Developing Accessible Web Components

Anu A. Gregory

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2019

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 24, 2019

Anu A. Gregory

Contents

Abstract

The world wide web as of today provides many opportunities for everyone, but there is still a lot of content available in the web which is not accessible to different groups of users. The web is quickly becoming a part of day to day life of people with disabilities. There are many types of assistive technologies available to help these people to understand and receive information from the web. Accessibility is also a legal requirement for many government websites around the world. Even though this is the case, there is still a large percentage of websites which do not follow accessibility guidelines. This can be due to different reasons, but it is the duty of the developer to implement the website in a way that it is accessible for all people. Nowadays the web pages are often built with custom web components which can be reused again in other projects. This master's thesis therefore tries to find a simple solution to create accessible web components.

Kurzfassung

Modernes Web bietet viele Möglichkeiten für alle, aber es gibt immer noch viele Inhalte im Web, die nicht für verschiedene Benutzergruppen verfügbar sind. Das Internet wird für Menschen mit Behinderungen immer mehr zum Alltag. Es gibt viele Arten von Hilfstechnologien, mit denen diese Personen Informationen aus dem Web verstehen und empfangen können. Zugänglichkeit ist auch eine Voraussetzung für viele Regierungswebsites auf der ganzen Welt. Auch wenn dies der Fall ist, gibt es immer noch einen großen Prozentsatz von Websites, die nicht den Richtlinien für Barrierefreiheit entsprechen. Dies kann verschiedene Gründe haben, aber es ist die Pflicht des Entwicklers, die Websites so zu implementieren, dass sie für alle Menschen zugänglich sind. Heutzutage werden Webseiten mit selbsterstellten, wiederverwendbaren Komponenten entwickelt, welche auch in zukünftigen Projekten eingesetzt werden können. In dieser Masterarbeit wird daher versucht, eine einfache Lösung zu finden, um die Webkomponenten barrierefrei in eine Webseite zu integrieren.

Chapter 1

Introduction

Web Accessibility is an important topic in modern day web development. The story of web accessibility is one of hard work, shared empathy and often a struggle to keep up. The success of web accessibility is mainly due to the dedication of law makers around the world. The web offers many opportunities to all kinds of people. It is the job of the developer to ensure that the required accessibility criteria is met and the end user can access the website without any barriers. There are guidelines available for the developers to follow and implement the accessibility features successfully. But most of the developers don't spend much time on web accessibility during implementation phase. The organization called World Wide Web Consortium (W3C)[1] develops international standards for web technologies and within the W3C there is another organization called Web Accessibility Initiative (WAI)[2], which develops the web accessibility standards. The guidelines provided by Web Accessibility Initiative (WAI) are called Web Content Accessibility Guidelines (WCAG)[3], an attempt that has guided accessibility efforts for the last 20 years.

In 2008, the World Wide Web Consortium (W3C) made an important update to their guidelines to make the internet more accessible for people with disabilities. But still, according to a report from 2018, less than 10% of sites are still accessible to everyone. Encouraging developers to follow the Web Content Accessibility Guidelines is a good place to start to improve the accessibility. This guarantees that the documents can be interpreted by different assistive technologies. These guidelines are well researched and provides the best way to improve the accessibility to provide the same information to all possible users. There are also example codes, that are available for developers, which show how to write the code to meet the requirements. These Web Content Accessibility Guidelines are updated from time to time when there are new technologies available. Putting these available guidelines to work and getting them to the end user is the job of the developer. But because of various reasons, most of the time this is not happening in this way. By looking into this and asking why this is so, is a good way to start tackling this problem.

The need for more accessible web pages is of utmost importance in the modern

[1] https://www.w3.org/

[2] https://www.w3.org/WAI/

[3] https://www.w3.org/WAI/standards-guidelines/#wcag

day web. There is a significant number of people with different types of disabilities. The web should be a place where there are no barriers to access information. All the people should be able to get the same information from the web. Even though there are standards and guidelines available, due to different reasons many developers don't bother implementing them in the development. Some talk about the workload, some about the time needed to implement and test the pages because of the deadline. So in this scenario the question was, what if there was a way so that the developer could implement the accessibility guidelines more efficiently and without much work.

The main goal was to find an easier way of approach to implement web accessibility on a web component. This is for the developers to implement the web accessibility in the base coding phase and test the component as early as possible. Once the component is tested to be accessible, then it can be reused as often as the developer wants without the hassle of checking for accessibility again. So this approach should be easy enough so that the developer can spend less time on the accessibility guidelines and still achieve better accessibility results. The research question is therefore the following:

> How to provide support for Web Accessibility in development using Web Components?

The implemented solution covers the necessary steps to make a component more accessible. By introducing this early into a project, the developer should notice significant reduction in time to implement accessibility guidelines. Furthermore, it should take a considerable amount of workload out of developers hands.

1.1 Structure

To express the considerations which led to the final solution, the following pages are structured into several chapters. The second chapter shows the background on which the research is built upon, the motivation, web content accessibility guidelines and future of accessibility guidelines. The third chapter introduces the state of the art evaluation tools for accessibility and comparison of different tools available. Chapter four gives an understanding about the approach behind the practical part of this thesis and describes the goals and requirements for a accessibility testing tool and a supporting programming library. Chapter five describes the full extent of the implementation of the project, the basic foundation and the main project setup. Chapter six informs about the evaluation of different components in the project, using online tools and the corresponding results are also provided. Chapter seven shows the conclusion of this thesis and gives a summery of the work done.

Chapter 2

Background

Practicing web accessibility helps to ensure that there are no barriers between people with disabilities and the world wide web. Websites can provide equal access to information and functionality if they are designed and developed correctly. The web is for everyone, not concerning their location, language, software, hardware or ability. Understanding the scope and impact of accessibility can improve the web development. At the end of the day, we can establish that people with disabilities can interact and contribute to the web. Web accessibility also helps people with temporary or conditional disabilities, which in some cases maybe aging, a broken arm, slow internet connection, etc. Currently many sites are developed with accessibility barriers that make them difficult for people to use. To understand this better, we need to understand the types of disabilities.

2.1 Definition and Categorization of Disabilities

According to Center for Disease Control and Prevention (CDC), "a disability is any condition of the body or mind (*impairment*) that makes it more difficult for the person with the condition to do certain activities (*activity limitation*) and interact with the world around them (*participation restrictions*)". Although the term "people with disabilities" sometimes refers to a single population, this is actually a diverse group of people with a wide range of needs. Same type of disability can be affect two people in very different ways. Some disabilities may be hidden or not easy to see [31]. There are three modes of disability [34]. The first one is the permanent disability, which is when a person has a disability for all their life. Example: blind, deaf or the like. The second type is the temporary disability, which is a mental or physical disability that interferes with the completion of ones duties for a short period of time. Example: broken limbs, hand injuries or the like. The third type of disability is the conditional or situational disability, which is simply when a person is not able to do things due to a current situation or condition. Example: slow internet connection.

2.2 Motivation for Web Accessibility

The internet is one of the best things that happened to people with disabilities. Before the internet came into existence the people with visual disabilities could not read newspapers themselves. Audiotapes or Braille printouts were expensive. At best, they could ask a friend or family member to read the newspaper to them. But this makes the blind people dependent upon others. Nowadays, most newspapers have their content online in a format that can be read by screen readers used by the blind people. These screen readers read online text out loud, so that blind people can understand the content of the page without any external help or they don't have to wait for expensive audio tapes and bulky Braille printouts. They can simply open a browser and listen as the screen reader reads the newspaper to them. They can do this independently and as soon as the content is published online. Similarly, people with motor disabilities who cannot pick up a newspaper or turn pages can now access online newspapers through their computer using assistive technologies. Sometimes these adaptations can be done easily, something like having the person place a stick in the mouth and use it to type keyboard commands. In other cases, the adaptations can be more sophisticated, such as in the case of using special keyboards or eye-tracking softwares that allow people to use a computer with nothing more than eye movements [51]. The Internet is an increasingly important aspect of daily life, including education, employment, government, commerce, health care, recreation, and many more sectors. So it is of utmost importance that the web is made accessible to everyone in order to provide equal opportunities to people with disabilities. By making the web more accessible, people with disabilities will be able to participate more actively in society. In the development of static and dynamic web pages, it is important to consider the intellectual disabilities throughout the entire design and implementation. The aim of the accessible web must be to give users the possibility to participate in the technological process [4]. There are several ways of adapting websites to users characteristics. Most of these are focused on a single user group. E.g. blind people, people with limited mobility or elderly people [9]. The web offers unprecedented access to information and interaction for many people. That is, the accessibility barriers to print, audio, and visual media can be overcome much more easily through web technologies [17]. Accessibility has a strong business case to it. In the beginning, the web was just a text medium. Then it started to be used as a business space, where its visual appearance, design and accessibility became important [2]. Accessibility and other best practices such as mobile web design, device independence, multi-modal interaction, usability, design for older users, and search engine optimization (SEO) overlap with each other. Accessible websites can have better search results, increased audience reach and they demonstrate corporate social responsibility (CSR) [34]. Governmental agencies have to care about accessibility because of legal aspects. Commercial businesses may be motivated by innovation and market expansion opportunities. Educational and nonprofit organisations may be especially drawn to brand enhancement. A common argument against accessibility is that the direct return on investment (ROI) is difficult to measure. Even though ROI is important, it is not by any means the only way to measure how an accessibility commitment profits organizations. It is most likely that the business will respond to a mix of driving factors as they consider implementing an integrated accessibility program. Businesses that integrate accessibility into their work-

ing modal are likely to be inventive, inclusive companies that meets emerging global legal requirements [26].

