Miguel Angel Cabra de Luna, "International Workshop on Accessibility Requirements for Public Procurement in the ICT Domain," 2004. http://europa.eu.int/information_society/policy/accessibility/regulation/pubproc_ws_2004/a_documents/eesc-speech.pdf.

[3] WebAIM, "Introduction to Web Accessibility," 2005. http://www.webaim.org/intro/.

[4] Parliament of Australia, "Disability Discrimination Act 1992," 2005. http://scaletext.law.gov.au/html/pasteact/0/311/top.htm.

[5] Human Rights and Equal Opportunities Commission, "Human Rights and Equal Opportunities Commission Website," 2008. http://www.hreoc.gov.au/.

[6] Human Rights and Equal Opportunities Commission, "Complaints under the Disability Discrimination Act," 2008. http://www.hreoc.gov.au/complaints_information/DDA_complaints.html.

[7] Hon. William Carter QC, "Bruce Lindsay Maguire v Sydney Organising Committee for the Olympic Games," 2000. http://www.hreoc.gov.au/disability_rights/decisions/comdec/2000/DD000120.htm.

[8] T. Worthington, "Olympic Failure: A Case for Making the Web Accesible," 2001. http://www.tomw.net.au/2001/bat2001.html.

[9] UN Ad Hoc Committee on a Comprehensive and Integral International Convention on the Protection and Promotion of the Rights and Dignity of Persons with Disabilities, "The Concept of Reasonable Accommodation in Selected National Disability Legislation," Jan. 2006. http://www.un.org/esa/socdev/enable/rights/ahc7bkgrndra.htm.

[10] Hon. William Carter QC, "Maguire v SOCOG3," 2000. http://www.hreoc.gov.au/disability_rights/decisions/comdec/Maguire%20v%20SOCOG3.htm.

[11] Australian Government, Department of Broadband, Communications and the Digital Economy, "Reviews and inquiries," Nov. 2006. http://www.dbcde.gov.au/communications_for_consumers/internet/internet_and_broadband_services_-_policy,_regulation__and__programs/reviews_and_inquiries.

[12] Online Council, "Twelfth Ministerial meeting of the Online Council," Aug. 2005. http://www.mrdb.nsw.gov.au/telecom/8wn5bxz5.htm.

[13] Online Council, "Seventh Ministerial meeting of the Online Council," 2000. http://www.dcita.gov.au/Article/0,,0_4-2_4008-4_15092,00.html.

[14] Human Rights and Equal Opportunities Commission, "World Wide Web Access: Disability Discrimination Act Advisory Notes," 2000. http://www.hreoc.gov.au/disability_rights/standards/www_3/www_3.html.

[15] The European Commission, "Communication from the Commission to the Council, the European Parliament [...]: eAccessibility," Sept. 2005. http://ec.europa.eu/information_society/activities/einclusion/docs/access/cec_com_eacc_2005.html.

[16] W3C / Web Accessibility Initiative, "Web Content Accessibility Guidelines 1.0," tech. rep., World Wide Web Consortium (W3C), May 1999. http://www.w3.org/TR/WAI-WEBCONTENT/.

[17] W3C / Web Accessibility Initiative, "WCAG 2 FAQ," Dec. 2007. http://www.w3.org/WAI/WCAG20/wcag2faq.html.

[18] M. G.-S. B. und Wohnberatung für Behinderte, "Barrierefreies Bauen für alle (alte, nichtbehinderte) Menschen," 2004.

[19] J. S. Britsios, "Why Accessibility is important to you?," 2005. http://www.webnauts.net/accessibility.html.

[20] C. Letourneau, "Accessible Web Design," 2003. http://www.starlingweb.com/webac.htm.

[21] M. F. Theofanos and J. Redish, "Guideline for Accessible and Usable Web Sites: Observing Users Who Work With Screen Readers," *interactions*, vol. X, pp. 36–51, 2003. http://www.redish.net/content/papers/interactions.html.

[22] WebAIM, "Assistive Technologies for Motor Disabilities," 2005. http://www.webaim.org/techniques/motor/assistive.php.

[23] U.S. Census Bureau, "Census Bureau Home Page," Mar. 2005. http://www.census.gov/.

[24] U.S. Census Bureau, "Census 2000 Brief- Disability Status 2000," Mar. 2003. http://www.census.gov/prod/2003pubs/c2kbr-17.pdf.

[25] Premier's Council on the Status of Disabled Persons, "Statistics on Persons with Disabilities," Dec. 2007. http://www.gnb.ca/0048/PCSDP/Statistics2006-e.asp.

[26] Eurostat, "EUROPA - Eurostat - Home Page," 2008. http://ec.europa.eu/eurostat/.

[27] Directorate General V, "Mainstreaming Disability within EU Employment and Social Policy," Apr. 2000. http://europa.eu.int/comm/employment_social/soc-prot/disable/dresden/workpaper_en.pdf.

[28] Eurostat, "Employment of disabled people in Europe in 2002," Nov. 2003. http://ec.europa.eu/employment_social/health_safety/docs/disabled_%202002_en.pdf.

[29] The European Commission, "Equal Opportunities for People with Disabilities: a European action plan," June 2004. http://europa.eu.int/scadplus/leg/en/cha/c11414.htm.

[30] Australian Bureau of Statistics (ABS), "Australian Bureau of Statistics Home Page," 2008. http://www.abs.gov.au/.

[31] Australian Bureau of Statistics (ABS), "4430.0 Disability, Ageing and Carers, Australia: Summary of Findings," Sept. 2004. http://www.abs.gov.au/Ausstats/abs@.nsf/0e5fa1cc95cd093c4a2568110007852b/c258c88a7aa5a87eca2568a9001393e8.

[32] W3C / Web Accessibility Initiative, "Fact Sheet: WCAG 1.0," 2005. http://www.w3.org/1999/05/WCAG-REC-fact.html.

[33] W3C, "How People with Disabilities Use the Web," tech. rep., World Wide Web Consortium (W3C), July 2004. http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/Overview.html.

[34] M. Williams, "Why is Web Accessibility important to you?," 2005. http://www.revs.org/article.html.

[35] H. Dilling, W. Mombour, and M. H. Schmidt, *Internationale Klassifikation psychischer Störungen*. Hans Huber, 2000.

[36] One Hundred Fifth Congress of the United States, "Assistive Technology Act of 1998," 1998. http://www.section508.gov/docs/AT1998.html.

[37] W3C, "W3C Launches Web Accessibility Initiative," 1997. http://www.w3.org/Press/WAI-Launch.html.

[38] Charles F. Goldfarb, *The SGML Handbook*. Clarendon Press, 1994.

[39] W3C / HTML Working Group, "HTML 4.01 specification," tech. rep., World Wide Web Consortium (W3C), Dec. 1999. http://www.w3.org/TR/html401/.

[40] W3C / HTML Working Group, "XHTML 1.0: The Extensible Hypertext Markup Language 1.0 (Second Edition)," tech. rep., World Wide Web Consortium (W3C), Aug. 2000. http://www.w3.org/TR/xhtml1/.

[41] W3C, "Extensible Markup Language (XML) 1.1," tech. rep., World Wide Web Consortium (W3C), Apr. 2004. http://www.w3.org/TR/xml11/.

[42] Sun Microsystems, Inc., "Java Technology," 2005. http://java.sun.com/.

[43] Macromedia, Inc., "Macromedia," 2005. http://www.macromedia.com/.

[44] CEN/TC 304, "8 bit character sets - Historical background," 1998. http://ra.dkuug.dk/CEN/TC304/guide/GHIST.HTM.

[45] ISO/IEC, "ISO/IEC 646:1991: Information technology – ISO 7-bit coded character set for information interchange," tech. rep., ISO/IEC, 1991. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=4777.

[46] Jukka Korpela, "A tutorial on character code issues," 2007. http://www.cs.tut.fi/~jkorpela/chars.html.

[47] SIL International, "Character Set Encoding Basics," 2003. http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=IWS-Chapter03.

[48] W3C / Web Accessibility Initiative, "Techniques for Authoring Tool Accessibility Guidelines 1.0," tech. rep., World Wide Web Consortium (W3C), Oct. 2002. http://www.w3.org/TR/ATAG10-TECHS/.

[49] W3C / Web Accessibility Initiative, "HTML Techniques for Web Content Accessibility Guidelines 1.0," tech. rep., World Wide Web Consortium (W3C), Nov. 2000. http://www.w3.org/TR/WCAG10-HTML-TECHS/.

[50] W3C / Web Accessibility Initiative, "Legal and Policy Factors in Developing a Web Accessibility Business Case for Your Organization," Aug. 2005. http://www.w3.org/WAI/bcase/pol.

[51] W3C / Web Accessibility Initiative, "Policies Relating to Web Accessibility," 2005. http://www.w3.org/WAI/Policy/.

[52] Section 508, "Section 508 home," 2005. http://www.section508.gov/.

[53] Section 508, "Section 508 Acquisition FAQ's," 2005. http://www.section508.gov/index.cfm?FuseAction=Content&ID=75.

[54] European Commission / Your Europe, "Public Procurement (European Union)," Dec. 2006. http://ec.europa.eu/youreurope/nav/en/business/public-procurement/info/index.html.

[55] European-American Business Council and Information Technology Industry Council, "Letter regarding Draft Communication from the Commission on eAccessibility," Oct. 2004. http://itic.org/archives/ITI-Blixt%20Letter.pdf.

[56] The Access Board, "The Telecommunications Act of 1996, Section 255," 2001. http://www.access-board.gov/about/laws/telecomm.htm.

[57] The Access Board, "Telecommunications Act Accessibility Guidelines," Feb. 1998. http://www.access-board.gov/telecomm/rule.htm.

[58] The Access Board, "Frequently Asked Questions about Section 255 of the Telecommunications Act of 1996," 2001. http://www.access-board.gov/telecomm/FAQ.htm.

[59] One Hundred First Congress of the United States of America, "The Americans with Disabilities Act of 1990," 1990. http://www.usdoj.gov/crt/ada/pubs/ada.htm.

[60] U.S. Department of Justice, "United States Department of Justice Policy Ruling, 9/9/96," 1996. http://www.usdoj.gov/crt/foia/tal712.txt.

[61] United States Court of Appeals for the First Circuit, "Carparts Distribution Center v. Automotive Wholesaler's Association of New England," 2004. http://harp.org/carparts.txt.

[62] Aaron P. Silberman, "Disabled Access and the Internet: Legal Issues," Aug. 2002. http://rjo.com/PDF/DisabledAccessAndInternet.pdf.

[63] S. Mendelsohn and M. Gould, "When the Americans with Disabilities Act goes Online: Application of the ADA to the Internet and the World Wide Web," 2004. http://www.smu.edu/csr/articles/2004/Winter/Mendelsohn.pdf.

[64] U.S. Department of Justice, "Hooks v. OKBridge, Brief of the US as amicus curiae in favour of appellant," 2004. http://www.usdoj.gov/crt/briefs/hooks.htm.

[65] United States Court of Appeals for the Eleventh Circuit, "Access Now, Inc. v. Southwest Airlines Company," 2004. http://www.altlaw.org/v1/cases/170176.pdf.

[66] theregister.com, "California Court tilts towards mandating web accessibility," 2007. http://www.theregister.co.uk/2007/10/14/california_target_web_accessibility/.

[67] Marylin Hall Patel, “Memorandum and Order: Re: Defandant’s Motion to Dismiss; Plaintiff’s Motion for Preliminary Injunction,” Sept. 2006. http://www.dralegal.org/downloads/cases/target/062_order_deny_PI_grant_part_MTD.txt.

[68] Marylin Hall Patel, “Memorandum and Order: Re: Plaintiffs’ Motion for Class; Plaintiffs, Certification and Motion to Bifurcate; Defendant’s Motion for Summary Judgment,” Sept. 2006. http://www.dralegal.org/downloads/cases/target/target-order.txt.

[69] Information Society / i2010 Editor, “Before i2010: eEurope Initiative,” 2008. http://ec.europa.eu/information_society/eeurope/i2010/archive/eeurope/index_en.htm.

[70] Information Society / eEurope Editor, “European Commission - Information Society - eEurope 2005 - Mission statement,” 2005. http://europa.eu.int/information_society/eeurope/2005/all_about/mission_statement/index_en.htm.

[71] The Council and the European Commission, “eEurope 2002 Action Plan,” 2005. http://europa.eu.int/information_society/eeurope/2002/action_plan/pdf/actionplan_en.pdf.

[72] The European Commission, “Communication from the Commission to the Council, the European Parliament [...] eEurope 2002: Accessibility of Public Web Sites and their Content,” Sept. 2001. http://europa.eu/eur-lex/en/com/cnc/2001/com2001_0529en01.pdf.

[73] The European Parliament, “European Parliament resolution on the Commission communication eEurope 2002: Accessibility of Public Web Sites and their Content,” 2002. http://ec.europa.eu/information_society/activities/einclusion/policy/accessibility/tech_services/com_wa2001/a_documents/ep_res_web_wai_2002.html.

[74] Information Society / e-Inclusion, “Focus: eAccessibility,” 2008. http://ec.europa.eu/information_society/activities/einclusion/policy/accessibility/index_en.htm.

[75] eAccessibility Experts Group, “eAccessibility Experts Group,” 2008. http://ec.europa.eu/information_society/activities/einclusion/policy/accessibility/expert_gp/index_en.htm.

[76] eAccessibility Initiative, “WAI contents guidelines for public Web sites in the EU - Final Progress Report,” Dec. 2002. http://ec.europa.eu/employment_social/knowledge_society/docs/eacc_wai.pdf.

[77] The European Council, "Council Resolution on "eAccessibility" - improving the access of people with disabilities to the Knowledge Based Society," Dec. 2002. http://ec.europa.eu/employment_social/knowledge_society/docs/res_eacc_en.pdf.