2.3 The Accessible Rich Internet Applications Specification

As the web started to get more diverse and complex, completely new sets of accessibility features and problems arose. Like the introduction of HTML5[1] brought a number of new semantic elements to define common page features (`<nav>, <footer>`, etc). Before these were available, programmers would simply use div elements with id or class attributes, e.g. `<div class="nav">`, but this was problematic, as there was no standard way to find a specific feature in a page like the main navigation. The initial solution was to add one or more hidden links at the top of the page to link to the navigation [37], for example: consider the code snippet `<a href="#hidden" class="hidden"> Skip to navigation </a>`. This code is still not completely precise, and can only be used when the screen reader starts reading from the top of a given page. When the apps started to feature more complex controls like date pickers for choosing dates, sliders for choosing values, HTML provided special input types to render such controls, like `<input type="date">` and `<input type="range">`. These elements are not that flexible on behalf of styling, making them not very useful for integrating with other website designs. As a result, developers quite often include JavaScript libraries that generate the required controls as a series of nested div elements or table elements with classnames, which are controlled using JavaScript and styled using CSS. Even though they work visually, screen readers cannot make any sense of what they are at all. The users will get told of this mixture of elements with no semantics, rather than with a description of what they actually mean. Here comes World Wide Web Consortium (W3C)'s Web Accessibility Initiative – Accessible Rich Internet Applications (WAI-ARIA)[2] into play. WAI-ARIA is an accessibility specification provided by the World Wide Web Consortium, defining a set of additional HTML attributes that can be added to HTML elements to provide additional semantics and improve accessibility wherever it is lacking so far.
There are three main features defined in the specifications [40]:

- *Roles*: These are the attributes added to HTML elements, which define what an element is or does. The main roles are called landmark roles, because they largely copy the semantic values of HTML5 elements. Example: navigation role used with `<nav>` HTML tag or the complementary role used with `<aside>` HTML tag. But there are also other roles in ARIA, that describes different page structures, such as the commonly used ones. Example of these are banner role, search role, tabgroup role, tab role etc.
- *Properties*: These define properties of HTML elements, which in turn can be used to give extra meaning or semantics to the elements. An example is aria-required attribute, which specifies that a form input field needs to be filled in order to be valid. Whereas the aria-labelledby attribute allows to give a specific ID to an element, then reference this as the label for anything else on the same page. This ID can be also used for multiple elements, which is not possible using the normal

[1] https://html.spec.whatwg.org/
[2] https://www.w3.org/TR/wai-aria/

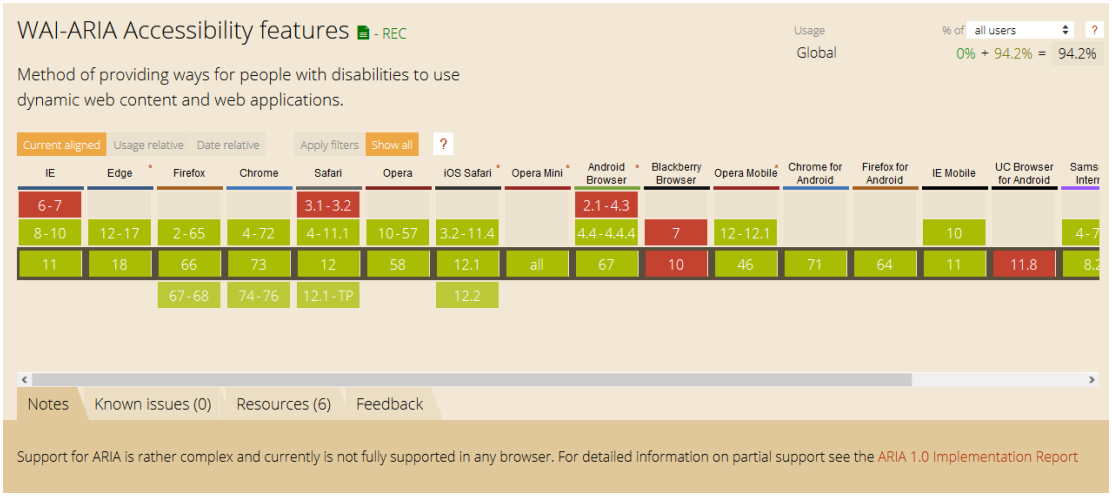


Figure 2.1: Current global browser support for different WAI - ARIA Accessibility features [39].

`<label for="input">`. As an example, aria-labelledby attribute can be used to specify that a key description contained in a `<div>` is the label for multiple table cells, or it can also be used as an alternative to image alt text, specifying existing information on the page as an image's alt text, rather than repeating it again in the image's alt attribute.

- *States*: These are special properties that define the current conditions of an element, like `aria-disabled="true"`, which specifies that the form input is currently disabled. States are different properties in a way that properties don't change throughout the lifecycle of a web application, whereas states can change throughout the lifecycle, mainly via JavaScript.

ARIA attributes don't affect anything other than the data exposed by the browser's accessibility APIs (where screen readers get their data from). It does not affect the structure of the web page, the DOM, etc., although the attributes can be used for selecting elements by CSS. It is difficult to find a decisive source that states which features of WAI-ARIA are supported and where, because there are a lot of features in the WAI-ARIA specifications and there are too many combinations of operating systems, browsers, and screen readers to examine. For example, to use a screen reader in the first place, the operating system needs to run browsers that have the required accessibility APIs to retrieve the information that screen readers need to work on. Most popular operating systems have one or two browsers that screen readers can work with. The next step is to check whether the browsers in use support ARIA features and expose them via their APIs and also if the screen readers recognise this information and present it to their users in an appropriate way. The global browser support for WAI-ARIA is around 94% (see Fig. 2.1). But Screen reader support for WAI-ARIA isn't quite at this level (see Fig. 2.2), but the most popular screen readers are improving the support for ARIA. There is also an increase in reliability of ARIA in different assistive technologies (see Fig. 2.3) over time. There are four main areas where WAI-ARIA is used:

Combo	Versions	Reliability
JAWS IE	JAWS 17.0.2619 with IE11	79%
JAWS Firefox	JAWS 17.0.2619 with FF48	71%
NVDA IE	NVDA 2017.3 with IE11	64%
NVDA Firefox	NVDA 2017.3 with FF60	76%
VoiceOver Mac	VoiceOver OSX 10.12 with Safari 10.1.2	69%
VoiceOver iOS	VoiceOver iOS 10.3 with Safari iOS 10.3	55%
WindowEyes IE	WindowEyes 9.2 with IE11	74%
Dolphin IE	Dolphin SR 15.05 with IE11	52%
SaToGo IE	SaToGo 3.4.96.0 with IE11	25%
Average	Including older versions	65%

Figure 2.2: ARIA role and attribute support in different screen reader / browser combinations. The solid area shows the percentage of tests that pass in all tested interaction modes. The cross hatched area shows partial passes that only work in some interaction modes. An example of a partial pass is when form labels are read when tabbing [41].

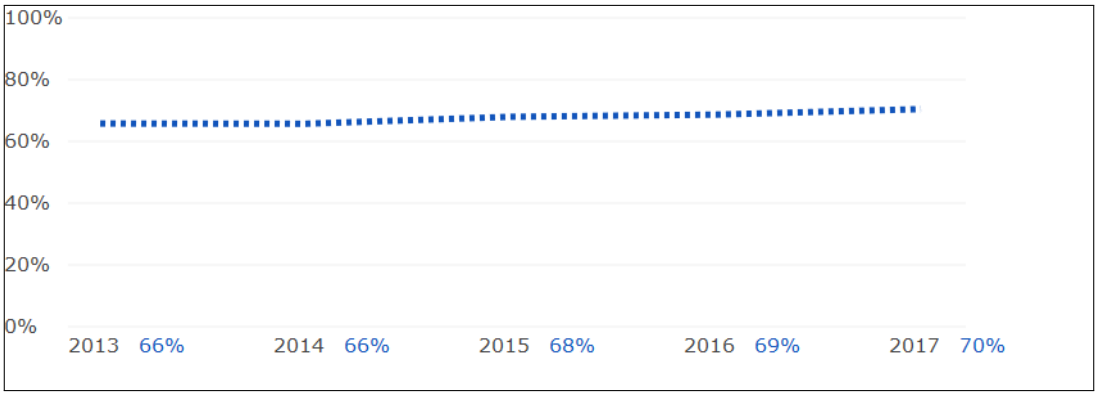


Figure 2.3: Reliability improvement over time for ARIA in NVDA, JAWS, WindowEyes and Voiceover [41].

Signposts or Landmarks: ARIA's role attribute values can be used as landmarks that either duplicate the semantics of HTML5 elements (e.g. `<nav>`), or go beyond that and provide signposts to different functional areas, e.g. search role, tabgroup role, tab role, listbox role, etc. The working of roles is fairly simple, WAI-ARIA adds the role attribute to the browser, which in turn helps to add extra semantic value to elements wherever they are needed. The major area where this is used is for providing information to screen readers, so that users can find common page elements.

```
<header>
  <h1>...</h1>
  <nav>
    <ul>...</ul>
    <form>
      <!-- search form -->
    </form>
  </nav>
</header>

<main>
  <article>...</article>
  <aside>...</aside>
</main>
<footer>...</footer>
```

Program 2.1: A basic website structure [42].

Figure 2.4: An example VoiceOver landmark menu [42].

When the HTML page listed in Prog. 2.1 is tested with a screen reader in a browser, the following information can be acquired:

- `<header>` element contains a heading and the `<nav>`,
- The `<nav>` element contains a list and a form,
- The `<main>` element contains an article and an aside,
- The `<aside>` element contains a heading and a list,
- The search form input, Search query, at beginning of text,
- The `<footer>` element has a footer item.

In the VoiceOver's landmarks menu (see Fig. 2.4), most of the landmark elements are listed so they can be accessed quickly. However, this can be improved. The search form is a really important landmark that people will want information about, but it is not listed in the landmarks menu or treated like an important landmark, other than the actual input being called as a search input (`<input type="search">`). Also, some older

```
<header>
  <h1>...</h1>
  <nav role="navigation">
    <ul>...</ul>
    <form role="search">
      <input type="search" name="q" placeholder="Search query"
      aria-label="Search through site content">
    </form>
  </nav>
</header>

<main>
  <article role="article">...</article>
  <aside role="complementary">...</aside>
</main>

<footer>...</footer>
```

Program 2.2: Addition of ARIA roles to elements in Prog. 2.1 [42].

browsers don't recognise the semantics of HTML5 elements. This can be improved by adding role attributes to the HTML structure.

In the VoiceOver menu, there will be some improvements. The search form will be recognized as a separate item, when browsing through the page and also in the landmarks menu. The label in the aria-label attribute is read out when the form input is focused. Now, the website is more likely to be accessible on older browsers. And if the website is built using just `<div>` tags for some reason, including the ARIA roles will provide much needed semantics. The improved semantics of the search form shows the possibilities when ARIA goes beyond the semantics available in HTML5.

Dynamic content changes: Screen readers tend to have difficulties with constantly changing content, but with ARIA, the aria-live attribute can be used to inform screen reader users that the content is updated. e.g. via XMLHttpRequest or DOM APIs. Screen readers can easily access a content loaded into the DOM, from textual content to alternative text or images. This makes it easier for the traditional static websites with mostly text content to make them accessible for people with visual impairments. But modern web apps are often not just static text, they have a lot of dynamically updating content, i.e. content that updates without a page reload using a mechanism like XMLHttpRequest, Fetch or DOM APIs.

Consider Prog. 2.3, JavaScript loads a JSON file via XMLHttpRequest, which contains a series of random quotes, then a `setInterval()` loop is introduced, which loads a new random quote into the quote box every 10 seconds. The JavaScript function used to set interval is `window.setInterval(showQuote, 10000)` [38].

This will work without any problem, but this is not good for accessibility, the content update will not be detected by screen readers. This is a simple example, but if there is a much more complex UI with lots of constantly updating content, e.g. a chat room, it would be impossible to use the web app in any effective way without alerting the user

```
<section>
  <h1>Random quote</h1>
    <blockquote>
      <p></p>
    </blockquote>
</section>
```

Program 2.3: simple random quote box.

about the updates. WAI-ARIA fortunately provides means to give these alerts using the aria-live property. Applying this property to an element makes the screen readers to read out the updated content. The attribute value determines how fast the content is read out [42]. For example, using `<section aria-live = "assertive">` will cause a screen reader to announce the updated content immediately after it is updated. The available attribute values are:

- `off`: default value, updates will not be read out,
- `polite`: updates should be read out, only when the user is idle,
- `assertive`: updates should be read out as soon as possible.

Keyboard Accessibility Enhancement: There are built-in HTML elements which have native keyboard accessibility. When other elements are used with JavaScript to simulate identical interactions, the performance of keyboard accessibility and screen reader reporting may drop. Where this is unavoidable, WAI-ARIA helps to allow other elements to receive focus (using tabindex attribute). This is one of the key strengths of HTML5 with respect to accessibility. Tab key can be used to move between controls and the Enter key to select or activate controls. Sometimes there may be a need to write a code that uses non-semantic elements as buttons or focusable controls, in order to fix a bad code or to build a complex widget or to make a non-focusable code focusable, WAI-ARIA extends the tabindex attribute with new values for these purposes.

- `tabindex="0"`: Using this value helps elements that are normally not tabbable to become tabbable. This is the most important value of tabindex,
- `tabindex="-1"`: Using this, the elements which are not normally tabbable can receive focus via JavaScript.

Accessibility for non-semantic controls: When a series of nested `<div>` tags are used along with CSS or JavaScript for creating a complex UI-feature or a native control is largely changed using JavaScript, accessibility may drop because of this. Screen reader users may find it hard to understand what the feature does, if there are no semantics or other details. In these types of situations, WAI-ARIA can be used to provide the missing piece with a combination of roles like button role, listbox role, or tabgroup role, together with properties like aria-required or aria-posinset to provide further details of functionality.

Ideally, it is recommended to use native HTML features to provide the semantics required by screen readers to tell their users what is going on. But sometimes this isn't

possible, either due to limited control over the code or because of the complex nature of the code, which does not have an easy HTML element to implement it. In these types of cases, WAI-ARIA can be used to enhance accessibility [38].

Web accessibility is required by law in many circumstances [17]. Web Accessibility Policy Resources link to available resources within organizations for addressing legal factors including relevant laws and policies from all around the world [34].

2.3.1 Web Content Accessibility Guidelines (WCAG) 2.1

Web Content Accessibility Guidelines (WCAG) 2.1 explain how to make web content more accessible to people with disabilities [48]. Accessibility covers a wide range of disabilities, including visual, auditory, physical, speech and neurological disabilities. Although using these guidelines, a wide range of issues can be covered, but these guidelines are not able to address every need of people with all types of disabilities. These guidelines are also there to make the web content more usable for older individuals with changing abilities and often improve usability for more users in general. WCAG 2.1 is the latest version of guidelines, which was developed through the W3C process in cooperation with individuals and organizations around the world, with a goal of providing shared standard for web accessibility around all platforms. WCAG 2.1 is designed to apply to all types of web technologies now and in the future and to be tested with a combination of automated and human evaluations. There were many challenges in defining additional criteria to address language, cognitive and learning disabilities. Web accessibility depends on accessible content, accessible web browsers and other user agents. Authoring tools play an important role in web accessibility.

2.3.2 Web Content Accessibility Layers of Guidance

The type of individuals and organizations that use WCAG vary widely and are consisting of web developers, policy makers, educational organizations and web designers. To meet these vastly different needs of the targeted audience, several different layers of guidance are provided, which include the overall principles, guidelines, testable success criteria and sufficient techniques, documentation, resources, codes etc. [49].

- *Principles*: The four principles that are the foundation of web accessibility are perceivable, operable, understandable, and robust.
- *Guidelines*: Under the principles there are 13 guidelines. These 13 guidelines set the basic goals that the developers should achieve in order to produce web pages that are more accessible to users with disabilities. These guidelines are not testable, but the web developers are provided with the framework and objectives to help to understand the success criteria and better techniques for implementation.
- *Success Criteria*: Each guideline is provided with testable success criteria to allow WCAG to be used where requirements and conformance testing are necessary. Three levels of conformance are defined to meet the needs of different groups and situations. They are: A (lowest), AA (Medium), and AAA (highest).
- *Sufficient and Advisory Techniques*: For each guideline and success criteria in the WCAG document, there is also a wide variety of techniques documented. The techniques fall into two categories: those are sufficient to meet the success criteria

and those which are advisory to meet the success criteria. The advisory techniques go beyond the requirement of the individual success criteria and allow developers to address the guidelines better. Some advisory techniques check for accessibility barriers that are not even covered by testable success criteria.

Together these layers (principles, guidelines, success criteria, and sufficient and advisory techniques) give guidance about how to make the web more accessible to users with disabilities. Web developers are encouraged to implement all the layers they are able to, including the advisory techniques in order to provide solutions to a wide range of users rather than a specific targeted audience. There is a set of requirements that is set for WCAG 2.1, which inherits requirements from WCAG 2.0. The overall framework of guidelines and backwards compatibility depends upon these requirements. The set of acceptance criteria used was less formal to ensure success criteria are similar in quality and style as in WCAG 2.0 [50].

2.3.3 Accessibility Guidelines Future

The Accessibility Guidelines Group is working towards another version of accessibility guidelines. The expected result is a more substantial restructuring of accessibility guidelines. There is much work focused on research and user centered design methods to produce a more effective and flexible solution including user agent support and authoring tool support. This will be in the long run, that is the reason that WCAG 2.1 was needed as an interim solution to provide the much needed update for the web accessibility guidance to keep up with the development in the web since WCAG 2.0. There are also possibilities for additional interim guideline versions, continuing with WCAG 2.2, while the major version is being completed [43].

Chapter 3

State of the Art

As the need for better web accessibility is increasing, there are many new pieces of research happening in this area. There are many new technologies which can help to increase the quality of life of elderly people and people with disabilities. Despite the availability of all these new technologies there are still accessibility barriers people experience when using different Information and Telecommunications Technology (ICT) solutions [3]. The use of Information and Telecommunications Technology in public sector is not an exception. eGovernment is an example of the transforming power of Information and Telecommunications Technology. eGovernment is a governmental service, which is aimed at providing information and services to citizens, employees and other government entities [1]. The government websites have better accessibility than many commercial websites. For example, Govt of India has a specific set of guidelines for websites belonging to any constitution of the Govt. These guidelines, which are called Guidelines for Indian Government Websites (GIGW), was introduced to ensure that the websites user friendly, accessible, secure and easy to maintain [6]. HTML has been designed to make websites more accessible, also for people with disabilities. However, website developers most of the time do not use these functionalities available to them. Web developers should make the websites accessible to all people regardless of physical barriers. Studies have shown that many websites are still not accessible to people with disabilities. Usually developers use different methods to check for accessibility, manual or automatic evaluations are performed. When manual evaluation is done, there is a chance that the developer may have some potential bias. In automatic evaluation, the web page is checked for compliance with the guidelines provided using a software. The use of software for testing minimizes the need for human intervention. Also, automatic testing is scalable and objective. But automatic evaluation has some limitations, as it cannot go to the same depth as manual evaluation nor can it be as complete. So in most of the cases both evaluations are used to get better results. Many researches in the field of web accessibility are also about how a better result can be given to the developer, which helps making the web page better.

As a response to the increasing demand for inclusive web, several web accessibility guidelines were developed. E.g. American Section 508 [20], the German BITV [24] and the Web Accessibility Initiative (WAI) [47]. The WAI was developed by the Word Wide Web Consortium (W3C). WAI also develops guidelines for accessibility in mobile devices. Mobile accessibility helps to check the accessibility of websites when they are

used in a mobile phone. The objective of WAI is to develop strategies, guidelines and resources, which help to make the web more accessible to everyone. WAI published several guidelines known as the Web Content Accessibility Guidelines (WCAG) and the latest version is WCAG 2.1, which was published on June 2018 [45].

3.1 Web Accessibility Evaluation Tools

The web accessibility evaluation is simplified by the use of software programs and online services that helps to have a better understanding about if the web content meets the accessibility criteria. Accessibility evaluation helps to identify potential accessibility issues quickly and are used in all phases of the web development process. The accessibility tools available can do fully automated checks and then the developer can do the review. Even though this is the case, all aspects of accessibility criteria cannot be checked automatically. User reviews are also required. There is also the possibility of evaluation tools producing false or misleading results. Accessibility evaluation tools cannot determine accessibility themselves, they just assist the developer in doing that. These evaluation tools target different type of users, including designers, developers, testers and also end users. Different web accessibility tools in the market provides different functionality and features, which targets different type of end users and this helps the users to compare the tools available and make a decision on which one is needed for a specific task [44].