[78] Directorate General Information Society, "Report: Public on-line consultation on a forthcoming Commission Communication on eAccessibility," 2005. http://ec.europa.eu/information_society/activities/einclusion/docs/access/comm_ea_2005/com_consult_res.pdf.

[79] empirica and Work Research Centre, "Measuring Progress of eAccessibility in Europe, Executive Summary," Oct. 2007. http://ec.europa.eu/information_society/activities/einclusion/docs/meac_study/meac_report_exec_sum_05_11.pdf.

[80] Information Society / i2010 Editor, "i2010 - A European Information Society for growth and employment," 2008. http://ec.europa.eu/information_society/eeurope/i2010/index_en.htm.

[81] UK Parliament, "Disability Discrimination Act 1995," 1995. http://www.legislation.hmso.gov.uk/acts/acts1995/1995050.htm.

[82] UK Parliament, "Disability Rights Commission Act 1999," 1999. http://www.hmso.gov.uk/acts/acts1999/19990017.htm.

[83] UK Parliament, "Disability Discrimination Act 2005," 2005. http://www.england-legislation.hmso.gov.uk/acts/acts2005/ukpga_20050013_en_1.

[84] UK Parliament, "Equality Act 2006," 2006. http://www.opsi.gov.uk/acts/acts2006/ukpga_20060003_en_1.

[85] Equality and Human Rights Commission, "Equality and Human Rights Commission - Homepage," 2007. http://www.equalityhumanrights.com/.

[86] Disability Rights Commission, "Disability Discrimination Act Code of Practice," 2006. http://www.equalityhumanrights.com/Documents/Disability/Services/DRC%20Access%20code%20of%20practice.pdf.

[87] Disability Rights Commission, "The Web: Access and Inclusion for Disabled People," Apr. 2004. http://www.equalityhumanrights.com/Documents/DRC/Useful%20Documents/The%20Web_Access%20and%20inclusion%20for%20disabled%20people.pdf.

[88] W3C / Web Accessibility Initiative, "W3C Web Accessibility Initiative Statement on Recent Report," Apr. 2004. http://www.w3.org/2004/04/wai-drc-statement.html.

[89] M. Sloan, “Web Accessibility and the DDA,” *Journal of Information, Law and Technology (JILT)*, July 2002. http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2001_2/sloan/.

[90] Der Deutsche Bundestag, “Gesetz zur Gleichstellung behinderter Menschen und zur Änderung anderer Gesetze,” 2002. http://www.gesetze-im-internet.de/bgg/index.html.

[91] Bundesministerium des Innern und Bundesministerium für Arbeit und Sozialordnung, “Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz,” 2002. http://www.einfach-fuer-alle.de/artikel/bitv/.

[92] Bundesministerium des Innern und Bundesministerium für Arbeit und Sozialordnung, “Begründung zur ’Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz’,” 2002. http://www.einfach-fuer-alle.de/artikel/bitv/begruendung/.

[93] U. Pidun, “Barrierefreies Web-Design,” *c’t: Magazin für Computer Technik*, pp. 186–??, may 2004. http://www.heise.de/ct/04/18/186/.

[94] H. Rauschhofer, “Gleichstellung im Internet - Gesetzlicher Auftrag und Konsequenzen der Ignoranz,” 2003. http://www.rechtsanwalt.de/barrierefrei/Barrierefreies-Internet_14Okt03.PDF.

[95] Stabsstelle der IKT-Strategie des Bundes- Bundeskanzleramt Republik Österreich, “Bericht über die Umsetzung der WAI-Richtlinien in Österreich,” 2004. http://www.cio.gv.at/egovernment/wai/2004_04_30_WAI-Umsetzungsbericht.pdf.

[96] Der Österreichische Nationalrat, “E-Government Gesetz - E-GovG,” 2004. https://labs.cio.gv.at/delivery/egov.pdf.

[97] The Austrian National Assembly, “The Austrian E-Government Act,” 2004. http://www.cio.gv.at/egovernment/law/E-Gov_Act_endg_engl_Fassung1.pdf.

[98] Der Österreichische Nationalrat, “Bundes-Gleichstellungsgesetz - BGStG,” 2005. http://www.parlament.gv.at/pls/portal/docs/page/PG/DE/XXII/I/I_00836/FNAMEORIG_036804.HTML.

[99] Der Österreichische Nationalrat, “Materialien zum Bundes-Gleichstellungsgesetz - BGStG,” 2005. http://www.parlament.gv.at/pls/portal/docs/page/PG/DE/XXII/I/I_00836/FNAMEORIG_036806.HTML.

[100] Stabsstelle der IKT-Strategie des Bundes- Bundeskanzleramt Republik Österreich, “Erhebung Barrierefreiheit 2007,” 2007. http://www.digitales.oesterreich.gv.at/DocView.axd?CobId=24558.

[101] W3C / Web Accessibility Initiative, "Financial Factors in Developing a Web Accessibility Business Case for Your Organization," Aug. 2005. http://www.w3.org/WAI/bcase/fin.

[102] W3C / Web Accessibility Initiative, "Developing a Web Accessibility Business Case for Your Organization," Aug. 2005. http://www.w3.org/WAI/bcase/Overview.

[103] W3C / Web Accessibility Initiative, "Technical Factors in Developing a Web Accessibility Business Case for Your Organization," Aug. 2005. http://www.w3.org/WAI/bcase/tech.

[104] W3C / Web Accessibility Initiative, "Draft: Selecting Web Accessibility Evaluation Tools," 2005. http://www.w3.org/WAI/EO/Drafts/eval/selectingtools.html.

[105] J. Thatcher, "Evaluation of Bobby," 2004. http://www.jimthatcher.com/bobbyeval.htm.

[106] Watchfire Corporation, "Watchfire Launches Free Online Testing Service," May 2003. http://www.watchfire.com/news/releases/5-12-03.aspx.

[107] HiSoftware and ICDRI, "About the HiSoftware Cynthia Says Portal," 2004. http://www.cynthiasays.com/cynthia/cynthia.htm.

[108] HiSoftware and ICDRI, "The HiSoftware Cynthia Says Portal," 2004. http://www.cynthiasays.com/.

[109] A. Bernardini, "Torquemada- web for all," 2008. http://www.webxtutti.it/testa.htm.

[110] WebAIM, "WAVE 3.0 - Web Accessibility Versatile Evaluator," 2005. http://www.wave.webaim.org/.

[111] Watchfire Corporation, "Watchfire provides enterprise software and services to manage online privacy, security, quality and compliance risks," 2005. http://www.watchfire.com/.

[112] HiSoftware, "HiSoftware: Leader in Web Content Quality and Compliance Solutions for the Enterprise," 2005. http://www.hisoftware.com/.

[113] Watchfire Corporation, "Watchfire WebXACT," 2008. http://webxact.watchfire.com/.

[114] Watchfire Corporation, "WebXACT Terms of Use," 2004. http://webxact.watchfire.com/themes/standard-en-us/termsofuse.htm.

[115] HiSoftware and ICDRI, "Terms of USE," 2007. http://www.cynthiasays.com/org/termsofuse.htm.

[116] W3C / Web Accessibility Initiative, "CSS Techniques for Web Content Accessibility Guidelines 1.0," tech. rep., World Wide Web Consortium (W3C), Nov. 2000. http://www.w3.org/TR/WCAG10-CSS-TECHS/.

[117] W3C / Web Accessibility Initiative, "Techniques for Accessibility Evaluation and Repair Tools," tech. rep., World Wide Web Consortium (W3C), Apr. 2000. http://www.w3.org/TR/AERT.

[118] W3C, "Cascading Style Sheets, level 2, CSS2 specification," tech. rep., World Wide Web Consortium (W3C), May 1998. http://www.w3.org/TR/1998/REC-CSS2/.

[119] C. Letourneau and G. Fred, "Curriculum for Web Content Accessibility Guidelines 1.0," Mar. 2000. http://www.w3.org/WAI/wcag-curric/.

[120] W3C, "W3C Intellectual Property FAQ," Sept. 2007. http://www.w3.org/Consortium/Legal/IPR-FAQ-20000620.

[121] Wab Cluster, "Unified web evaluation methodology 1.0," tech. rep., Wab Cluster, July 2006. http://www.wabcluster.org/uwem1/UWEM_1_0.pdf.

[122] The WAB Cluster, "The EU Web Accessibility Benchmarking Cluster, Evaluation and benchmarking of Accessibility," 2008. http://www.wabcluster.org/.

[123] The European Internet Accessibility Observatory, "European Internet Accessibility Observatory," 2008. http://www.eiao.net/.

[124] Support-EAM, "Support-EAM: project home page," 2008. http://www.support-eam.org/.

[125] BenToWeb, "BenToWeb :: Project description," 2008. http://www.bentoweb.org/desc.

[126] The Perl Foundation, "About Perl," 2008. http://www.perl.org/about.html.

[127] The Drupal Foundation, "About Drupal," 2008. http://www.drupal.org/about.

[128] The Apache Foundation, "About the Apache HTTP Server Project," 2007. http://httpd.apache.org/ABOUT_APACHE.html.

[129] MySQL AB, "MySQL AB :: About MySQL," 2007. http://www.mysql.com/company/.

[130] The PHP Group, "PHP: Hypertext Preprocessor," 2007. http://www.php.net/.

[131] D. Grune and B. Berliner, “CVS: Open Source Version Control,” 2006. http://www.nongnu.org/cvs/.

[132] CollabNet, “Subversion,” 2008. http://subversion.tigris.org/.

[133] Edgewall Software, “The Trac Project,” 2007. http://trac.edgewall.com/.

[134] The Eclipse Foundation, “About the Eclipse Foundation,” 2007. http://www.eclipse.org/org/.

[135] Drupal API system, “Hooks — Drupal API,” 2008. http://api.drupal.org/api/group/hooks/.

[136] Nigel McNie, “GeSHi - Generic Syntax Highlighter,” 2007. http://qbnz.com/highlighter/.

[137] John Albin, “Zen,” 2008. http://drupal.org/project/zen.

[138] W3C / HTML Working Group, “XHTML Media Types,” tech. rep., World Wide Web Consortium (W3C), Aug. 2002. http://www.w3.org/TR/xhtml-media-types/.

[139] Bjoern Hoehrmann, “HTML::Encoding - Determine the encoding of HTML/XML/XHTML documents.,” Dec. 2007. http://search.cpan.org/dist/HTML-Encoding/lib/HTML/Encoding.pm.

[140] The PHP Group, “PHP: Strings,” 2008. http://php.net/manual/en/language.types.string.php.

[141] W3C / Web Accessibility Initiative, “User Agent Support for Accessibility,” 2003. http://www.w3.org/WAI/Resources/WAI-UA-Support.

[142] W3C / Web Accessibility Initiative, “Web Content Accessibility Guidelines 2.0 Working Draft 27 April 2006,” Apr. 2006. http://www.w3.org/TR/2006/WD-WCAG20-20060427/.

[143] J. Thatcher, “Testing Section 508 Web Accessibility,” Apr. 2004. http://jimthatcher.com/testing.htm.

[144] Y. Jo, “ASCII Art,” Nov. 2000. http://www.w3.org/WAI/ER/IG/ert/AsciiArt.htm.

[145] H. Alvestrand, “Tags for the identification of languages, RFC 1766,” Mar. 1995.

[146] W3C / HTML Working Group, “HTML 4 Errata,” Mar. 2005. http://www.w3.org/MarkUp/html4-updates/errata.

[147] WATS.ca, “Using Accesskeys - Is it worth it?,” 2002. `http://www.wats.ca/show.php?contentid=32`.

[148] D. Dardailler, “Tablin: an HTML Table linearizer,” 2000. `http://www.w3.org/WAI/References/Tablin/`.

A. Techniques for Evaluation of Conformance to WCAG 1.0

This appendix contains a list of techniques to evaluate Web pages for conformance with the Web Content Accessibility Guidelines 1.0 (WCAG 1.0). Please see Chapter 5 for a description of the evaluation methodology this appendix is a part of.

Please note that although some WCAG 1.0 checkpoints can be addressed with software algorithms, there are checkpoints which cannot be tested automatically. They require human judgment to determine conformance to the accessibility guidelines and must be checked manually.

A.1. Structure of this Chapter

Each WCAG 1.0 guideline is described in a separate section of this chapter. The guideline titles, as well as the descriptions of the guidelines and checkpoints are *direct excerpts* from the Web Content Accessibility Guidelines 1.0. They are set in *italic* and in a different background color to clearly distinguish them from the rest of the text. The guideline titles, although direct excerpts, are not italicized and provide no reference to the WCAG 1.0 document because this would interfere with the readability of this document.

For reasons of convenience, the structure of this chapter is closely based on the WAI working draft on "Techniques for Accessibility Evaluation and Repair Tools" (hereafter referred to as AERT). However, its contents are not a mere copy of that document. Some of the techniques have been modified or split, some of them have been left out; some of the techniques in our document are drawn from other sources, while others still are our own. Most of the techniques in the W3C working draft can be found in WCAG 1.0 technical documents. If a technique is not our own, the source is given in the "Source" section of a technique. The technique's titles and descriptions are close paraphrases, but not direct excerpts, of those resources.

A checkpoint subsection may include any number of evaluation techniques. Each evaluation technique includes the following information:

a) Technique ID and title: The technique identifier and a short title describing the technique.

b) Category: Each technique falls into one of the following categories:

Automatic: An automatic check can be performed without user interaction. Non-conformance to this kind of check is to be reported as an error in the accessibility report.

An example for this kind of check is the verification of the absence of the `blink` element.