3.1.1 Standards and Guidelines

To understand accessibility tools, is to understand the standards in which the tool was developed. Before selecting a specific accessibility evaluation tool, it is important to take into account the standards and guidelines used by these different accessibility evaluation tools. The most commonly used two standards are, WCAG 2.0 and Section 508 of the united states Rehabilitation Act [33]. Another very important aspect to the accessibility evaluation tool is the budget. For many budget conscious web designers or developers this is very important. There are many good free accessibility evaluation tools available in the market, but after checking the requirements and weighing in the specific needs of the developer, it may be the best to choose a commercially available tool. These decisions can be based on these following criteria [30]:

- *Who will be using the tool*: This is considered the main aspect before choosing an evaluation tool, that the developer has the necessary knowledge to use it. Free evaluation tools available usually assume the user has a greater understanding and spends less time in giving tutorials to users.
- *The size of the site being examined*: If the content to be checked is very large, it is important that the tool being used will look through the entire site so the developer doesn't have to check the site one page at a time. Most of the time only the commercially available evaluation tools have this feature. Free tools usually check one page at a time.
- *The information that must be collected*: Commercially available evaluation tools often give more detailed and specific reports.

3.1.2 Functioning of Accessibility Tools

Another important classification of accessibility evaluation tools involves details about where these tools are meant to function. Some tools are available in websites, where it is possible to evaluate the content of a page quickly and easily without downloading or installing a software. Some evaluation tools can be integrated to the browser as an extension and finally there are tools which are required to be installed like a normal software [10, 32].

- *Online services*: There are many free accessibility evaluation tools that are available online. They work when the user inputs the URL of a web page which needs to be checked, and then selecting from the evaluation guidelines available to check the page with, and then finally choosing the method provided to initialize the program, to start the checking process.
- *In a browser*: There are many accessibility evaluation tools that has been created as extensions of Internet browsers. After installation in the browser, they provide an additional menu in the browser. This menu can be accessed to know the different accessibility checking functions available in the extension. Then the user can use these functions to do the different tasks to evaluate the page that is currently active in the browser window. These extensions are usually single page evaluation tools.
- *Using an authoring tool*: Some evaluation tools can be used as a part of web authoring tools like Adobe Dreamweaver. These plugins and extensions will help developers to check the page content in the same environment where the content is created.
- *Software installed on the hard drive*: Many of the high-end evaluation tools need to be installed on the host machine like normal softwares. These type of accessibility evaluation tools are useful when the developer needs to check large and complex websites.

3.1.3 Scope of Evaluation Tools

Mainly there are two major categories where the accessibility evaluations tools are classified into based on what they examine and the scope of the evaluation tool. Some evaluation tools, mainly the online tools have limited scope, as most of them can only be used to evaluate just a few things and also only one page at a time. There are other tools available, which focuses on only one element of a website for evaluation. Then there are more detail oriented evaluation tools which examines large websites and check for a wide variety of issues [36].

- *Single page at a time*: Accessibility evaluation tools that evaluate only one page at a time, these are the ones which usually found online and used as part of browsers. The functionality of these tools are limited. They are used to evaluate a page which is found on a single URL. Even though this is the case, normally they are helpful in providing many details about the accessibility of the page.
- *Accessibility of specific items*: There are many accessibility evaluation tools that focus on just a single page element on a web site. Some of these tools may have options to show the content from the perspective of someone with visual disabilities.

These limited scoped evaluation tools are commonly found online or as browser extensions.

- *Accessibility of the whole website*: Accessibility evaluation tools that can be installed as a software are often helpful to inspect a large website for a wide range of errors. These tools are mainly used by large organizations, who has more specific requirements and a large number of developers. This feature is available in paid evaluation tools.

3.1.4 Evaluation and Reports

Another classification of evaluation tools are based on the availability of repair functionality in these tools. Many tools in the market can only do the evaluation, but there are some tools which after checking for accessibility, offers the functionality to guide the repairing process. These functionality is mostly seen on commercial evaluation tools and these tools gives feedback to the users and help them understand what is needed to be done to improve the accessibility [12]. Accessibility tools generate different type of reports based on the analyses performed on the web page. Good evaluation tools checks for obvious errors (errors due to not following guidelines) and alerts (these errors the developer need to check manually). Beyond this general functionality of the evaluation tools, the report styles may vary based on the target audience and chosen accessibility standards. Web developers should know which type of reports are expected from the accessibility tools before an evaluation tool is selected [13].

- *Text Based Report*: Most common type of accessibility evaluation reports are text based reports, which consists the guidelines used to scan the web page and the instances where accessibility errors occurred.
- *Graphic Reports*: These type of reports use graphics like icons to specify the accessibility errors on the page. In this type of report, there will be specific icons for specific errors in the page and in the report these icons will appear next to the item with accessibility errors.
- *Evaluation and Reporting Language (EARL) Report*: EARL reports are result of an attempt from W3C to standardize accessibility evaluation reports. These are machine readable reports of the accessibility guidelines the page validates to. These reports highlights the accessibility issues, the date of validation and the manual checks to be performed. EARL reports is helpful to users to compare the effectiveness of different accessibility tools.

3.2 Different Evaluation Tools

There are many different types of evaluation tools available in the market with different options to choose from. The developer can choose any of these tools based on requirements for a specific project and the different criteria mentioned before. Developers from different countries may also choose an accessibility tool based on the requirements of the governmental law in that country. Four different types of evaluation tools are discussed here, to completely understand how varied these tools can be.

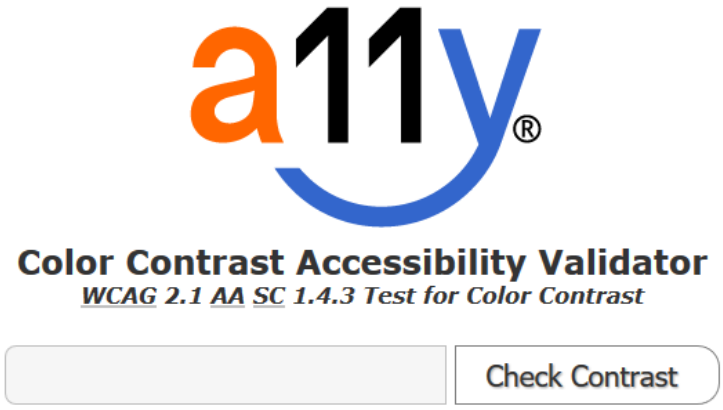


Figure 3.1: Color Contrast Accessibility Validator [29].

3.2.1 A11Y Color Contrast Accessibility Validator

The contrast between text and its background color is an important concern for visually impaired users. Color Contrast Accessibility Validator[1] is a compliance tool, which helps the developer to check for color contrast issues on a web page. It validates the web page according to the Web Content Accessibility Guidelines 2.1 [46]. Color contrast, indicate how the bright or dark colors on a web page appear against each other on screen in regard to relative gray-scale luminosity. The result of this test will show the color combinations that failed the contrast checkpoints. A contrast ratio of 4.5:1 is the minimum requirement for texts and images embedded as text, against the background color of the text. If the text is of size 18 point+ or 14pt+ bold, then it requires a minimum contrast ratio of 3:1 with the text background color. As this is an automatic check tool, it cannot analyze text embedded in images. So there is a chance for misdiagnose or ignore critical issues [29]. Color Contrast Accessibility Validator is an online checking tool and checks a single web page at a time. The URL to the page that need to be checked is entered in the text field. When the Check Contrast button is pressed it will check for the contrast issues in the page and then provides the developer with analysis report of the page and also provides a guide to overcome the errors, if there is any errors present.

The Color Contrast Accessibility Validator has a simple user interface (see Fig. 3.1). The user can give the address of the page that needs to be checked and press the check contrast button to get a feedback. The report (see Fig. 3.2) includes the text colour used, the background color of the text, text font, content of the text and the reason for failure in compliance with WCAG 2.1 guidelines, if there is any failures.

3.2.2 Accessible Brand Colors

Accessible Brand Colors[2] is a tool which provides information about how compliant the colors are in relation to each other. It was made to check the compliance of the

[1] https://color.a11y.com/?wc3
[2] https://abc.useallfive.com/

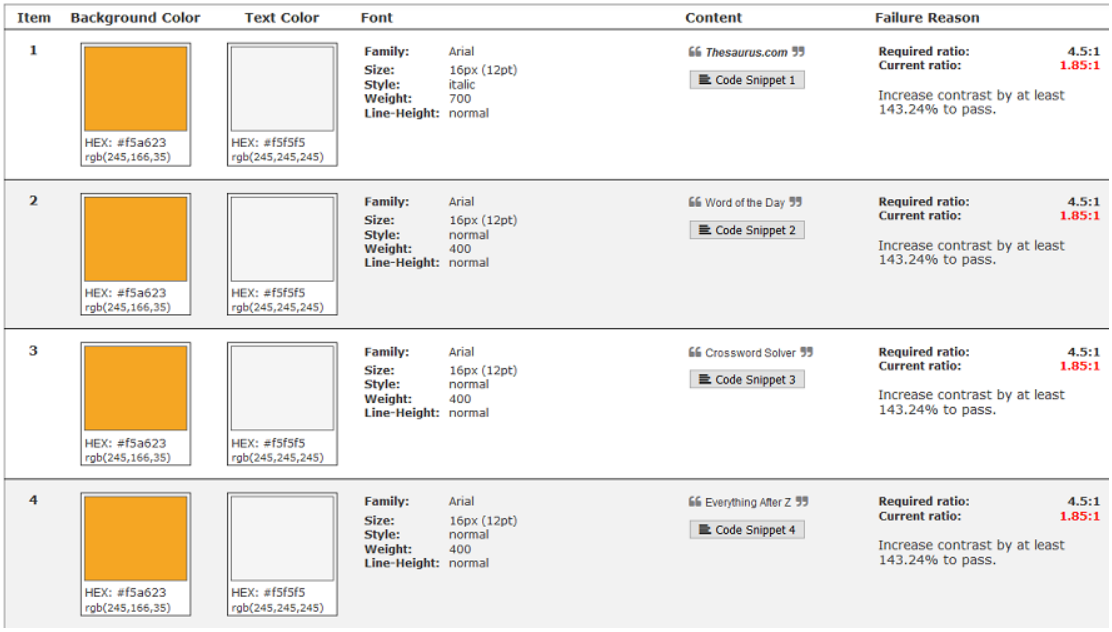


Figure 3.2: Example Color Contrast Accessibility Validator Report.

colors with American Disability Act (ADA). The tool is quite simple to use and the result will show the compliance level (see Fig. 3.3) and gives marks for different levels of compliance level [18].

There is also option to add a third color to this and check for compliance with the first selected two colors. The result page shows the compliance level grades using 4 different levels of compliance. Fig. 3.4 shows these 4 different levels and using that the verification can be done to see if the selected colours have passed the test. Here it is not the case, as the the result shows DNP, which means it did not pass the test.

3.2.3 Accessibility Insights for Web

Accessibility Insights for web[3] is a chrome browser extension menu(see Fig. 3.5) that helps to find and fix accessibility issues in web apps and websites. This tools offers support mainly in two scenarios, first is to help developers identify common high impact accessibility issues. This is done as a two step process called *FastPass* (see Fig. 3.6). The first process is automated checks on the page, where the tool checks for compliance with accessibility requirements. The second step is called Tab stops. In this step the tool provide measures to deal with keyboard accessibility. Instructions and a visual helper is provided, making it easier to identify critical accessibility issues. E.g. keyboard traps, tab stops, incorrect tab order [14]. The next scenario is *Assessment*. In this case the tool helps anyone with basic HTML skills to verify if the web app or website is 100% compliant. Automated and Manuel checks are available to do the check. In automated checks, the tool takes care of everything for the developer. In manual check the tool provides instructions, so the developer can identify and evaluate specific instances [22].

[3] https://accessibilityinsights.io/docs/en/web/overview

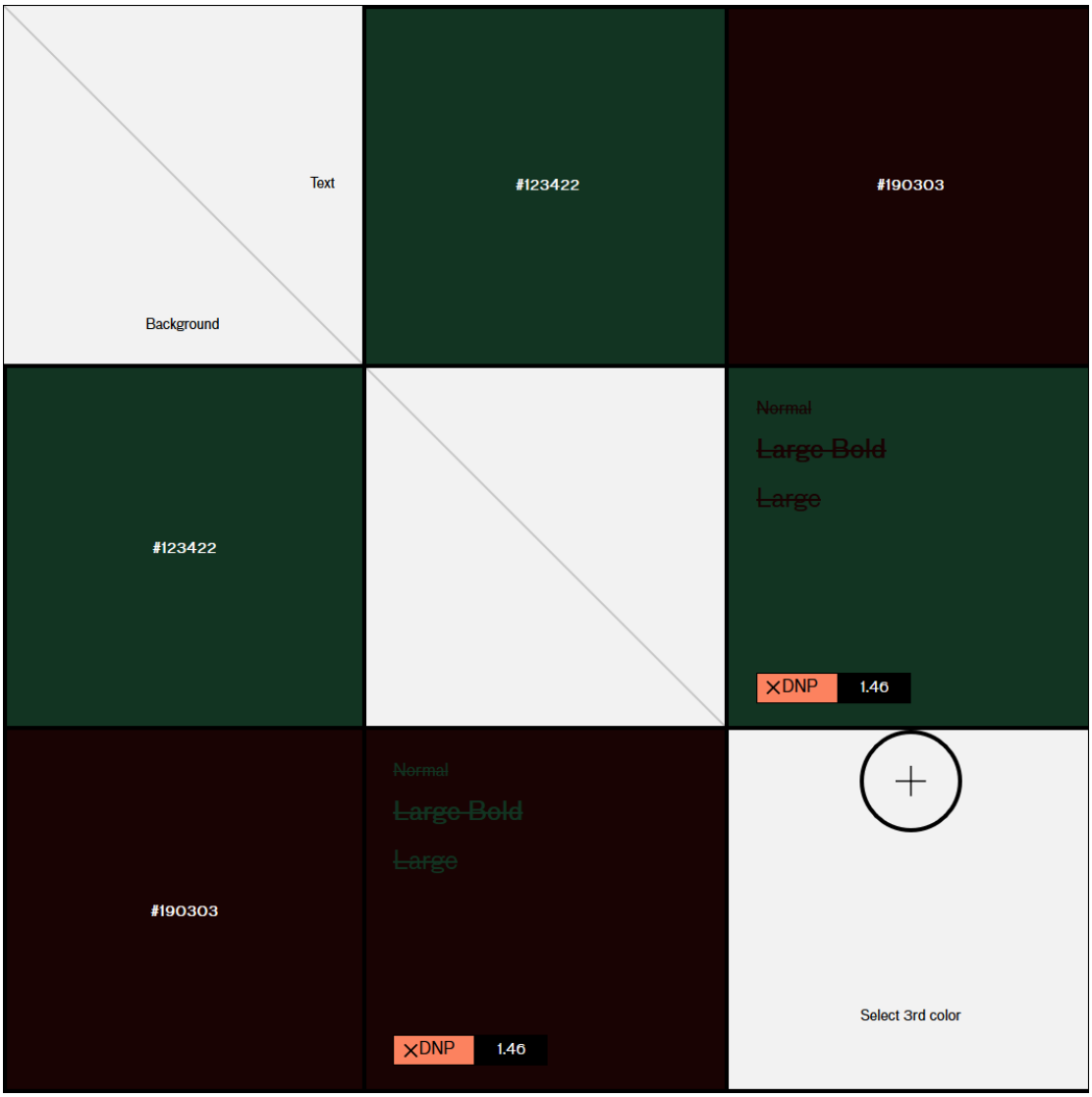


Figure 3.3: Color test results. Here the first color selected is Bush(#123422) and the second color selected is Creole(#190303) [19].

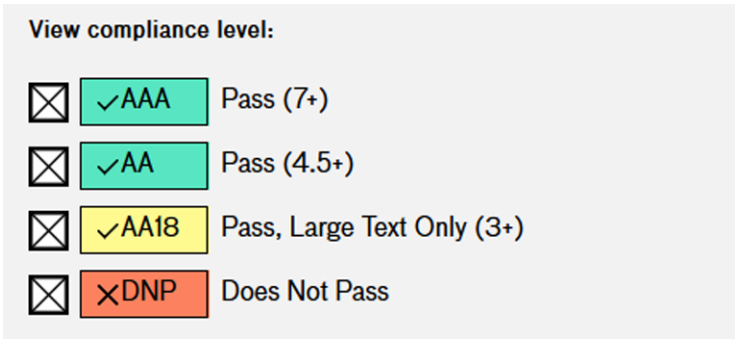


Figure 3.4: Different accessibility compliance levels [19].

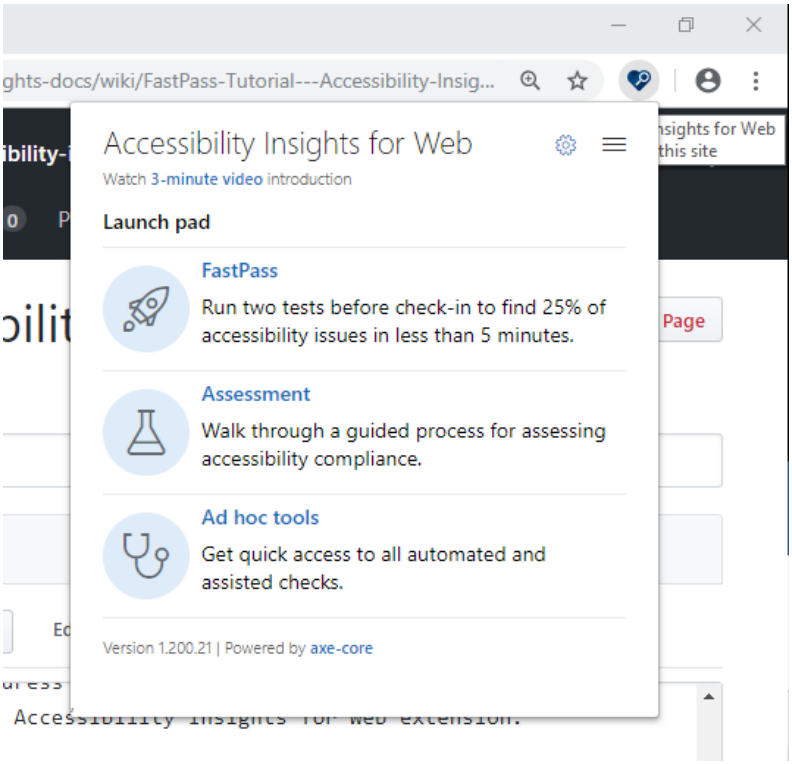


Figure 3.5: Accessibility Insights for Web, Browser Extension[15].