Semi-Automatic: A semi-automatic check is triggered by certain elements of a Web page. Some accessibility barriers can be detected automatically; in this case, an error is reported. However, if no error can be detected, the user has to manually check the Web page for conformance to the technique. This may require an analysis of the Web page as it is displayed by a user agent, or an examination of the page's HTML and/or CSS code. Web Accessibility evaluation and repair tools should assist the user in these tasks.

Semi-automatic checks therefore consist of two parts: an automatic part, which can detect possible accessibility barriers, and a manual part to confirm or deny the presence of an accessibility barrier, which has to be carried out be a human evaluator.

An example for this kind of check is the verification that the value of the `alt` attribute of `img` elements must be an appropriate text equivalent of the image. All `img` elements that have no `alt` attribute fail the automatic part of this check. A human evaluator must determine whether elements that pass the automatic part satisfy the manual part of the check, e.g. whether the value of the `alt` attribute is an appropriate description or textual representation of the `img` element.

Manual: A manual check always requires user interaction. There are two types of manual checks: Some are concerned with the overall design of a site and are thus always present in an accessibility report ("full manual check"), others are triggered by specific site content ("manual check"). Like the manual part of a semi-automatic check, a manual check requires an analysis of the Web page as displayed by a user agent, or an examination of the page's HTML and/or CSS code. Web Accessibility evaluation and repair tools should assist the user in these tasks.

An example for this kind of check is the verification that changes in the natural language of a document are clearly identified in the document's HTML code.

Warning: Like automatic checks, warnings are checks performed without user interaction. Unlike automatic checks, warnings are triggered by suspicious content, which may or may not be a problem in terms of Web Accessibility. Human interaction is required to determine whether a warning actually constitutes an error in terms of accessibility barriers. Warnings are usually heuristic algorithms which may yield false positives.

An example for this kind of check is the verification that the value of the `alt` attribute of `img` elements does not end in a filename suffix associated with an image file. Filenames are not an appropriate text equivalent for an image, however it is possible (though very unlikely) that such a character sequence occurs naturally at the end of a text equivalent for an image.

A note on categorization: All the tools that we have evaluated mark some items which are subject to some amount of human judgment as "automatic", whereas our implementation marks such items as "semi-automatic". Consider that our example for a semi-automatic check is categorized as "automatic" by all the implementations that we have seen, failing to take into account that text equivalents have to be verified by human evaluators.

We believe that the common approach disregards the WCAG requirement for appropriate equivalent descriptions, textual or otherwise, and gives users who are unfamiliar with Web Accessibility an incorrect understanding of the requirements of human intervention when evaluating Web pages for accessibility conformance.

Moreover, a number of tools use the term "Warning" for manual checks, the rationale being that the check may or may not apply. We chose different terminology because we believe "manual" is a less ambiguous term.

c) Automatic check: Techniques in the categories automatic, semi-automatic and warning include a description of algorithmic tests that may be applied to Web page content.

d) Manual check: This subsection applies only to semi-automatic and manual techniques. It describes the test procedures human evaluators may follow to determine whether page content passes this check.

e) Source: References the source of the check; see next section.

f) Modification(s): This subsection describes any modifications that have been made to a referenced check.

g) Note(s): Remarks on this check (optional).

A.2. References

We have used a numerical bibliography-style for references in the main part of our thesis because most of our references are to Web pages or technical reports which have an Institution as the "author", which makes alphanumeric references difficult to interpret. Since we use only a small number of Web pages and technical reports throughout the appendix, we chose to use the following alphanumeric references for documents that are referenced frequently:

WCAG10: W. Chisholm, G. Vanderheiden, I. Jacobs (Eds), "The Web Content Accessibility Guidelines 1.0", 1999, W3C Recommendation 5-May-1999. `http://www.w3.org/TR/WCAG10/`

WCAG10-HTML-TECHS: W. Chisholm, G. Vanderheiden, I. Jacobs (Eds), "HTML Techniques for Web Content Accessibility Guidelines 1.0", 1999, W3C Note 6 November 2000. `http://www.w3.org/TR/WCAG10-HTML-TECHS/`

WCAG10-CSS-TECHS: W. Chisholm, G. Vanderheiden, I. Jacobs (Eds), "CSS Techniques for Web Content Accessibility Guidelines 1.0", 1999, W3C Note 6 November 2000. `http://www.w3.org/TR/WCAG10-CSS-TECHS/`

AERT: C. Ridpath, W. Chisholm (Eds), "Techniques For Accessibility Evaluation And Repair Tools", 2000, W3C Working Draft 26-April-2000. `http://www.w3.org/TR/AERT`

HTML4: D. Raggett, A. Le Hors, I. Jacobs (Eds), "HTML 4.01 Specification", 1999, W3C Recommendation 24-December-1999. `http://www.w3.org/TR/html401/`

CSS2: B. Bos, H. Wium Lie, C. Lilley, I. Jacobs (Eds), "Cascading Style Sheets Level 2", 1998, W3C Recommendation 12-May-1998. `http://www.w3.org/TR/REC-CSS2/`

RFC1766: H. Alvestrand, "Tags for the Identification of Languages", 1995, Request for Comments: 1766, IETF. `http://www.ietf.org/rfc/rfc1766.txt`

JS-TESTING: J. Thatcher, M. Burks, C. Heilmann, A. Kirkpatrick et al, "Web Accessibility: Web Standards and Regulatory Compliance", 2006, Ch. 13. `http://www.jimthatcher.com/testing.htm`

A.3. Methodology Conformance

Any number of techniques can be associated with a checkpoint. A technique may apply to a document any number of times in different ways: some content may pass, while other content fails.

An automatic technique shall be satisfied if there are no instances of failures. A semi-automatic technique shall be satisfied if the whole document passed the automatic part of the check, and the manual part was verified by a human evaluator. A manual technique shall be satisfied if it was verified by a human evaluator.

A WCAG 1.0 checkpoint shall be satisfied if all techniques relevant to the checkpoint are satisfied; conversely, a checkpoint shall fail if any one technique associated with the checkpoint fails.

An accessibility report shall be a duly completed version of the Report Template provided by WAI at `http://www.w3.org/WAI/eval/template.html` including the following additional information:

- The Executive Summary shall indicate that this Methodology was used to conduct the evaluation;

- An Appendix shall indicate which techniques the document failed conformance to.

Tools may generate "partial" accessibility reports that only determine results that can be evaluated by software algorithms alone. These reports shall clearly indicate that manual checks by a human evaluator are necessary to determine whether manual checks, including the "manual" part of semi-automatic checks, apply to the document.

A "Full" Accessibility Report shall include all techniques failing the conformance checks of this methodology, where all "manual" checks have been verified by a human evaluator, by himself or with the assistance of an evaluation tool implementing this methodology.

WCAG 1.0 compliance levels and priorities remain unchanged.

If a document contains frames or inline frames, it shall be deemed in conformance to WCAG Level A, AA or AAA only if all documents referenced by the `src` attribute of those elements are valid, *accessible* (X)HTML *of the same conformance level or higher.*

A document shall not be accessible if any externally referenced content cannot be retrieved. Missing external content may prevent the discovery of accessibility barriers related to the document.

A.4. The application of WCAG 1.0 to (X)HTML and CSS

WCAG 1.0 is a document meant to be stable across evolving technologies. There is room for interpretation and judgment on how to apply its general concepts to (X)HTML and CSS.

WCAG 1.0 consists of 14 guidelines and 65 checkpoints. 13 of the latter include the phrase "*Until user agents...*". They refer to accessibility needs that should be met by user agents; however, not all user agents or assistive technologies provide the required features. Checkpoints containing this phrase require additional accessibility support until most user agents include the necessary accessibility features.

The "*Until user agents...*" entry in the WCAG glossary refers to a Web site that is supposed to provide current information about user agent support for those accessibility features. The site was last updated on December 2003 and does not provide information for 6 of the 13 checkpoints [141]. At the time of writing there seems to be no official WAI resource that states if any of these checkpoints have been deprecated. Appendix D of the WCAG 2.0 Last Call Working Draft released on 27 April 2006 designates checkpoints 1.5, 7.3, 10.2, 10.3, 10.4 and 10.5 as deprecated, however WCAG 2.0 is still a draft and subject to change [142]. Implementations of this Evaluation Methodology may choose not to implement Techniques that are omitted in future versions of WCAG.

Guideline 5 is concerned with the correct use of tables and clearly states that their use for layout purposes should be avoided. Checkpoint 5.3 notes that once user agents support style sheet positioning, tables should not be used for layout. CSS is now supported by the most common browsers, and although not all of them implement the full specification, CSS positioning is supported to an extent that should eliminate the need to use tables for layout in HTML. It is thus our belief that tables need no longer be used for layout.

A.5. Guideline 1. Provide equivalent alternatives to auditory and visual content.

"Provide content that, when presented to the user, conveys essentially the same function or purpose as auditory or visual content." [WCAG10]

Checkpoint 1.1

"Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). <u>This includes:</u> images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video. [Priority 1]" [WCAG10]

Technique 1.1.1 All `img` elements must have an `alt` attribute.

Category: Semi-automatic.

Automatic check: FAILS for `img` elements without an `alt` attribute. SUCCEEDS for `img` elements with an `alt` attribute; these must pass the MANUAL CHECK.

Manual check: Make sure the ALT text is an appropriate text equivalent for the image. If the ALT text is too long, the `longdesc` attribute must be used to provide more extensive information.

Source: [WCAG10].

Note: The value of the `alt` attribute is often referred to as "ALT text".

Technique 1.1.2 The value of the `alt` attribute of `img` elements not embedded in links must not be "empty" or consist only of white space.

Category: Warning.

Automatic check: Targets `img` elements *not* embedded in `a` elements. FAILS if the ALT text is "empty" (NULL ALT text) or consists only of white space (blank ALT text). See also Section 2.6.10 on white space.

Source: [AERT, Technique 1.1.1].

Note: There seems to be no consensus when blank or NULL ALT text should be used, or if either should be allowed at all. For instance, [WCAG10-HTML-TECHS, section 6.1.1] suggests a single space for the ALT text of an image if link text is provided. However, [AERT, Technique 1.1.1] requires that ALT text never be blank, and a value of one or more spaces be allowed only if the image is not within an `a` element. We can only assume the latter to be a mistake in [AERT].

Modifications: [AERT, Technique 1.1.1] allows one or more spaces if the image *is not* embedded within an `a` element. This has been changed to *allow white space* if the image *is* embedded within an `a` element.

Note: Blank or NULL ALT text may make sense for graphical bullets [JT-TESTING] and transparent images used for layout (also called "spacers").

Technique 1.1.3 Check the value of the `alt` attribute of `img` elements for suspicious content.

Category: Warning.

Automatic check: FAILS if the value of the `alt` attribute of `img` elements:

- ends in "bytes" [AERT, Technique 1.1.1];
- ends in a filename suffix associated with an image file [AERT, Technique 1.1.1];
- ends in any of the words "image", "photo", "images" (as in "turn on images") [AERT, Technique 1.1.1];
- includes any of the words "spacer", "null", "go", "click" [JT-TESTING];
- is longer than 150 characters. In this case, suggest that a description file be created (via the `longdesc` attribute) [AERT, Technique 1.1.1].

Source: See list.

Technique 1.1.4 All `input` elements of `type="image"` must have a valid `alt` attribute.

Category: Semi-automatic.

Automatic check: FAILS if there is no `alt` attribute or if the ALT text is "empty" (NULL ALT text) or if it consists only of white space (blank ALT text). See also Section 2.6.10 on white space. SUCCEEDS for all other `input` elements of `type="image"`; these must pass the MANUAL CHECK.

Manual check: Make sure that the ALT text is an appropriate text equivalent for the graphical submit button.

Source: [AERT, Technique 1.1.3].

Modifications: [AERT, Technique 1.1.3] requires that the value of the `alt` attribute must not consist of *spaces*. This has been changed to *white space*.

Technique 1.1.5 Check the value of the `alt` attribute of `input` elements of `type="image"` for suspicious content.

Category: Warning.

Automatic check: FAILS if the value of the `alt` attribute of `input` elements of `="image"`:

- ends in "bytes" [AERT, Technique 1.1.3];
- ends in a filename suffix associated with an image file [AERT, Technique 1.1.3];
- ends in any of the words "image", "photo", "images" (as in "turn on images") [AERT, Technique 1.1.3];
- includes any of the words "spacer", "null", "go", "click" [JT-TESTING].

Source: See list.

Technique 1.1.6 All `applet` elements must have a valid `alt` attribute *and* element content.

Category: Semi-automatic.

Automatic check: FAILS unless a text equivalent is provided in both the `alt` attribute *and* textual element content and the value of the `alt` attribute is not empty or consists only of white space. SUCCEEDS for all other `applet` elements; these must pass the MANUAL CHECK.

Manual check: Ensure that the value of the `alt` attribute and the textual content of the `applet` element are appropriate text equivalents of the applet.

Source: [WCAG10-HTML-TECHS].

Note: Requiring a text equivalent in both the `alt` attribute and element content enables the content to transform gracefully for user agents that support only one of the two mechanisms.

Modifications: [WCAG10-HTML-TECHS] contains no provisions about NULL or BLANK ALT text.

Technique 1.1.7 Check the value of the `alt` attribute of `applet` elements for suspicious content.

Category: Warning.

Automatic check: FAILS if the value of the `alt` attribute of `applet` elements:

- ends in “bytes” [AERT, Technique 1.1.4];
- ends in a filename suffix associated with an image file [AERT, Technique 1.1.4];
- ends in any of the words “image”, “photo”, “images” (as in “turn on images”) [AERT, Technique 1.1.4];
- includes any of the words “spacer”, “null”, “go”, “click” [JT-TESTING].

Source: See list.

Technique 1.1.8 All `object` elements must have textual element content.

Category: Semi-automatic.