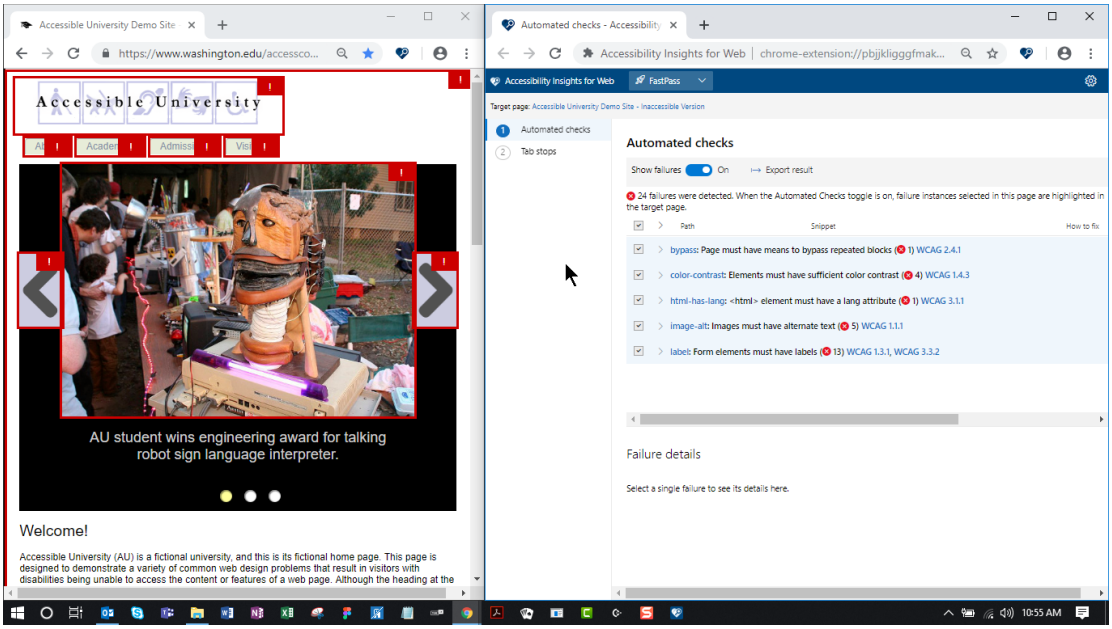


Figure 3.6: Automated Checks - FastPass [16].

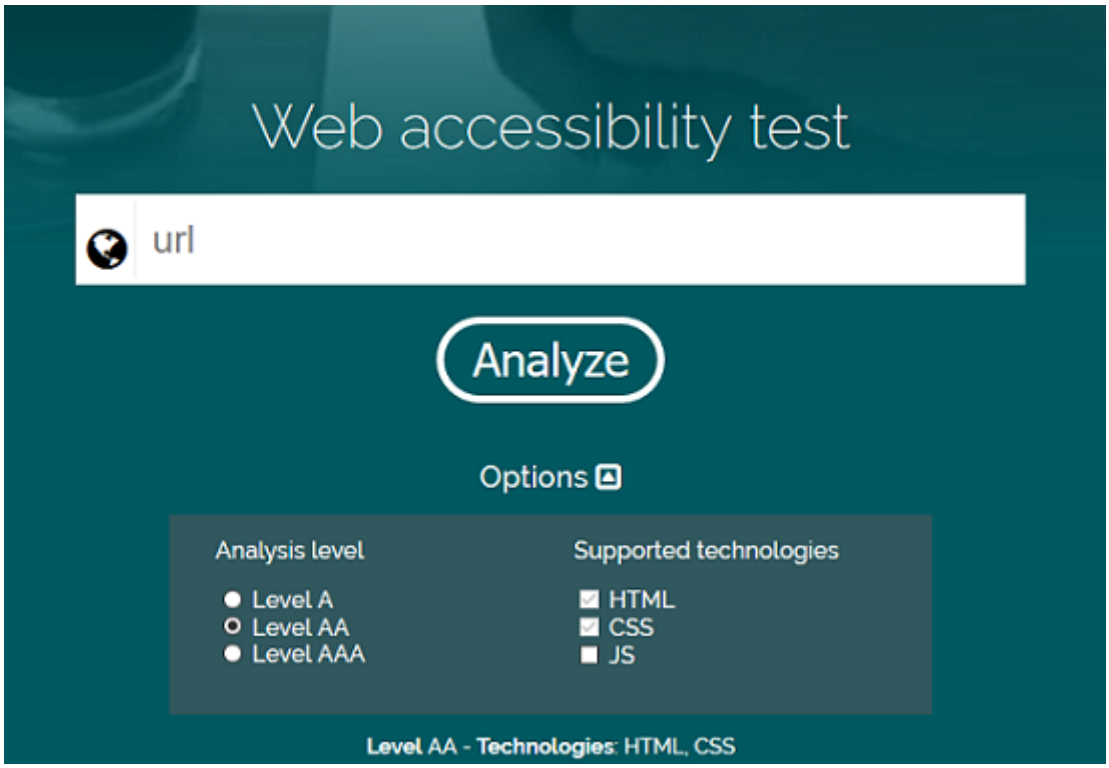


Figure 3.7: TAW Accessibility Analysis Tool.

Accessibility Insights generates reports of evaluation results and provides step-by-step evaluation guidance. As it is a browser plugin, it is used to check a single web page rather than an in-depth evaluation of the complete website. The failed instances will be highlighted in the page itself. In addition, a secondary page will open with details about the failed rules and instances where the failure occurred (see Fig. 3.6). When the assessment is finished, it is possible to review the failures in the target page itself by selecting a failure icon or the review can be done in a secondary page which opens up with the failed instances. These failure details include a statement of the violated rule, the name and link to the details about the rule, path to the element that failed the test and also the instructions for fixing the failure. When the failure details is reviewed in the target page, the element's code can be seen using the inspect HTML button option available in the developer tools. In the secondary page the failed element's code snippet is already provided. There is also option available to copy the failure details into other document or bug report for future references.

3.2.4 TAW

TAW[4] is an accessibility tool developed by the Spanish Foundation Center for the Development of Information and Communication Technologies in Asturias (CTIC)[5]. It

[4] https://www.tawdis.net/index
[5] https://www.fundacionctic.org/

has adaptations in English, Spanish and Portuguese [3]. TAW tests for accessibility using web content accessibility guidelines 2.0. The user interface of TAW is simple (see Fig. 3.7). There is option to input the url to check and then there is additional options which can be selected. These additional options include options to select the analysis level and supported technologies. After the required options are selected, press the analyse button to start the analysis of the page. The result of the analysis is shown in next page and this page includes the summary of the test. Summary of the analysis include problems and warnings found on the page. There is also detailed information available about the contents on the page which was not perceivable, operable, understandable and robust [5]. TAW also has the option to send the detailed analysis report to an user.

3.3 Accessibility Evaluation Tools Comparison

Developers prefer to use automated testing tools to check accessibility. However, some of these tools may have serious limitations and it is important to have a better understanding about what a tool can achieve before selecting a tool. Many accessibility testing tools of poor quality get more attention sometimes, than the good ones. The Automated WCAG Monitoring community group[13] is a W3C community which stands to improve the rules for web content accessibility guidelines testing. The purpose of this is to help web developers with better accuracy and completeness of their tools [8].

From Table 3.1, it is evident that there are many accessibility testing tools available in the market as a free software or commercial or even as a browser plugin, which follows different accessibility guidelines. It is the task of the developer to choose one tool which follows a specific guideline which is required for the task in hand. Most of the free softwares check a single web page at a time. The paid commercial accessibility testing tools like Access Alchemy can be used to check a number of pages and also websites at a time and these kind of paid softwares also help the developer by immediately deploying automated fixes for the most obvious accessibility violations. 508checker is an online accessibility checking tool which specifically check if, the web page is in compliance with American Rehabilitation Act section 508. This is a free tool, but can be only used to check single web pages.

As mentioned before, Accessibility Insights for web can be installed as an add-on on Google Chrome or Microsoft Edge web browser to check the accessibility. There is even an option for assessment of all the rules one by one in this browser plugin. A11Y Color Contrast Validator and Accessible Brand Colors are two free tools in the list, which are mainly used to check for color accessibility criteria on web pages. These tools help developers to focus the accessibility checking mainly on the background and foreground color combinations. Both of these tools can be used to check web pages which

[6] http://www.508checker.com
[7] https://accessibilityinsights.io/docs/en/web/overview
[8] https://color.a11y.com/?wc3
[9] https://www.levelaccess.com/solutions/software/access-alchemy/
[10] https://www.tawdis.net/index
[11] https://abc.useallfive.com/
[12] https://www.equalweb.com/
[13] https://www.w3.org/community/auto-wcag/

Tool Name	Guidelines following	Supported formats	License
508checker[6]	Section 508, US federal procurement standard	CSS, HTML, XHTML	Free Software
Accessibility Insights for Web[7]	W3C Web Content Accessibility Guidelines 2.1	HTML	Open Source
a11y Color Contrast Accessibility Validator[8]	WCAG 2.0 — W3C Web Content Accessibility	HTML	Open Source
Access Alchemy[9]	WCAG 2.1 — W3C Web Content Accessibility	CSS, HTML, XHTML	Commercial, Enterprise
Taw[10]	WCAG 2.1 — W3C Web Content Accessibility Guidelines 2.1	CSS, HTML, JS	Open Source
Accessible Brand Colors[11]	WCAG 2.1 — W3C Web Content Accessibility Guidelines 2.1	CSS	Open Source
EqualWeb[12]	German government standard, Irish National IT Accessibility Guidelines, JIS, Japanese industry standard, Israeli web accessibility guidelines, Stanca Act, Italian accessibility legislation	WAI-ARIA, CSS, HTML, XHTML, SVG, PDF documents, Images, Microsoft Office documents	Trial or Demo, Commercial, Enterprise

Table 3.1: Comparison of different accessibility checking tools available.

target people with partial visual disabilities. The tools which big corporations use are the likes of Access Alchemy and EqualWeb. These tools have the option to check for accessibility of whole web pages in one step. EqualWeb supports many different file types and guidelines. EqualWeb has guideline options for many countries. There is also different type of license available. Individual developers can also try the software using demo version rather than paying for the full version. This makes it one of the sought after tools on the market.

Chapter 4

Own Approach

The idea was to implement the project in two phases. First phase consist of a browser extension to test existing websites for the lack of aria integration and the second phase consist of a library, to ease the addition of aria roles when creating custom web components.

4.1 Basic Idea

Integration of the ARIA attributes into a HTML element gives that specific element more semantics. In return, this helps the screen reader to understand the role of that element in that web page and convey it to the end user. The screen readers use different types of accessibility APIs available to them for this purpose.

Websites nowadays are mostly made of different JavaScript frameworks. There are so many different JavaScript based frameworks and libraries available on the market now and out of these frameworks, many significant and more frequently used ones are using a component based structure for development. For example, from the start of the development of a website, using one of these frameworks, the developer can make a decision on the different parts of the project that should be componentized. The developer can choose to split the whole website into different components, like a header component, footer component, slider component etc.

By following this kind of componentized architecture helps the main web developer to split a single page into components and give the task of implementing individual components to different developers. Another idea behind having componentized architecture is that this way of coding helps with better code reusability. If a slider component is created for one page, the same component can be used in another page of the same project or also in another project without having to make so many changes to the basic code. This approach makes room for more questions about how this approach can be used to the advantage of the developer in order to implement accessibility. Each of these questions need to be answered to successfully finish this research.

4.1.1 Challenges in current approach

Starting with individual components and how these components can be made accessible. Using ARIA attributes is a good solution to extend the elements and make a component

accessible. But how to integrate different ARIA roles into different HTML elements is another challenge. Like any other challenge, there were different ways of looking at the problem. The first option to do this was to create a JavaScript library which can be added to a web page or component. When running the web page, the DOM elements will be created and when the elements are loaded, the library will automatically assign the ARIA features into the HTML elements. This method mainly targets the custom created elements which are more generic, the basic HTML tags already have these features inbuilt in them. In this approach, the elements in the page can be identified using the HTML DOM `tagName` property (see Prog. 5.3). By doing this the screen reader will be able to get more details about the elements in the page. This was one way of looking for a solution to the challenge. But by following this solution, it may be possible that other issues may arise on future testing. One main problem may arise from this approach is the accessibility of custom elements in the web page, which is explained in the following section.

4.1.2 Structure of the project

Custom elements are also HTML elements that are user defined. There is a provision to define elements, which can be new or extended from the native HTML elements. Nowadays, many web developers use this option of creating custom elements to have HTML elements with more features that they want. When creating the elements, the developer can define the name for the element. This means, that there are possibilities of different element names in the web page rather than having only native HTML elements. This is the reason behind the idea to implement a browser add-on to verify if there are any custom elements in the page and if there is any elements then, if the aria roles are correctly implemented in these custom elements. This is concerned with the first phase of the project, where the developer should be able to check and verify the aria integration into custom elements on the page. Using custom elements can be an issue if the first approach mentioned above is used to integrate ARIA into a page, because the library is using `tagName` property to identify the HTML elements, assign roles and other attributes. By having a custom named element on the same page, the accessibility of the page cannot be guaranteed. This accessibility concern regarding custom elements makes it important to look into the problem of accessibility and ARIA integration from a different point of view. Now there is a new criteria to be satisfied in order to provide accessible web pages. This criteria can only be satisfied if a different approach is taken than the first one mentioned. Web developers use different type of selectors to select different HTML elements on the web page. In this way, different operations can be done on a single element. If the same approach is taken into consideration for the ARIA integration into a web page, this opens up a new possibility for the accessibility of custom elements in a page also.

When using selectors in an HTML page, individual elements including the custom elements in the same page can be selected. The developer can define a selector like an id or class attributes for this purpose. It is a normal process for a web developer to define a selector for styling purposes or related. This same idea is used in the new approach, where a HTML selector attribute is used to get a specific HTML element. By doing so, the custom components can also be queried and it is possible to make changes to

the element, like adding ARIA attributes. Using this idea, the implementation of this project is carried out in two parts. First the planning and implementation of the browser add-on and the second being the planning and implementation of the JavaScript library.

4.1.3 Formal Requirements

The development phase of a project can always be a challenging part. The first and the most important challenge is to fully understand the things to be implemented in the project and come up with a good build pipeline to code the solution to the problem. Being a web project, choosing the correct web technology to construct the project is of utmost importance. There are many frameworks and libraries available to the developer on the current market, so choosing one of them and understanding the working and the future possibilities of the technology is very important. Modern JavaScript projects need more than just a framework to test, build and run the project. When creating a project, it is another task of the developer to choose other technologies which is needed for the testing, building and running of the project without any issues. Once the developer figures out the needed technologies for the building the project, the next part of the task will be to look into the future of the project. As this is a library for web projects, it is important that the library gets regular updates whenever there is an update in ARIA specifications or if there is a need to make a change in an already implemented component of the project. These new updates should be integrated into the library in a timely manner. There must be also emphasis into how the project should be maintained and how it can be made available to other developers who are working with other projects and would like to use the library in their projects. There should be also a possibility for other web developers to make contributions to the library if that is possible. For this purpose, the developer must find a good online solution. Like a package manager and repository, using these technologies, the developer can have a local installation of the library in the web project they are working on. Also, every developer will be able to access the source code of the library and make changes to it, maybe for extending functionality or cleaning a bug if needed. The online version of the library can be used to keep track of the issues in implementation or bug fixing and provide further support to other developers who also use this library.

4.1.4 Implementation and Integration

The implementation and integration are two parts of the project, which gives a better understanding about the current situation and future working of the project. The implementation part of the project gives an emphasis on how the library can be implemented with the available web technologies and what can be achieved if the implementation is carried out in a certain way. From starting the project from scratch to building it up into a complete library is a challenging task itself. In the implementation part, the developer looks more into own code and the integration of dependencies need for development, into one big project.

When it comes to the integration part of the project, the main objective here is the seamless binding of the library into the web component and how this can be maintained. The future use of the library in a web project should be smooth. If a web developer creating a web application or a website wants to include ARIA accessibility features into

the web components in the development state, then the web developer should be able to access the library online and install a copy of the library in the current developing project. After the installation of the library, the integration of the ARIA features to the web components come into play.

4.2 Browser Extension

The first part of the thesis project was to find a way to help the developer to understand the current scenario related to custom elements in a web page and the accessibility concerns regarding this. For this purpose a browser add-on is a good option, as the web pages which the developer want to check can be easily verified if a browser add-on is used. Also it can focus directly on the web page interface.

4.2.1 Requirements

As the method to focus on the first part of the project is checked, the requirements for this is also defined. The main requirements were, that the programmer should be able to see the custom elements in the page. It should be possible to identify and understand what are the accessibility concerns with a specific custom element in the page, if there is any. The issues related to an element can be of various types, ranging from absence of a role attribute to missing of update attributes and other aria attributes and states which are needed for accessibility. But the main attribute is the role attribute, which describes the element to the screen reader. So if that attribute is missing the developer should know it first hand. The options to do all these must be available easily to the programmer and also the browser add-on should have good support and should reach many users as possible. These were the main requirements regarding the browser add-on, the first phase.

4.2.2 Abstract Solution

The first and the important task the phase one, is to choose the API and Architecture for the implementation of the browser extension. A browser extension's architecture will depend on its functionality, but most extensions has multiple components. The main components being manifest[1], background script, content script [28]. The manifest file contains all the information about the extension, for example name, version, description to name a few details on the manifest file. The next component is the background script, this is the extension's event handler component. The listeners for the browser events are contained in the background script. It stays silent until a specific event is fired and then it performs the instructed logic. It is only loaded when it is needed. The next component is the content script, which is used by the extension to read or write to web pages. This contains JavaScript codes which executes in the page that has been loade into the browser. Content scripts can read and modify the DOM of web pages inside the browser. Fig. 4.1 shows the basic architecture behind the working and implementation of the browser extension.

[1] https://developer.chrome.com/extensions/manifest

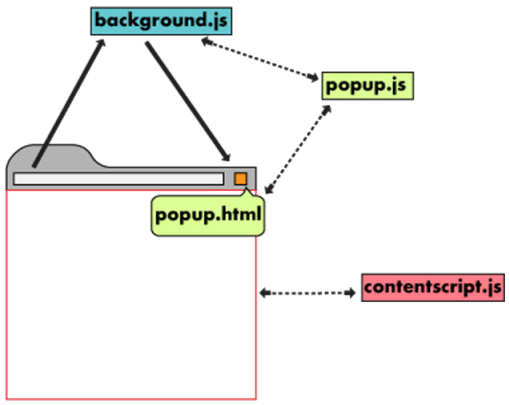


Figure 4.1: Browser Extension Architecture.

4.3 JavaScript Library

The second phase of the project consist of implementation of the JavaScript library. This part of the implementation looks more into the development of the library, integration of the library and also the testing of the components which is made using the library.

4.3.1 Requirements

In the second part of the project also the requirements are set at first. The main requirements included, the accessibility of individual components, individual component accessibility checking possibility, simplifying the integration of the library, testing the library with different tools. The main challenge in the integration part of the project is about how this is realized in projects from different developers. It is also important that the web developer who is using the library has an understanding about the working of the library. Only by knowing how the library can be used in another project, it is possible to use it in a real world environment without any issues. The components, which are made using the library should produce the intended results when used with a screen reader.

4.3.2 Abstract Solution

The primary task would be the focusing on the implementation of very important and basic ARIA attributes on the library, so when using the library, this data can be added to the main project and accessibility can be achieved.

Development of the Library

When it comes to the development phase of the project, there are many new technologies available on the market, which can be used for this purpose. But choosing the right technology is very important for the timely and structured development process. The challenge is to do the development of the library in a framework that is suitable for the implementation and integration of the library into future web projects. The library

Role	Attribute	Element	Usuage
`Button`		`div`, `a`	• Identifies the element as a button widget, • Accessible name for the button is defined by the text content of the element.
	`tabindex="0"`	`div`, `a`	• Includes the element in the tab sequence, • Needed on the `a` element because it does not have a `href` attribute.
	`aria-pressed="false"`	`a`	• Identifies the button as a toggle button, • Indicates the toggle button is not pressed.
	`aria-pressed="true"`	`a`	• Identifies the button as a toggle button, • Indicates the toggle button is pressed.