Automatic check: FAILS if the `object` element contains no textual element content, or only white space. SUCCEEDS for all other `object` elements; these must pass the MANUAL CHECK.

Manual check: Make sure that the textual content of the `object` element is an appropriate text equivalent of the object.

Source: [WCAG10].

Modifications: [WCAG10] does not require the content to be textual. We believe that requiring textual content for all `object` elements allows Web content to transform graceful, regardless of disability, environmental or technological constraints. [WCAG10] also contains no provisions about NULL or BLANK ALT text.

Note: The contents of the object element will be rendered if the user agent cannot render the object itself.

Technique 1.1.9 All `frameset` elements must have a `noframes` element in their element content.

Category: Semi-automatic.

Automatic check: FAILS for `frameset` elements that do not have a `noframes` element as element content. SUCCEEDS for all other `frameset` elements; these must pass the MANUAL CHECK.

Manual check: Verify that the textual content of the `noframes` element is an appropriate text equivalent of the frame.

Source: [WCAG10-HTML-TECHS].

Note: This allows user agents to display a non-frames version of the same information if they do not support frames or if frames are disabled.

Technique 1.1.10 The link that describes a frame (the value of the `longdesc` attribute of the `frameset` element) must be provided, along with alternative content, within the `noframes` element of a `frameset` element's content.

Category: Warning.

Automatic check: FAILS for `frameset` elements that do not include a link to the `frameset` description, which is the URI in the `frameset` element's `longdesc` attribute.

Source: [WCAG10-HTML-TECHS].

Note: We have not seen any other Accessibility Evaluation Tool perform this check.

Technique 1.1.11 All `iframe` elements must have textual element content.

Category: Semi-automatic.

Automatic check: FAILS for `iframe` elements that have no textual content, or whose content consists only of white space. SUCCEEDS for all other `iframe` elements; these must pass the MANUAL CHECK.

Manual check: Make sure that the textual content of the `iframe` element is an appropriate text equivalent of the inline frame.

Source: [WCAG10].

Note: This requirement enables the content to transform gracefully for user agents that do not support inline frames.

Modifications: [WCAG10] contains no provisions about NULL or BLANK ALT text.

Technique 1.1.12 The link that describes an inline frame (the content of the `longdesc` attribute of the `iframe` element) must be provided, along with alternative content, within the element content of the `iframe` element.

Category: Warning.

Automatic check: FAILS for `iframe` elements that do not include a link to the `iframe` description, which is the URI in the `iframe` element's `longdesc` attribute, in their element content.

Source: This is an adaptation of Technique 1.1.10 to `iframes`. We have not seen any other Accessibility Evaluation Tool perform it.

Technique 1.1.13 All `area` elements must have a valid `alt` attribute.

Category: Semi-automatic.

Automatic check: FAILS for all `area` elements that have no `alt` attribute or whose ALT text is empty or consists only of white space. SUCCEEDS for all other `area` elements; these must pass the MANUAL CHECK.

Manual check: Ensure that the value of the `alt` element is an appropriate text equivalent of the geometric region that the `area` element describes.

Source: [WCAG10].

Modifications: [WCAG10] contains no provisions about NULL or BLANK ALT text.

Technique 1.1.14 A `noscript` element is required if the `script` element is found in the document.

Category: Semi-automatic.

Automatic check: FAILS if there is a `script` element in the document, but there are no `noscript` elements in the body of the document. SUCCEEDS otherwise, and all `script` and `noscript` elements are returned for a MANUAL CHECK.

Manual check: Make sure that the `noscript` element(s) provide(s) functionality that is equivalent to that provided by client-side scripts.

Note: According to [HTML4], the content of a `noscript` element will be rendered in the following cases:

- the user agent is configured not to process client-side scripts;
- the user agent does not support the scripting language specified by a `script` element.

Providing the `noscript` element ensures graceful transformation.

Source: [AERT, Technique 1.1.10].

Note: The `noscript` element must contain accessible HTML.

Note: If the content represented by the `script` element is dynamic, make sure that the contents of the `noscript` element also change accordingly.

Modifications: [AERT, Technique 1.1.10] requires exactly one `noscript` element for each script element found in the document, which must immediately follow the `script` element. This does not make any sense if the `script` element appears in the `head` section of the page. Moreover, discussion on the WAI mailinglists indicates that there is no consensus on requiring multiple `noscript` elements, and that it is unclear how different user agents would handle multiple `noscript` elements. We thus require *at least* one `noscript` element in the `body` section of a page.

Technique 1.1.15 Appropriate text equivalents must be provided for character-based artwork (ASCII art).

Category: Full Manual.

Manual check: All ASCII art must have an associated text equivalent.

Source: [WCAG10].

Note: It is possible to use heuristic algorithms to detect ASCII art. [144] is an example of such an algorithm, however it gives a number of false positives.

Note: Implementations of this methodology should prompt the user to substitute ASCII art with an accessible image by using `alt` and `longdesc` attributes to provide equivalent text.

Technique 1.1.16 Verify whether long descriptions and/or D-links are necessary for images.

Category: Manual.

Manual check: Determine whether long descriptions and/or D-links are necessary for `img` elements and `object` elements of `type="image"`. The amount of information in the image and the context in which it is used will determine how detailed the description should be This is not necessary for graphical bullets and images used for horizontal rules.

Source: [AERT, Technique 1.1.2].

Modifications: Apply the technique to *all* images, including `object` elements of `type="image"`.

Technique 1.1.17 Text equivalents must be provided for *linked* audio and video files.

Category: Manual.

Manual check: Make sure that for all links (the `a` element with the `href` attribute) to audio files, the audio content is described within the document, or that the document contains a link to a text file that contains an appropriate description, or a full transcript.

Source: [AERT, Technique 1.1.6].

Note: Sound files are detected by sound file suffixes, e.g. `.wav, .mp3, .snd`, etc.

Technique 1.1.18 Text equivalents must be provided for *embedded* audio or video files.

Category: Manual.

Manual check: Ensure that for all `object` elements with type audio or video, the content is described within the document, or that the document contains a link to a file with a text equivalent description, or a full transcript. Also return all `embed` elements for manual inspection.

Source: [AERT, Technique 1.1.7].

Note: Embedded audio and video content should be detected using the `object` element's `type` attribute as well as a check for suffixes on the `data` attribute, and a check for the `classid` attribute for Macromedia Flash objects.

Checkpoint 1.2

"Provide redundant text links for each active region of a server-side image map. [Priority 1]" [WCAG10]

Technique 1.2.1 For all server-side image maps, an alternative list of image map choices must be provided.

Category: Manual.

Manual check: The document must provide text links corresponding to all regions of the server-side image map created by an `img` element with the `ismap` attribute.

Source: [WCAG10].

Note: Server-side image maps should not be used at all if the same functionality can easily be obtained using client-side image maps. Please refer to Technique 9.1.1 for a discussion on server- and client-side image maps.

Technique 1.2.2 For all `input` elements of `type="image"`, the action taken must not depend on the x and y coordinates of the pointing device.

Category: Manual.

Manual check: `input` elements of type image are graphical submit buttons, however they also create a server-side image-map. When an image map region is activated, the x and y coordinates of the pointing device are sent to the server [HTML4].

The server must not take different action depending on the coordinates of the pointing device, because users of non-graphical browsers will be disadvantaged

[WCAG10-HTML-TECHS]. Multiple submit buttons or a client-side image map should be used instead. In case this is not feasible, the page must provide text links corresponding to all regions of the server-side image map.

Source: [WCAG10].

Note: We have not seen any other Accessibility Evaluation Tool perform this check.

Checkpoint 1.3

"Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation. [Priority 1]" [WCAG10]

Technique 1.3.1 Multimedia content must have audio descriptions.

Category: Manual.

Manual check: For all `object` elements with type multimedia, audio descriptions must also provided within the document.

Source: [AERT, Technique 1.3.1].

Note: See Technique 1.1.18 for how to detect embedded multimedia content.

Checkpoint 1.4

"For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation. [Priority 1]" [WCAG10]

Technique 1.4.1 Multimedia content must have synchronized equivalents.

Category: Manual.

Manual check: Identify all multimedia content and make sure that synchronized equivalents are available.

Source: [AERT, Technique 1.4.1].

Note: See Technique 1.1.18 for how to detect embedded multimedia content.

Checkpoint 1.5

"Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map. [Priority 3]" [WCAG10]

Technique 1.5.1 The document must contain text links for each active area of a client-side image map created by an `img` element.

Category: Semi-automatic.

Automatic check: FAILS for `img` elements with the `usemap` attribute if:

- the given `usemap` does not exist in the document, or exists more than once;
- the map contains `area` elements, but the document contains no textual links to each active region.

SUCCEEDS for all other `img` elements with a `usemap` attribute; these must pass the MANUAL CHECK.

Manual check: The content of the `a` elements must be an appropriate description of the link target.

Source: [AERT, Technique 1.5.1].

Note: If `a` elements are used instead of `area` elements to create the image map, only the manual part of the check applies.

Technique 1.5.2 The document must contain text links for each active area of a client-side image map created by an `object` element.

Category: Semi-automatic.

Automatic check: FAILS for `object` elements with the `usemap` attribute if:

- the given `usemap` does not exist in the document, or exists more than once;
- the map contains `area` elements, but the document contains no textual links to each active region.

SUCCEEDS for all other `object` elements with a `usemap` attribute; these must pass the MANUAL CHECK.

Manual check: The text within the `a` elements must be an appropriate text representation of the link target.

Source: [WCAG10-HTML-TECHS].

Note: If `a` elements are used instead of `area` elements to create the image map, only the manual part of the check applies.

Note: There are several ways to create a client-side image map with the `object` element. For accessibility purposes, the best method is probably to enclose the `map` element inside the `object` element using `a` elements to define the shapes of the image map, which renders the image map's contents when the object is not rendered. An alternative is to place the map element outside the object element, which always renders the image map's contents.

A.6. Guideline 2. Don't rely on color alone.

"Ensure that text and graphics are understandable when viewed without color." [WCAG10]

Checkpoint 2.1

"Ensure that all information conveyed with color is also available without color, for example from context or markup. [Priority 1]" [WCAG10]

Technique 2.1.1 Information conveyed with color must also be available without color.

Category: Manual.

Manual check: If the document contains any of the following color-possible elements, make sure that any information conveyed with color is also available without color:

- `img, applet, object, script, input`
- `body bgcolor | text | alink | vlink | background`
- `table | tr | td | th bgcolor`
- `any_element style=''any_color_specification''`
- `style ''any_color_specification''`

`''any_color_specification''` is defined as any CSS specification which contains:

```
color | background | background-color | background-image
border | border-color | outline | outline-color
content | list-style-image
```

Source: [AERT, Technique 2.1.1].

Modifications: [AERT, Technique 2.1.1] also mentions the `hr` element with the `color` attribute, and the `table` element with the `bordercolor` attribute. However, there is no such thing in the HTML specification [HTML4]. Such issues will be taken care of by Technique 3.2.1.

Checkpoint 2.2

"Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen. [Priority 2 for images, Priority 3 for text]." [WCAG10]

Technique 2.2.1 Foreground and background colors of all text content must provide sufficient contrast.

Category: Automatic.

Automatic check: FAILS for elements where the contrast between foreground and background color is not sufficient. According to "Techniques for Accessibility Evaluation and Repair Tools" [AERT, Technique 2.2.1], color visibility can be determined by to the following algorithm, which is a direct excerpt from that document:

Two colors provide good color visibility if the brightness difference and the color difference between the two colors are greater than a set range. Brightness difference is determined by the following formula:

```
bright_diff = ((red * 299) + (green * 587) + (blue * 114)) / 1000
```

Color difference is determined by the following formula:

```
color_diff = max(red1, red2)     - min(red1,red2) +
             max(green1, green2) - min(green1, green2) +
             max(blue1, blue2)   - min(blue1, blue2)
```

The range for brightness difference is 125. The range for color difference is 500.

Source: [AERT, Technique 2.2.1].

Note: This is complex because CSS files and any CSS references in the document have to be parsed, and applied to the HTML code to determine, for each HTML element, the foreground and background color of text content.

Technique 2.2.2 Foreground and background colors of images must provide sufficient contrast.

Category: Automatic (if image processing is available).

Manual check: Check that the foreground and background color of images provide sufficient contrast. Images are `img` elements and `object` elements of `type=image`

Source: [AERT, Technique 2.1.1].

Note: Implementing this technique requires image processing capabilities. Also note that not all image file formats have the concept of a background-color.

A.7. Guideline 3. Use markup and style sheets and do so properly.

“Mark up documents with the proper structural elements. Control presentation with style sheets rather than with presentation elements and attributes.” [WCAG10]

Checkpoint 3.1

“When an appropriate markup language exists and is supported, use markup rather than raster-based images to convey information. [Priority 2]” [WCAG10]

Technique 3.1.1 Make sure that no part of the document needs to be replaced by a more appropriate markup language.

Category: Manual.

Manual check: Identify all `pre, img, object and applet` elements. When an appropriate markup language exists, use markup rather than images to convey information. For instance, use MathML to mark up mathematical equations, and use style sheets to format text content layout.

Source: [AERT, Technique 3.1.1].

Checkpoint 3.2

“Create documents that validate to published formal grammars. [Priority 2]” [WCAG10]

Technique 3.2.1 Verify that the document contains a published `!doctype` declaration and validate the document, as well as CSS style sheet information used within the document or included from an external source

Category: Automatic (if SGML/XML/CSS Schema validation is available).

Automatic check: This check applies to the (X)HTML document as well as any CSS code it contains both inline or referenced externally using `link` with `rel="stylesheet"`. The (X)HTML document must contain a valid published `!doctype` declaration[46].

Validate the (X)HTML document using the specified document type and a SGML parser for HTML, or XML parser for XHTML. We suggest using the W3C Validation service, which is available under the W3C software license.

[46]The use of the `doctype` attribute on the `html` element is deprecated.

Validate the CSS style rules from style attribute, the `style` element, as well as CSS files from external sources using a CSS Parser. The W3C CSS Validator can be used to check for valid CSS.

Source: [WCAG10].

Note: Be aware that the HTML specification imposes syntax restrictions that cannot be expressed with a DTD, and are therefore not checked by the W3C Markup Validator. Using the W3C Markup Validator to implement this technique is sufficient under this Evaluation Methodology, though other validators, such as Relax NG, are available.

Checkpoint 3.3

"Use style sheets to control layout and presentation. [Priority 2]" [WCAG10]

Technique 3.3.1 Warn if the document uses no style sheets.

Category: Warning.

Automatic check: FAILS if the document does not contain at least one of the following:

- `link rel=''stylesheet''` within the `head` element;
- a `style` element;
- a `style` attribute on any element.

Source: [AERT, Technique 3.3.1]:

Technique 3.3.2 Make sure that the document uses style sheets.

Category: Manual.

Manual check: Make sure the document does not misuse any of the following elements for presentation:
`img` | `font` | `basefont` | `center` | `u` | `strong` | `em`
`any_element align` | `valign` | `border` | `hspace` | `vspace`
`vlink` | `link` | `alink`.

Source: [AERT, Technique 3.3.1].

Modification: Added several properties not in [AERT, Technique 3.3.1].

Technique 3.3.3 The document must not use the `i` and `b` elements.

Category: Automatic.

Automatic check: FAILS if `i` or `b` elements are used.

Source: [WCAG10-HTML-TECHS].

Note: The proper HTML elements should be used to mark up emphasis: `em` and `strong`. The `i` and `b` elements must not be used; they create a visual presentation effect.

Technique 3.3.4 Make sure that `em` and `strong` are used according to specification.

Category: Manual.

Manual check: Make sure that `em` and `strong` are used for emphasis, not presentation.

Source: [WCAG10-HTML-TECHS].

Note: CSS should be used for presentation effects; `em` and `strong` may be rendered in a variety of ways.

Checkpoint 3.4

"Use relative rather than absolute units in markup language attribute values and style sheet property values. [Priority 2]" [WCAG10]

Technique 3.4.1 The document must use relative units of measure.

Category: Automatic.

Automatic check: FAILS for any of the following elements (or style rules) that have the described attributes (or properties) but do not meet the required conditions:

Relative units of measure, using a percentage rate or a relative length, must be used for the values of the following combination of HTML elements and attributes:

- colgroup: width
- col: width

A comma-separated list of relative units of measure must be used for the values of the `rows` and `cols` attributes of the `frameset` element. A percentage rate must be used for the values of the following combination of HTML elements and attributes:

- table: width, cellspacing, cellpadding

- colgroup, col, thead, tbody, tfoot, tr, th, td: charoff
- hr: width
- td, th: width, height
- iframe: width, height

The following CSS properties must use any of the `em`, `ex` or `px` units if a length is specified as the property's value: `background-position`, `border-spacing`, `bottom`, `font-size`, `height`, `left`, `letter-spacing`, `line-height`, `marker-offset`, `max-height`, `max-width`, `min-height`, `min-width`, `right`, `size`, `text-indent`, `text-shadow`, `top`, `vertical-align`, `width`, `word-spacing`. Some of these properties accept lists of lengths as their value. Please refer to the CSS specification [CSS2] for more information.

Source: [HTML4], [CSS2].

Checkpoint 3.5

"Use header elements to convey document structure and use them according to specification. [Priority 2]" [WCAG10]

Technique 3.5.1 Header elements must be properly nested.

Category: Automatic.

Automatic check: FAILS for header elements that do not conform to the following rules:

- header levels can only increase by one level.

  For example: `h2` may follow `h1`, but `h3` cannot follow `h1`.
- header levels can decrease by any level.

  For example: `h3` may follow `h5`.

Source: [WCAG10-HTML-TECHS].

Technique 3.5.2 The document must not be missing any header markup.

Category: Warning.

Automatic check: FAILS for all `p` elements that meet *all* of the following requirements:

- it contains text elements;

- it contains < 10 words;
- all text is formatted in bold and/or italicized and/or underlined.

Any `p` element that meets these requirements may be a heading not marked up as such.

Source: [AERT, Technique 3.5.2].

Technique 3.5.3 Header elements must not be used for presentation.

Category: Warning.

Automatic check: FAILS for header element whose text content is longer than 20 words (may be misused for presentation).

Source: [AERT, Technique 3.5.3].

Note: In this context, we understand "words" to mean character sequences separated by white space.

Technique 3.5.4 Header elements must convey document structure and be used according to specification.

Category: Manual.

Manual check: Make sure that all headings are properly marked up as such, and that headings are not misused for formatting purposes.

Source: [WCAG10].

Checkpoint 3.6

"Mark up lists and list items properly. [Priority 2]" [WCAG10]

Technique 3.6.1 Lists and list items must be nested properly.

Category: Automatic.

Automatic check: FAILS for `ul` and `ol` list containers that do not enclose at least one `li` element, and `dl` list containers that do not enclose at least one `dt/dd` pair.

Source: [AERT, Technique 3.6.1].

Note: Lists may be nested, but according to the HTML specification, nested `ol` or `ul` elements must be enclosed in an `li` element, and nested `dl` elements must be enclosed in a `dd` element.

Technique 3.6.2 Lists must not be used for presentation.

Category: Warning.

Automatic check: FAILS for `ul` or `ol` elements followed by a single `li` element, or a `dl` tag followed by only one `dt/dd` pair UNLESS the list container has a parent list container.

Source: [AERT, Technique 3.6.1].

Modifications: [AERT, Technique 3.6.1] mentions a single `li` element following a `dl` element. However, according to the HTML specification, `dl` elements, which define definition lists, are to be followed by a pair of "term" and "description" elements: the `dt` and `dd` elements [HTML4].

We also allow lists that contain single list items if they are enclosed in another list element.

Technique 3.6.3 Images must not be used as bullets in definition lists.

Category: Warning.

Automatic check: FAILS for `dl` elements enclosing `dt` elements enclosing images.

Source: [WCAG10-HTML-TECHS].

Technique 3.6.4 Lists and list items must be marked up properly.

Category: Manual

Manual check: Verify that

- all lists are marked up using list elements;
- there are no list elements misused for formatting purposes;
- nested lists use compound numbers, or make sure the numbering patterns differ.

Source: [WCAG10].

Checkpoint 3.7

"Mark up quotations. Do not use quotation markup for formatting effects such as indentation. [Priority 2]" [WCAG10]

Technique 3.7.1 Identify instances where quote markup may be missing.

Category: Warning.

Automatic check: FAILS for any text meeting the following requirements:

- any text that is enclosed by quote marks;
- more than 10 words of emphasized text.

Text detected by this technique may be a potential quote not marked up properly.

Source: [Technique 3.7.1]tr-aert.

Technique 3.7.2 The `q` and `blockquote` elements must be used according to specification.

Category: Warning.

Automatic check: FAILS

- for inline quotes (marked up with `q`) with more than 10 words or no word before and no word after the quote;
- for long quotes (marked up with `blockquote`) with less than 10 words.

Source: [AERT, Technique 3.7.1].

Technique 3.7.3 The `blockquote` element must not be used for presentation.

Category: Warning.

Automatic check: FAILS nested `blockquote` elements.

Source: [AERT, Technique 3.7.3].

Technique 3.7.4 Quotations must be marked up properly.

Category: Full Manual.

Manual check: Make sure that

- all quotations in the document are marked up properly, using `q` for inline quotes and `blockquote` for long quotes;
- quotation markup is not misused to create formatting effects.

Source: [AERT, Technique 3.7.3].

A.8. Guideline 4. Clarify natural language usage.

"Use markup that facilitates pronunciation or interpretation of abbreviated or foreign text." [WCAG10]

Checkpoint 4.1

"Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions). [Priority 1]" [WCAG10]

Technique 4.1.1 Changes in the natural language of the document must be marked up properly.

Category: Full Manual.

Automatic check: Make sure that any words or phrases that are not the document's primary language must be marked up properly.

Source: [WCAG10].

Checkpoint 4.2

"Specify the expansion of each abbreviation or acronym in a document where it first occurs. [Priority 3]" [WCAG10]

Technique 4.2.1 The `abbr` and `acronym` elements must be used correctly.

Category: Automatic.

Automatic check: FAILS

- for each `abbr` element without a `title` attribute for which the document does not contain a preceding `abbr` element (with the same content) with an expansion;
- same for the `acronym` element.

If for a given element content the document contains different expansions in the `title` attribute, then, to make clear which abbreviation / acronym the element is referring to, all such elements must include the expansion in the `title` attribute.

Source: This technique is our own.

Technique 4.2.2 Identify potential abbreviations and acronyms that are not marked up properly.

Category: Warning.

Automatic check: FAILS

- any word in all capital letters that is longer than 2 characters and not the content of an `abbr` element;

- any word that starts with a capital letter, contains lower case characters, ends with a period and is not the content of an `acronym` element.
- same for the `acronym` element.

Source: [AERT, Technique 4.2.1].

Modifications: [AERT, Technique 4.2.1] fails to exclude words within `abbr` and `acronym` elements from this technique.

Technique 4.2.3 Abbreviations and acronyms must be marked up properly.

Category: Manual

Manual check: Make sure that abbreviations and acronyms are marked up correctly throughout the document.

Source: [WCAG10].

Checkpoint 4.3

"Identify the primary natural language of a document. [Priority 3]" [WCAG10]

Technique 4.3.1 The primary language of a document must be specified.

Category: Automatic.

Automatic check: FAILS if the document does not contain a `html` element, or if the `html` element does not contain a valid `lang` attribute. Please refer to the HTML 4.01 specification [HTML4] and [RFC1766] for further information about valid `lang` attributes.

Source: [WCAG10].

Note: It is sometimes stated that the value of the `lang` attribute must be one of the ISO 639 language codes. This is not entirely correct, because ISO 639 only defines two-letter primary language codes, and codes such as "en-US" or "i-cherokee" are valid language codes for HTML. The language codes that must be used in HTML documents are defined in RFC 1766 [RFC1766].

Note: At the time of writing the list of ISO 639 language codes on the W3C Web site is out of date.

A.9. Guideline 5. Create tables that transform gracefully.

"Ensure that tables have necessary markup to be transformed by accessible browsers and other user agents." [WCAG10]

Checkpoint 5.1

"For data tables, identify row and column headers. [Priority 1]" [WCAG10]

Technique 5.1.1 Data tables must have row and/or column headers.

Category: Automatic.

Automatic check: FAILS a `table` unless it has at least one complete set of row *or* column headers.

Source: [WCAG10].

Note: According to this Methodology, tables need not be used for layout; therefore *all* tables must have row or column headers. See Section A.4 for more information.

Checkpoint 5.2

"For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells. [Priority 1]" [WCAG10]

Technique 5.2.1 Data tables with two or more logical levels of headers must use markup to associate data and header cells.

Category: Manual.

Manual check: Tables with two or more logical levels of headers must be marked up properly.

Source: [WCAG10].

Checkpoint 5.3

"Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version). [Priority 2]" [WCAG10]

Technique 5.3.1 Tables must not be used for layout.

Category: Manual.

Manual check: Ensure that there are no tables used for layout purposes.

Source: [WCAG10].

Note: According to this Methodology, tables need not be used for layout. See Section A.4 for more information.

Checkpoint 5.4

"If a table is used for layout, do not use any structural markup for the purpose of visual formatting. [Priority 2]" [WCAG10]

Note: Checkpoint does not apply. Tables need not be used for layout. See Section A.4 for more information.

Checkpoint 5.5

"Provide summaries for tables. [Priority 3]" [WCAG10]

Technique 5.5.1 All `table` elements must have a `summary` attribute.

Category: Semi-automatic.

Automatic check: FAILS for `table` elements with no `summary` attribute, or a `summary` attribute whose value is "empty" or consists only of white space. SUCCEEDS for all other `table` elements; these must pass the MANUAL CHECK.

Manual check: Make sure the value of the `summary` attribute is an appropriate description of the corresponding `table`.

Source: [WCAG10].

Modifications: [WCAG10] contains no provisions about NULL or BLANK values.

Technique 5.5.2 All `table` elements must either have a `title` attribute or enclose a `caption` element.

Category: Semi-automatic.

Automatic check: FAILS for `table` elements which do not enclose a `caption` element *and* do not have a `title` attribute. SUCCEEDS for all other `table` elements; these must pass the MANUAL CHECK.

Manual check: Make sure that the contents of the `caption` element and the value of the `title` attribute are appropriate descriptions of the corresponding `table`.

Source: [AERT, Technique 5.5.2].

Modifications: Require *either* a `title` attribute *or* a `caption` element.

Checkpoint 5.6

"Provide abbreviations for header labels. [Priority 3]" [WCAG10]

Technique 5.6.1 All `th` elements must have a valid `abbr` attribute.

Category: Semi-automatic.

Automatic check: FAILS all `th` elements that do not have an `abbr` attribute, or have an `abbr` attribute whose value is empty, or consists only of white space, or be "abbr". SUCCEEDS for all other `th` elements; these must pass the MANUAL CHECK.

Manual check: Make sure that the value of the `abbr` attribute is an appropriate abbreviation of the corresponding header.

Source: [AERT, Technique 5.6.1].

Modifications: [AERT, Technique 5.6.1] requires an abbreviation only for header names longer than 15 characters. We require abbreviations for *all* header elements. Moreover, *spaces* has been changed to *white space.*

A.10. Guideline 6. Ensure that pages featuring new technologies transform gracefully.

"Ensure that pages are accessible even when newer technologies are not supported or are turned off." [WCAG10]

Checkpoint 6.1

"Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document. [Priority 1]" [WCAG10]

Technique 6.1.1 The document must be readable without style sheets.

Category: Manual.

Manual check: Make sure that the document can be read without style sheets. This technique applies if the document contains one of the following:

- a `link` element with `rel="stylesheet"`;
- a `style` element;
- at least one `style` attribute on any element.

Source: [AERT, Technique 6.1.1].

Checkpoint 6.2

"Ensure that equivalents for dynamic content are updated when the dynamic content changes. [Priority 1]" [WCAG10]

Technique 6.2.1 The source of `frame` and `iframe` elements must be valid markup files.

Category: Automatic.

Automatic check: FAILS for `frame` and `iframe` elements whose `src` attributes do not point to a valid HTML file. If the `HTTP Content-Type` header is available, it must indicate a valid (X)HTML file. According to [138], valid (X)HTML media types are:

`text/html, application/xhtml+xml, text/xml` and `application/xml`.

Source: [AERT, Technique 6.1.1].

Modifications: [AERT, Technique 6.1.1] would match the value of the `src` attribute against file suffixes that are known to be or generate HTML files. This is unfeasible and error-prone. We think it makes more sense to check the `HTTP Content-Type` for valid (X)HTML media types.

Technique 6.2.2 Identify all elements that could generate dynamic content.

Category: Manual.

Manual check: Elements that could generate dynamic content are programmatic objects and intrinsic event handlers. If the document contains any of the following, human judgment must determine whether equivalents for dynamic content are kept up to date:

- an `object` element with either a `classid` attribute or a `type` (or `codetype`) attribute of value `application/*`;
- a `script` element;
- an `applet` element;
- an `embed` element;
- `any_element intrinsic_event_handler=anything`.

Please refer to the HTML specification [HTML4] for a list of intrinsic event handlers. Also note that intrinsic event handlers that generate dynamic content are only to be taken into account if they do not merely change the presentation of the document.

Source: [AERT, Technique 6.2.2].

Note: For `object` elements that have a `type` attribute of `application/*`, the object SHOULD be requested using `HTTP HEAD`, because according to the HTML 4.01 specification [HTML4], the `HTTP Content-Type` takes precedence over the value of the `type` attribute.

Modifications: We added the check for the `HTTP Content-Type` for `object` elements.

Checkpoint 6.3

"Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page. [Priority 1]" [WCAG10]

Technique 6.3.1 A page must be usable when programmatic objects are disabled.

Category: Manual.

Manual check: This test applies only when the document contains any of the following:

- an `object` element with either a `classid` attribute or a `type` (or `codetype`) attribute with a value of `application/*` [AERT, Technique 6.3.1];
- a `script` element [AERT, Technique 6.3.1];
- an `applet` element [AERT, Technique 6.3.1];
- Javascript URLs in the `area` or `a` elements [WCAG10].

Source: See list.

Note: For `object` elements that have a `type` attribute of `application/*`, the object SHOULD be requested using `HTTP HEAD`, because according to the HTML 4.01 specification [HTML4], the `HTTP Content-Type` takes precedence over the value of the `type` attribute.

Modifications: We added the check for the `HTTP Content-Type` for `object` elements.

Checkpoint 6.4

"For scripts and applets, ensure that event handlers are input device-independent. [Priority 2]" [WCAG10]

Technique 6.4.1 If device-dependent intrinsic event handlers are used, they must not require the use of a pointing device.

Category: Automatic.

Automatic check: FAILS if elements that use a device-dependent intrinsic event handler requiring the use of a pointing device do not also include the keyboard equivalent event handler, or if the device-dependent event handler does not have a keyboard equivalent event handler.

Device-dependent intrinsic event handlers defined in the HTML specification are `on*click, onmouse*, onkey*`. Please refer to the HTML specification [HTML4] for a full list of intrinsic event handlers.

The following list includes event handlers specific to a pointing device and their keyboard equivalents:

- `onmousedown <=> onkeydown`
- `onmouseup <=> onkeyup`
- `onclick <=> onkeypress`
- `onmouseover <=> onfocus`
- `onmouseout <=> onblur`

Note: There is no keyboard equivalent for double clicking (`ondblclick`) or mouse movement (`onmousemove`) in HTML 4.01. These event handlers must therefore not be used at all.

Source: [AERT, Technique 9.5.1].

Modifications: [AERT, Technique 9.5.1] suggests that the `onkey*` event handlers are device independent, which they are not; some of them are the keyboard equivalents to event handlers associated with a pointing device.

Technique 6.4.2 Verify that any programmatic objects are usable independently of the input device.

Category: Manual.

Manual check: This check applies only when the document contains programmatic objects:

- an `object` element with either a `classid` attribute or a `type` (or `codetype`) attribute of value `application/*`;
- a `script` element;
- an `applet` element;
- an `embed` element.

The functionality provided by these objects must not depend on the input device.

Source: [AERT, Technique 6.4.1].

Note: For `object` elements that have a `type` attribute of `application/*`, the object SHOULD be requested using `HTTP HEAD`, because according to the HTML 4.01 specification [HTML4], the `HTTP Content-Type` takes precedence over the value of the `type` attribute.

Modifications: We added the check for the `HTTP Content-Type` for `object` elements.

Checkpoint 6.5

"Ensure that dynamic content is accessible or provide an alternative presentation or page. [Priority 2]" [WCAG10]

Note: Part of this checkpoint should be covered with the techniques for checkpoints 6.2 to 6.4 as well as Technique 1.1.9.

Technique 6.5.1 Warn if documents that use programmatic objects or Javascript do not provide a text-only page.

Category: Warning.

Automatic check: FAILS if programmatic objects or javascript links are used in the document and no text-only version is available.

A text-only version of a document is identified by a `link` element with `rel="alternate"` [HTML4] and a `media` attribute of `Braille`, `aural` or `tty` [WCAG10-HTML-TECHS].

The `link` element must not have a `hreflang` attribute, as this would imply a translated version of the document in which the link occurs [HTML4][47].

Please refer to the previous checkpoints in this section on how to identify programmatic objects and Javascript links.

Source: [WCAG10].

Note: Unfortunately, many sites do not identify text-only versions of their documents with the `link` element.

[47]The HTML 4.01 specification states that the alternate link type combined with the `lang` attribute is used to specify a translated version of the document in which it occurs. This is one of the known HTML 4.01 errata [146].

Technique 6.5.2 Ensure that dynamic content is accessible or an alternative presentation or page is being provided.

Category: Manual.

Manual check: If the document provides dynamic content, make sure that it is accessible, or that an alternative presentation or page is provided.

Source: [WCAG10].

A.11. Guideline 7. Ensure user control of time-sensitive content changes.

"Ensure that moving, blinking, scrolling, or auto-updating objects or pages may be paused or stopped.
Note. *All of the following checkpoints involve some content developer responsibility until user agents provide adequate feature control mechanisms. "* [WCAG10]

Checkpoint 7.1

"Until user agents allow users to control flickering, avoid causing the screen to flicker. [Priority 1]" [WCAG10]

Technique 7.1.1 Identify all elements that could cause the screen to flicker.

Category: Manual.

Manual check: Elements that produce any kind of motion on the screen could cause it to flicker. This includes any programmatic elements as well as embedded movies and animated images:

- all `img` elements with a filename that ends in any of the following: `.gif, .mng`;
- all `object` elements of `type="image"` with a filename that ends in any of the following: `.gif, .mng`;
- all `object` elements of type video;
- `object` elements with either a `classid` attribute or a `type` (or `codetype`) attribute of value `application/*`;
- all `script` elements;
- all `applet` elements;
- all `embed` elements.

Source: [AERT, Technique 7.1.1].

Note: It should be possible to automatically detect animated images with special software algorithms.

Note: For `object` elements that have a `type` attribute of `application/*`, the object SHOULD be requested using `HTTP HEAD`, because according to the HTML 4.01 specification [HTML4], the `HTTP Content-Type` takes precedence over the value of the `type` attribute.

Modifications: We added the check for the `HTTP Content-Type` for `object` elements.

Checkpoint 7.2

"Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off). [Priority 2]" [WCAG10]

Technique 7.2.1 The `blink` element must not be used.

Category: Automatic.

Automatic check: FAILS for all `blink` elements.

Source: [WCAG10].

Note: There is no such thing as a `blink` element in the HTML specification [HTML4]. This test was included anyway because it is mentioned in the Web Content Accessibility Guidelines [WCAG10], as well as in many documents referencing them.

Technique 7.2.2 The CSS property `text-decoration` with value `blink` must not be used.

Category: Warning.

Automatic check: FAILS for HTML elements or CSS rules with the CSS property `text-decoration` with a value of `blink`.

Source: [CSS2].

Note: Since users may turn this off either by turning off style sheets or overriding the value in a user style sheet [WCAG10-CSS-TECHS], this technique is categorized as a warning.

Technique 7.2.3 Identify all elements that could cause the screen to blink.

Category: Manual.

Automatic check: Please refer to Technique 7.1.1 for a list of elements that could cause the screen to blink.

Source: This technique is our own.

Checkpoint 7.3

"Until user agents allow users to freeze moving content, avoid movement in pages. [Priority 2]" [WCAG10]

Technique 7.3.1 The `marquee` element must not be used.

Category: Automatic.

Automatic check: FAILS for `marquee` elements.

Source: [AERT, Technique 7.3.1].

Note: There is no such thing as a `marquee` element in the HTML specification [HTML4]. This test was included anyway because it is mentioned in the Web Content Accessibility Guidelines [WCAG10], as well as in many documents referencing them.

Technique 7.3.2 Identify all elements that could cause content to move.

Category: Manual.

Automatic check: Please refer to Technique 7.1.1 for a list of elements that could cause content to move.

Source: [AERT, Technique 7.3.2].

Checkpoint 7.4

"Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages. [Priority 2]" [WCAG10]

Technique 7.4.1 Auto-refresh attributes must not be used in the `meta` element.

Category: Automatic.

Automatic check: FAILS for `meta` elements with `http-equiv="refresh"` and a `content` attribute with an integer value greater than zero.

Source: [AERT, Technique 7.4.1].

Note: According to the HTML specification, this use of the `meta` element is supported only by some user agents [HTML4].

Technique 7.4.2 Identify all elements that could cause a refresh.

Category: Manual.

Manual check: Aside from `meta http-equiv`, only `script` elements can cause a refresh. Identify all `script` elements for manual inspection.

Source: This technique is our own.

Checkpoint 7.5

"Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects. [Priority 2]" [WCAG10]

Technique 7.5.1 Auto-redirect attributes must not be used in the `meta` element.

Category: Automatic.

Automatic check: FAILS for `meta` elements with `http-equiv="refresh"` and a `content` attribute having a value of the type `integer; URL=<URI>`.

Source: [AERT, Technique 7.5.1].

Note: Server-side redirects should be used to forward users to different pages.

Technique 7.5.2 Identify all elements that could cause a redirect.

Category: Manual.

Manual check: Identify all `scripts` element and check whether they may cause a redirect.

Source: This technique is our own.

A.12. Guideline 8. Ensure direct accessibility of embedded user interfaces.

"Ensure that the user interface follows principles of accessible design: device-independent access to functionality, keyboard operability, self-voicing, etc." [WCAG10]

Checkpoint 8.1

"Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise Priority 2.]" [WCAG10]

Technique 8.1.1 Identify all programmatic elements.

Category: Manual.

Manual check: Make sure that the following programmatic elements are directly accessible or compatible with assistive technologies:

- an `object` element with either a `classid` attribute or a `type` (or `codetype`) attribute with a value of `application/*` [AERT, Technique 6.3.1];
- a `script` element [AERT, Technique 6.3.1];
- an `applet` element [AERT, Technique 6.3.1];
- Javascript URLs in the `area` or `a` elements [WCAG10].

Source: [WCAG10].

Note: This technique is strongly tied to checkpoint 6.4.

Note: For `object` elements that have a `type` attribute of `application/*`, the object SHOULD be requested using `HTTP HEAD`, because according to the HTML 4.01 specification [HTML4], the `HTTP Content-Type` takes precedence over the value of the `type` attribute.

Modifications: We added the check for the `HTTP Content-Type` for `object` elements.

A.13. Guideline 9. Design for device-independence.

"Use features that enable activation of page elements via a variety of input devices." [WCAG10]

Checkpoint 9.1

"Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape. [Priority 1]" [WCAG10]

Technique 9.1.1 Identify server-side image maps within the document.

Category: Warning.

Automatic check: FAILS if server-side image maps are used within the document.

Server-side image maps are created by the following elements [HTML4]:

- `img` elements with the `ismap` attribute;
- `input` elements of `type="image"`.

Source: [WCAG10].

Note: Since any arbitrary shape can be defined by a polygon, according to this checkpoint, this means that server-side image maps should not be used at all.

Our interpretation of this checkpoint however is that server-side image maps should only be used if the shapes involved are quite complex, and the use of a client-side image map would be difficult and tedious. Server-side image maps are still important for applications like geographical information systems and mapping applications where each point or coordinate pair is an active region [119]; these are examples in which this warning is meant to be overridden by the reviewer.

Checkpoint 9.2

"Ensure that any element that has its own interface can be operated in a device-independent manner. [Priority 2]" [WCAG10]

Note: This checkpoint is strongly tied to checkpoints 6.4 and 8.1. The issue should already have been addressed within those two checkpoints.

Checkpoint 9.3

"For scripts, specify logical event handlers rather than device-dependent event handlers. [Priority 2]" [WCAG10]

Note: This issue has been addressed in checkpoint 6.4.

Checkpoint 9.4

"Create a logical tab order through links, form controls, and objects. [Priority 3]" [WCAG10]

Technique 9.4.1 Links, form controls and objects must provide a logical tab order.

Category: Semi-automatic.

Automatic check: FAILS for any of the following elements that do not have a valid `tabindex` attribute:

- `a`
- `area`
- `button`
- `input`
- `object`
- `select`
- `textarea`

A valid value for the `tabindex` attribute is natural number between 0 and 32767 [HTML4].

SUCCEED for all listed elements with a valid `tabindex` attribute; these must pass the MANUAL CHECK.

Manual check: The tab order through the controls must be sensible.

Source: [WCAG10].

Checkpoint 9.5

"Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls. [Priority 3]" [WCAG10]

Technique 9.5.1 Keyboard shortcuts must be provided using the `accesskey` attribute.

Category: Warning.

Automatic check: FAILS for any of the following elements that do not have a valid `accesskey` attribute:

- `a`
- `area`
- `button`
- `input`

- `label`
- `legend`
- `textarea`

Source: [AERT, Technique 9.5.1].

Note: The value of the `accesskey` attribute must be a single character.

Note: In 2002, the Canadian Web Accessibility services company WATS.ca conducted a non-scientific study [147] to determine whether implementing access keys would result in conflicts with assistive technologies, which use a wide variety of keyboard shortcuts. The research has shown that most keyboard combinations are already in use, and WATS.ca have subsequently recommended that access keys be avoided. The W3C has since deprecated the *accesskey* attribute in its XHTML 2 working drafts. For this reason this technique categorized as a warning.

A.14. Guideline 10. Use interim solutions.

"Use interim accessibility solutions so that assistive technologies and older browsers will operate correctly.
Note. *The following checkpoints apply until user agents (including assistive technologies) address these issues. These checkpoints are classified as "interim", meaning that the Web Content Guidelines Working Group considers them to be valid and necessary to Web accessibility as of the publication of this document. However, the Working Group does not expect these checkpoints to be necessary in the future, once Web technologies have incorporated anticipated features or capabilities."* [WCAG10]

Checkpoint 10.1

"Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user. [Priority 2]" [WCAG10]

Technique 10.1.1 Frame targets must not open a new window.

Category: Warning.

Automatic check: FAILS for `a`, `area` [AERT, Technique 10.1.1] and `base` [HTML4] elements that have an invalid `target` attribute of `_blank` or `_new`.

Source: See description.

Note: This technique is categorized as a warning because Web developers may allow the user to chose whether or not to open links in new windows, e.g. by allowing the user to set her preferences and generating dynamic page content, or by letting the user know that a link will open a new window.

Note: New windows will also be spawned if a non-existent `target` name is used. The value of the `target` attribute may also reference a window name, e.g. a window previously opened by a script.

Technique 10.1.2 Scripts must not spawn new windows.

Category: Automatic.

Automatic check: FAILS for `script` elements and intrinsic event handlers that do any of the following:

- a call of `window.open()` [AERT, Technique 10.1.2];
- a call of `open()`;
- a call of `document.open()` with more than two parameters.

Source: In part [AERT, Technique 10.1.2], in part our own (see description).

Technique 10.1.3 If the document contains any scripts or intrinsic event handlers, ensure that the document does not spawn any new windows, or change the current window without informing the user.

Category: Manual.

Manual check: When the document contains any scripts (either in the `script` element, or as the value of intrinsic event handlers), a manual check must be made to ensure that no new windows are spawned, and that the current window is not changed without informing the user.

Source: [WCAG10].

Checkpoint 10.2

"Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned. [Priority 2]" [WCAG10]

Technique 10.2.1 `label` elements must be positioned properly.

Category: Manual.

Manual check: Labels are implicitly associated with form controls either through markup or positioning on the page. A label must immediately precede or follow its control, or group of controls, on the same line if there are multiple controls or groups per line. A label may be in the line preceding a control if there is only one control or group of controls per line.

Source: [WCAG10-HTML-TECHS].

Modifications: WCAG requires that a label immediately precede its control if there are multiple controls or groups per line [WCAG10]. Although this requirement is fine for text input fields, it does not make sense for radio buttons. Related WCAG documents suggest that if there are multiple controls per line, a label may also immediately follow its control [119].

Note: This technique is strongly tied to checkpoint 12.4.

Checkpoint 10.3

"Until user agents (including assistive technologies) render side-by-side text correctly, provide a linear text alternative (on the current page or some other) for all tables that lay out text in parallel, word-wrapped columns. [Priority 3]" [WCAG10]

Technique 10.3.1 A linearized version of data tables must be provided.

Category: Manual.

Manual check: For each table, a linearized text alternative version must be provided.

Source: [AERT].

Note: Check the WAI curriculum [119] and the WAI HTML table linearizer Tablin [148] for information and examples on how to linearize tables.

Checkpoint 10.4

"Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas. [Priority 3]" [WCAG10]

Technique 10.4.1 The `input`, `textarea` and `select` elements must have valid default values.

Category: Warning.

Automatic check: FAILS for `input`, `textarea` and `select` elements that do not have default values:

- `input type=(text | checkbox | radio)` must have at least one word of text in their `value` attribute;
- `textarea` elements must contain textual content;
- one `option` element in each group contained by `select` elements must have a `selected` attribute.

Source: [AERT, Technique 10.4.1].

Note: This technique is categorized as a warning because it is our opinion that by now most user agents should correctly handle empty controls.

Checkpoint 10.5

"Until user agents (including assistive technologies) render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links. [Priority 3]" [WCAG10]

Technique 10.5.1 There must be non-white space text characters between consecutive `a` elements.

Category: Automatic.

Automatic check: FAILS for adjacent `a` elements not separated by non-white space characters.

Two `a` sections are considered adjacent if a user agent would present them within the same line of text (if there were enough space to do so). Two link sections are therefore adjacent if the HTML code between the two `a` elements does not include any block-level elements or the `br` element.

The requirement of this checkpoint is that `a` sections within the same line of text must be separated by printable characters surrounded by white space, in other words, a sequence of `white space, non-white space, white space`.

For the purpose of this checkpoint, white space is considered to be any combination of ASCII space (`&#x0020;`), ASCII tab (`&#x0009;`), ASCII form-feed (`&#x000C;`) and the metacharacter `&nbsp;`. Non-white space is defined as any sequence of printable text characters. Printable text can also be enclosed in HTML elements. Because of the original requirement that both links must be rendered on the same line, only HTML elements that do not cause line-breaks are relevant here. These are (mostly) the HTML inline elements.

According to the HTML 4.01 [HTML4] DTD, inline elements consist of font style, phrase, special and form control elements. The form control elements and the special elements `a, img, object, br, script` and `map` are of no relevance for this checkpoint because they either do not render printable characters, or do so only under certain circumstances. This leaves the following inline elements:

- the `%fontstyle` elements: `tt, i, b, big, small`;
- the `%phrase` elements: `em, strong, dfn, code, samp`, `kbd`, `var`, `cite`, `abbr`, `acronym`;
- the `%special` elements `sub, sup, span, q, bdo`.

Non-white space is thus defined as any sequence of text, optionally enclosed within the HTML elements listed above.

Source: [WCAG10].

Note: According to the HTML 4.01 specification [HTML4], authors should not rely on user agents to render white space immediately after a start tag or immediately before an end tag. Such occurrences of white space should therefore be ignored when searching for a sequence of `white space, non-white space, white space`.

A.15. Guideline 11. Use W3C technologies and guidelines.

"Use W3C technologies (according to specification) and follow accessibility guidelines. Where it is not possible to use a W3C technology, or doing so results in material that does not transform gracefully, provide an alternative version of the content that is accessible." [WCAG10]

Checkpoint 11.1

"Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported. [Priority 2]" [WCAG10]

Technique 11.1.1 W3C technologies must be used where possible and appropriate.

Category: Manual.

Manual check: The following technologies should be replaced with W3C technologies:

- convert PDF, Postscript, RTF and Microsoft word documents into a valid equivalent using HTML, CSS, SMIL and PNG or SVG;
- convert movies into SMIL presentations;

- convert GIF and JPEG images that represent textual information to textual information using style sheets;
- convert any non-PNG images to PNG images;
- convert Flash animations to a combination of valid HTML, SMIL, and PNG or SVG;
- convert images of mathematical equations to MathML.

The following is a list of W3C technologies reviewed for accessibility [WCAG10-HTML-TECHS]:

- use MathML for mathematical equations;
- use HTML, XHTML, XML to represent structured data;
- use RDF to represent metadata;
- use SMIL for multimedia presentations;
- use CSS and XSL to define style sheets;
- use XSLT to transform styles;
- use PNG for images.

Source: [AERT, Technique 11.1.1].

Note: Many of these W3C formats are not widely supported yet. Human judgement is required to determine whether any of the above listed technologies should be replaced with W3C technologies.

Checkpoint 11.2

"Avoid deprecated features of W3C technologies. [Priority 2]" [WCAG10]

Technique 11.2.1 Deprecated HTML elements and attributes must not be used.

Category: Automatic.

Automatic check: FAILS

- for deprecated HTML 4.01 elements: `applet`, `basefont`, `center`, `dir`, `font`, `isindex`, `menu`, `s`, `strike` and `u`;
- for deprecated HTML 4.01 attributes.

The following is a list of deprecated HTML 4.01 attributes and the elements they are deprecated on:

- align: caption, applet, iframe, img, input, object, legend, table, hr, div, h1-h6, p

- alink: body
- alt: applet
- archive: applet
- background: body
- bgcolor: table, tr, td, th, body
- border: img, object
- clear: br
- code: applet
- codebase: applet
- color: basefont, font
- compact: dir, dl, menu, ol, ul
- face: basefont, font
- height: td, th, applet
- hspace: applet, img, object
- language: script
- link: body
- name: applet
- noshade: hr
- nowrap: td, th
- object: applet
- prompt: isindex
- size: hr, font, basefont
- start: ol
- text: body
- type: li, ol, ul
- value: li
- version: html
- vlink: body
- vspace: applet, img, object
- width: hr, td, th, applet, pre

Source: [HTML4].

Note: Please check the HTML specification [HTML4] for the exact meaning of the term "deprecated".

Checkpoint 11.3

"Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.) [Priority 3]" [WCAG10]

Technique 11.3.1 Documents must be served per user preferences.

Category: Manual.

Manual check: If there are multiple versions of the same content, the user must be able to receive documents according to her preference:

- use content negotiation whenever possible;
- link to other versions of the document (other formats or languages);
- if it is not possible to use content negotiation, indicate content type or language through markup in `link` elements (use the `type` and `hreflang` attributes);
- use cookies to remember user preferences between sessions;
- serve style sheets based on user preferences.

Source: [AERT, Technique 11.3.1].

Checkpoint 11.4

"If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible (original) page. [Priority 1]" [WCAG10]

Technique 11.4.1 The page must pass all checkpoints of the desired conformance level.

Category: Depends on result of all other checkpoints and the desired conformance level.

Check: If the page passes all checkpoints of the desired conformance level, it automatically passes this checkpoint as well. In case it does not, it passes this checkpoint only if an alternative page with equivalent information or functionality is provided.

Source: [WCAG10].

A.16. Guideline 12. Provide context and orientation information.

"Provide context and orientation information to help users understand complex pages or elements." [WCAG10]

Checkpoint 12.1

"Title each frame to facilitate frame identification and navigation. [Priority 1]" [WCAG10]

Technique 12.1.1 All `iframe` and `frame` elements must have valid a `title` attribute.

Category: Semi-automatic.

Automatic check: FAILS for `frame` and `iframe` elements that do not have a `title` attribute, or have a `title` element whose value is empty, consists only of white space, or is "title". SUCCEEDS for all other `frame` and `iframe` elements; these must pass the MANUAL CHECK.

Manual check: Verify that the value of the `name` and `title` attributes are appropriate descriptions of the `frame` or `iframe`.

Source: [AERT, Technique 12.1.1].

Modifications: [AERT, Technique 12.1.1] requires the `title` attribute only for the `frame` element; *spaces* has been changed to *white space*.

Checkpoint 12.2

"Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone. [Priority 2]" [WCAG10]

Technique 12.2.1 All `frame` and `iframe` elements must have a valid `longdesc` attribute.

Category: Semi-automatic.

Automatic check: FAILS for `frame` and `iframe` elements without a valid `longdesc` attribute. The value of the `longdesc` attribute must be an URI which must provide a frame description in accessible HTML. If the `HTTP Content-Type` header is available, it must indicate a valid (X)HTML file. According to [138], valid (X)HTML media types are:

`text/html, application/xhtml+xml, text/xml` and `application/xml`.

SUCCEEDS for all other `frame` and `iframe` elements; these must pass the MANUAL CHECK.

Manual check: The URI pointed to by the `longdesc` attribute must be accessible HTML and the content of that document must be an appropriate description of the frame or inline frame, or of relationships between frames.

Source: [WCAG10].

Note: An implementation of this Methodology should prompt the user to evaluate the URIs of `longdesc` attributes for accessibility conformance just like the "current" document.

Checkpoint 12.3

"Divide large blocks of information into more manageable groups where natural and appropriate. [Priority 2]" [WCAG10]

Technique 12.3.1 Information must be grouped where necessary and appropriate.

Category: Manual.

Manual check: Make sure related elements are grouped, if possible: `fieldset` and `legend` must be used to group form controls into semantic units, `optgroup` must be used to organize long lists of menu options into smaller groups, tables must be used for tabular data, ordered and unordered lists must be nested, section headings must be used to create structured documents, long lines of text should be broken up into paragraphs, and related links must be grouped together.

Source: [WCAG10-HTML-TECHS].

Checkpoint 12.4

"Associate labels explicitly with their controls. [Priority 2]" [WCAG10]

Technique 12.4.1 Labels and form controls must be explicitly associated.

Category: Automatic.

Automatic check: FAILS for `label` elements whose `for` element does not match the `id` attribute of a form control.

Form controls are:

- input type = (text | password | checkbox | radio | file | hidden)
- `select`
- `textarea`
- `button`

Source: [AERT, Technique 12.4.1].

Modifications: [AERT, Technique 12.4.1] mentions only `input` elements, no other form controls.

Technique 12.4.2 Identify form controls that have no label associated with them.

Category: Warning.

Automatic check: FAILS for form controls who have no `label` whose `for` attribute is not associated with their `id`.

Source: This technique is our own.

Note: See previous technique for a list of form controls.

A.17. Guideline 13. Provide clear navigation mechanisms.

"Clear and consistent navigation mechanisms are important to people with cognitive disabilities or blindness, and benefit all users." [WCAG10]

Checkpoint 13.1

"Clearly identify the target of each link. [Priority 2]" [WCAG10]

Technique 13.1.1 The target for each link must be clearly identified.

Category: Automatic.

Automatic check: FAILS if the same link phrase and `title` is used on different link targets. If no `title` is given, only the link phrase is checked.

Source: [WCAG10-HTML-TECHS].

Note: Links are represented by `a` and `area` elements.

Technique 13.1.2 Check for suspicious anchor names.

Category: Warning.

Automatic check: FAILS if any anchor name is longer than 60 characters or if it matches any of the following phrases:

- `click here`
- `more`
- `follow this`

Source: [AERT, Technique 13.1.1].

Technique 13.1.3 Link phrases must be appropriate.

Category: Manual.

Manual check: Make sure that link phrases are be meaningful when read out of context, and that the `title` attribute is used to provide additional information where needed. The link text and `title` text must describe the effects of following the link.

Source: [WCAG10-HTML-TECHS].

Checkpoint 13.2

"Provide metadata to add semantic information to pages and sites. [Priority 2]" [WCAG10]

Technique 13.2.1 The document must contain a valid `title` element.

Category: Semi-automatic.

Automatic check: FAILS unless the document contains a `title` element with textual content within the `head` element. SUCCEEDS otherwise; the `title` element must pass the MANUAL CHECK.

Manual check: The value of the `title` element must be appropriate and meaningful when read out of context.

Source: [AERT, Technique 13.2.1].

Modifications: [AERT] requires the presence of either one of `title, address, meta`, and `link`. [AERT, Technique 13.2.1] has been split up into three techniques.

Technique 13.2.2 The document must contain `meta` elements.

Category: Automatic.

Automatic check: FAILS unless the document contains `meta` elements for the `keywords` and `description` properties, and a `lang` attribute on at least one `meta` element.

Source: [AERT, Technique 13.2.1].

Modifications: [AERT] requires the presence of either one of `title, address, meta`, and `link`. [AERT, Technique 13.2.1] has been split up into three techniques.

Technique 13.2.3 The document must contain an `address` *or* a `link` element.

Category: Warning.

Automatic check: FAILS if the document contains no `address` element *and* no `link` element with a `type` attribute value other than `"stylesheet"`. SUCCEEDS otherwise.

Source: [AERT, Technique 13.2.1].

Modifications: [AERT] requires the presence of either one of `title, address, meta`, and `link`. [AERT, Technique 13.2.1] has been split up into three techniques.

Checkpoint 13.3

"Provide information about the general layout of a site (e.g., a site map or table of contents). [Priority 2]" [WCAG10]

Technique 13.3.1 Verify site layout information.

Category: Full Manual.

Manual check: Check the Web site and make sure it contains a page that describes the layout of the site, such as a site map or a table of contents. Make sure this page can be reached from the index page.

Source: [WCAG10].

Checkpoint 13.4

"Use navigation mechanisms in a consistent manner. [Priority 2]" [WCAG10]

Technique 13.4.1 Site layout and navigation mechanisms must be consistent between pages.

Category: Full Manual.

Manual check: Make sure the navigation structure, layout, position and functionality is consistent across several pages of the same Web site.

Source: [WCAG10].

Checkpoint 13.5

"Provide navigation bars to highlight and give access to the navigation mechanism. [Priority 3]" [WCAG10]

Technique 13.5.1 Navigation bars must be provided.

Category: Full Manual.

Manual check: Make sure the page provides navigation bars.

Source: [WCAG10].

Checkpoint 13.6

"Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group. [Priority 3]" [WCAG10]

Technique 13.6.1 Related links must be grouped together.

Category: Manual.

Manual check: Links must be are grouped into logical sets and marked up as a unit, and users must be allowed to bypass groups of links. The `div, span,` `p` or `map` elements must be used to mark up logical groups of links as a unit. Return all `a` elements for user inspection.

Source: [AERT, Technique 13.6.1].

Checkpoint 13.7

"If search functions are provided, enable different types of searches for different skill levels and preferences. [Priority 3]" [WCAG10]

Technique 13.7.1 Search mechanisms must satisfy a varying degree of skills and preferences.

Category: Full Manual.

Manual check: If a `form` element is used to submit a search, the search mechanisms must satisfy different skill levels and preferences. For instance, provide a keyword search as well as a tool that can handle complex boolean expressions, include a spell checker, offer "best guess" alternatives, similarity searches, etc

Source: [AERT, Technique 13.7.1].

Checkpoint 13.8

"Place distinguishing information at the beginning of headings, paragraphs, lists, etc. [Priority 3]" [WCAG10]

Technique 13.8.1 Distinguishing information must be placed at the beginning of headings, paragraphs, lists, etc.

Category: Full Manual.

Manual check: Make sure important information is placed at the beginning of headings, lists, and paragraphs.

Note: This technique is commonly referred to as "front-loading". It reduces the amount of information readers have to process in order to find important information.

Source: [WCAG10].

Checkpoint 13.9

"Provide information about document collections (i.e., documents comprising multiple pages.). [Priority 3]" [WCAG10]

Technique 13.9.1 Information about document collections must be provided.

Category: Full Manual.

Manual check: If the page is part of a collection, such as a slide show, or a chapter in a book, the navigation mechanisms must be marked up using the `link` element.

Source: [AERT, Technique 13.9.1].

Checkpoint 13.10

"Provide a means to skip over multi-line ASCII art. [Priority 3]" [WCAG10]

Technique 13.10.1 It must be possible to skip over multi-line ASCII art.

Category: Full Manual.

Manual check: Make sure that it is possible to skip over multi-line ASCII art.

Source: [WCAG10].

A.18. Guideline 14. Ensure that all documents are clear and simple.

"Ensure that documents are clear and simple so they may be more easily understood." [WCAG10]

Checkpoint 14.1

"Use the clearest and simplest language appropriate for a site's content. [Priority 1]" [WCAG10]

Technique 14.1.1 The used language must be the most simple and appropriate for the site.

Category: Full Manual.

Manual check: Make sure that the language of the document is appropriate to its content.

Source: [WCAG10].

Checkpoint 14.2

"Supplement text with graphic or auditory presentations where they will facilitate comprehension of the page. [Priority 3]" [WCAG10]

Technique 14.2.1 The page must not lack graphic or auditory presentation.

Category: Full Manual.

Manual check: Make sure page the document is comprehensible. If graphic and/or auditory presentation would substantially improve the comprehensibility of the page, the document fails this checkpoint.

Source: [WCAG10].

Checkpoint 14.3

"Create a style of presentation that is consistent across pages. [Priority 3]" [WCAG10]

Technique 14.3.1 The style of presentation must be consistent across documents.

Category: Full Manual.

Manual check: Make sure page layout is consistent, and that graphics are recognizable, allowing users to locate navigation mechanisms more easily.

Source: [WCAG10].

B. Results of Evaluation Tool review using ART

This appendix includes the detailed data of the Evaluation of our tool using the Access Tool Reviewer (ART) by Chris Ridpath. See Section 7.1 for an overview. *C.Id* is the number of the WCAG checkpoint being tested; *T.Id* is the Technique Id corresponding to our methodology (see Appendix A); *F.Id* is the file id of the ART test file.

C.Id	T.Id	F.Id	Description
1.1	1.1.1	1-1-1-f4	Not fully automatable.
		1-1-1-f5	Different interpretation: we do not allow empty (NULL) ALT text.
		1-1-1-f6	Different interpretation: we do not allow ALT text to consist only of white space.
	1.1.2	1-1-2-f1	Not fully automatable.
		1-1-2-f2	Not fully automatable.
	1.1.4	1-1-4-f3	Different interpretation: we require both ALT text and element content for applets.
	1.1.5	1-1-5-f3	Different interpretation: we require the alternative to be textual.
	1.1.6	1-1-6-f7	Not fully automatable.
	1.1.7	1-1-7-f2	Not fully automatable.
	1.1.8	1-1-8-f2	Different interpretation: we require frames to always provide a description using the longdesc element.
	1.1.12	1-1-12-f1	Not fully automatable.
1.2	1.2.1	1-2-1-f2	Not fully automatable.
2.2	2.2.1	2-2-1-f5	Not checked automatically due to its inherent complexity.
3.1	3.3.1	3-3-1-f5 3-3-1-f6 3-3-1-f7 3-3-1-f8 3-3-1-f9 3-3-1-f10 3-3-1-f11	Different interpretation: This techique always triggers if there are no style sheets in a document.
3.5	3.5.1	3-5-1-f5	Test file error. The first heading is a `h6`, however according to documents accompanying WCAG 1.0, headers should not "skip" levels.
3.6	3.6.2	3-6-1-f1 3-6-1-f2 3-6-1-f3	Not fully automatable.

Continued on next page ...

C.Id	T.Id	F.Id	Description
		3-6-1-f6	Different interpretation: We implemented this heuristic algorithm differently.
4.1	4.1.1	4-1-1-f1 4-1-1-f2 4-1-1-f3	Cannot be determined automatically, applies to all pages.
4.2	4.2.2	4-2-1-f3 4-2-1-f4	Test file error. The acronyms / abbreviations in the test file are expanded, but not properly marked up.
5.1	5.1.1	5-1-2-f5	Test file error. The table in the test file has complete row headers, so the check should *not* be triggered.
5.2	5.2.1	5-2-1-f3	Not fully automatable.
5.3	5.3.x	5-3-1-f2	Not fully automatable. Moreover, we believe tables need not be used for layout See Section A.4.
5.4	-	5-4-1-f1 5-4-2-f1	Not applicable; we believe tables need not be used for layout. See Section A.4.
6.2	6.2.1	6-2-1-f2 6-2-2-f3 6-2-2-f4 6-2-2-f5 6-2-2-f6 6-2-2-f7 6-2-2-f8 6-2-2-f9 6-2-2-f10 6-2-2-f11	Test suite error. Most tests fail because the files referenced by a frame do not exist. Moreover, the tests are designed to check for valid markup using the file extension; if it exists, using the HTTP Content-Type header is a better indication.
	6.2.2	6-2-2-f3	Test file error. The test file contains `<object></object>`, however not all objects represent dynamic content.
6.3	6.3.1	6-3-1-f2	
6.5	1.1.9	6-5-1-f3 6-5-1-f4 6-5-1-f5 6-5-1-f6	Test suite error. `noframes` elements are present in the test files, so this test should not be triggered. Whether the contents of the `noframes` section is appropriate cannot be determined automatically.
7.1	7.1.1	7-1-1-f2	Test file error. The test file contains `<object></object>`, however not all objects represent dynamic content.
7.3	7.3.1	7-3-2-f2	
7.5	7.5.1	7-5-1-f1	Test file error. The syntax for an URL redirect in the test file is incorrect.
8.1	8.1.1	8-1-1-f2	Test file error. The test file contains `<object></object>`, however not all objects represent dynamic content.

Continued on next page . . .

C.Id	T.Id	F.Id	Description
9.1	9.1.1	9-1-1-f2	Test file error. `usemap` does not designate a server-side image map.
10.2	10.2.1	10-2-1-f3	Not fully automatable.
12.3	12.3.1	12-3-1-f2	Cannot be determined automatically, applies to all pages.
13.1	13.1.4	13-1-1-f8	Not fully automatable.
13.2	13.2.*	13-2-1-f4 13-2-1-f5 13-2-1-f7	Different interpretation: we require `meta` elements regardless of `title` or `link`.
		13-2-1-f6	Different interpretation: we require an `address` *or* `link` element, regardless of whether `meta` is used.

C. W3C®Document License

`http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231`

Public documents on the W3C site are provided by the copyright holders under the following license. By using and/or copying this document, or the W3C document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to copy, and distribute the contents of this document, or the W3C document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the document, or portions thereof, that you use:

1. A link or URL to the original W3C document.
2. The pre-existing copyright notice of the original author, or if it doesn't exist, a notice (hypertext is preferred, but a textual representation is permitted) of the form: "Copyright ©[$date-of-document] World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. `http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231"`.
3. If it exists, the STATUS of the W3C document.

When space permits, inclusion of the full text of this NOTICE should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, if additional requirements (documented in the Copyright FAQ) are satisfied, the right to create modifications or derivatives is sometimes granted by the W3C to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

This formulation of W3C's notice and license became active on December 31 2002. This version removes the copyright ownership notice such that this license can be used with materials other than those owned by the W3C, moves information on style sheets, DTDs, and schemas to the Copyright FAQ, reflects that ERCIM is now a host of the W3C, includes references to this specific dated version of the license, and removes the ambiguous grant of "use". See the older formulation for the policy prior to this date. Please see our Copyright FAQ for common questions about using materials from our site, such as the translating or annotating specifications. Other questions about this notice can be directed to site-policy@w3.org.

Joseph Reagle <site-policy@w3.org>

Last revised $Id: copyright-documents-20021231.html,v 1.6 2004/07/06 16:02:49 slesch Exp $