Table 4.1: ARIA button element authoring practices [21].

must be easily accessible to other web components. It should also be possible to give the accessibility features to the HTML elements in the component without much coding from the end developer (see Prog. 4.1). The basic ARIA features should be available and mainly the roles should be specified. The aria authoring practices is used as a reference to check which attributes are needed for a specific element to be accessible (see Table 4.1). It is also important to note that the library should be online in the future and easily accessible to developers all around the world. Upon finalising on a framework that will give solutions to the previously mentioned challenges, the next step in the process is to have a clear understanding about the development dependencies needed for the production of the library. There are different development dependencies needed for a web project to work. In the library project, apart from choosing the framework for development, it is also necessary to finalize on a test for testing purposes, compiler for compiling the code and a build library to build the project to run. These additional libraries are used in the project as development dependencies, because the end user who

```
class CustomButton extends window.HTMLElement {

  static propTypes = {
    text: 'aria-toggle-button'
  }

  connectedCallback() {
    this.root = this.attachShadow({mode: 'open'})
    ReactDOM.render(<MyReactComponent class='aria-toggle-button' />, this.root)
  }

  disconnectedCallback() {
    ReactDOM.unmountComponentAtNode(this.root)
  }
}

window.customElements.define('my-button', CustomButton)

export default class MyReactComponent extends React.Component {
  render() {
    return (
      <div>
        <button style={styles.button} className='aria-toggle-button'>
          my Button
        </button>
      </div>
    )
  }
}

const styles = {
  button: {
    backgroundColor: 'aquamarine'
  }
}
```

Program 4.1: Custom button element implemented with integrated aria toggle button attributes.

uses the final product, which is the ARIA library does not need to know the working of the dependencies used in creating the library.

Implementation and Integration

The implementation of the project is done in such a manner, that even the developer who is using the library components can easily check and verify the working of the components of the library. It is very important to follow a component based architecture. There can be different functions provided for different ARIA attributes, states and properties, because ARIA does not only consist of role attributes, but also with state and properties attributes. Even within the roles there are different sub divisions, named widget roles, composite roles, document structure roles and landmark roles. It is made

possible for the developer to use the library to add all these criteria into an HTML element in a single code or split these attributes and add the ones to the HTML element, that the developer wants.

The integration of the library is done using a component within the library. So the developer can add the component to any page and access the accessibility functions in that component and give these functions to the target HTML element in order to make it accessible. All this is done using different types of selectors. It is very important that there is a document available, explaining how to integrate the components from the library into the working project and test the components of the working project. A detailed description about the implementation of the project will be given later.

Chapter 5

Implementation

After shaping a basic theoretical approach and making different considerations, the next part was to define the required tools and set up the project. It is also important to verify that the final version of the project is dependent on additional libraries as little as possible. The structure of the project is very important from the beginning, having a folder structure at the start of the development makes it easier for the project to be maintained in the long term through asynchronous collaboration workflow. Also this will help in focusing on certain parts of the development process.

5.1 Considerations towards implementation

After looking at the challenges and possible solutions, the following topics can be identified as the essential ones:

- *Browser Add-On*: Best way to identify basic accessibility problems in a web page,
- *Framework*: Best solution for the implementation of the project,
- *Compiler*: Compile the code and produce output,
- *Build System*: To build the library,
- *Versioning*: Keep track of the development and possible revert, if needed,
- *Branching*: Helps to collaborate with different accessibility developers in future.

Git[1] qualifies as a core component in any web project, especially when there are multiple versions and the developers maintaining the project wants to keep track of the build process. Being aware of Bitbucket[2] and the benefits of its API, it serves as the foundation for the solution. Since the library should be accessible to all the developers, it makes sense to use the build pipeline and deploy the package to a package manager like Node package manager[3]. Use of a package manager will ensure that the library package is accessible to developers all over the world. Integrating the bitbucket development pipeline for the node package manager, will ensure timely deployment of the package when there is a new version available and the package passes test constraints.

[1] https://git-scm.com/
[2] https://bitbucket.org/product/
[3] https://www.npmjs.com/

5.1.1 Browser Add-on

The main factors to consider in the implementation process of the browser add-on is the API to use, the framework to work with the options to add to the add-on and finally the programming language.

Choosing a Framework

There are many frameworks available in the market now. The best practice in choosing a framework will be to choose a modern framework, which is easier to code. The framework should have good component based architecture capability implementation. It should also support good integration of development dependencies. React library was the best option available including all the criteria and ease of development.

Choosing an API for Browser Add-on

Nowadays there are different types of browsers available in the market. Before starting programming with a specific API, it is important to check if the browser add-on is available to most users. After some research it was found out that Chrome[4] browser has the best market share, standing at 63.69% [25]. The best way to reach more users was to choose Chrome API for the implementation of the browser add-on.

Choosing the language

For the browser add-on, due to the particularities of the task in hand, TypeScript[5] was chosen as the language for the implementation. This is a superset of JavaScript and it compiles to plain JavaScript on runtime. This was also an opportunity to work with TypeScript and understand its working and the difference with ECMA Script, which is the standardized JavaScript.

5.1.2 Library

The main factors to consider in the implementation of the library is the framework, Build system, Compiler and the versioning and branching system.

Choosing a Framework

To start building the project, the basic necessity is the framework on which it must be build upon. As it was stated before, there are many JavaScript frameworks available in the market now. Choosing the right framework for the job is important. First and foremost, the installation process of the chosen framework should be as easy as possible. There must be an option to use command line tools to start the project. The framework should provide component based architecture for implementation. It should support a good testing functionality and good integration of development dependencies. Using a framework with many packages already available online can seriously reduce the time needed for implementation.

[4] https://www.google.com/chrome/
[5] https://www.typescriptlang.org/

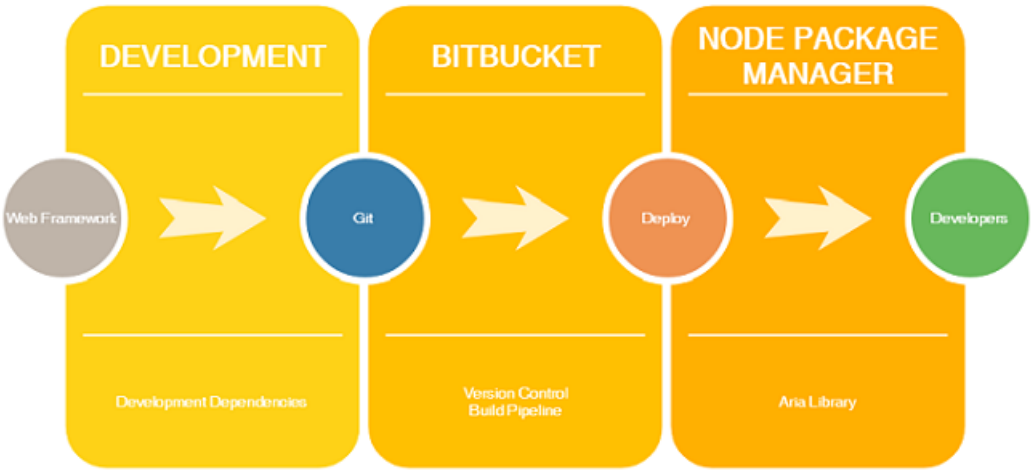


Figure 5.1: Visualization of the planned development and deployment phases.

ECMA Script programming language is being used for development. ECMA Script is the standardized name for JavaScript and ES6 refers to the version 6 of ECMA script. So the framework must have ES6 compiler support to compile the pages and also build support to build the project. The framework should also have a good community and development support.

All in all, the best solution currently seems to be React JavaScript framework[6], as it combines the user interface and the behaviour of the components. It has a component based architecture as required for the project. Also good support for different ES6 compilers and it is also easier to build and deploy online. Instead of using regular JavaScript for templating, React uses JSX. JSX is a simplified JavaScript, which allows HTML tag syntax to render subcomponents. This HTML tag syntax is then processed into JavaScript calls of react framework.

Compilation and Build process

In development phase, the project needs to be compiled and build before running and seeing the results. The compiler that is most commonly used with React projects is Babel[7]. One of the main reasons for the use of React and Babel together is because Babel can convert JSX syntax and it also tries to stay true to the ECMAScript as much as possible. JavaScript proved its universality due to its usage on both client and server side. The integration of React and Babel can be done seamlessly because of the extended support Babel gives to React projects.

Looking for a module bundler to build the project, Rollup[8] is well suited for this task, as it only consist of a very basic setup and can easily be setup (see Prog. 5.1). Rollup bundles small pieces of code into something larger and more complex. It uses a standardized format for code modules included in the JavaScript ES6 revision. The

[6] https://reactjs.org/
[7] https://babeljs.io/
[8] https://rollupjs.org/guide/en/

```
1 // rollup.config.js
2 import resolve from 'rollup-plugin-node-resolve';
3 import babel from 'rollup-plugin-babel';
4
5 export default {
6   input: 'src/main.js',
7   output: {
8     file: 'bundle.js',
9     format: 'cjs'
10  },
11  plugins: [
12    resolve(),
13    babel({
14      exclude: 'node_modules/**' // only transpile the source code
15    })
16  ]
17 };
```

Program 5.1: An example for a Basic Rollup.js and Babel setup [23].

main purpose of such a bundler in the application is to make the task of the developer easier by collecting all the pieces needed for the running of the library and then making a complex program out of these small pieces which can be later run in a server.

Fig. 5.1 shows an overall visualization of the planned development and deployment process. The web framework is used together with other development dependencies to create a project and then Bitbucket is used to manage and maintain the project online. Based on the respective configuration given, a build pipeline process may be automatically triggered whenever there is a new commit into a specific branch of the repository. After a successful build, the package is then deployed to the Node package manager.

Branching and Versioning

As mentioned before, maintaining the library on the long run is as important as the development of the library. Using Git, Bitbucket and Node package manager together will provide a good build pipeline and deployment solution for the library project. The setting up of these technologies can be a bit hard if the developer has no previous experience with these technologies. But once the developer gets in touch with these technologies, then it becomes rather easy for future development processes. Git is the most commonly used version control system on the market. Bitbucket provides a platform to store and share Git repositories.

Using a version control with the project is important in order to keep track of the the different versions of the project and it is also easier to collaborate with other developers for the project. Version control gives a better idea about the history of the implementation of the project. As the Git repository is stored in the Bitbucket as a public repository, then any individual looking into the repository can see the history of the project and also the currently working branches. Bitbucket makes it easier to collaborate with other people for the project. Another main part here is the build

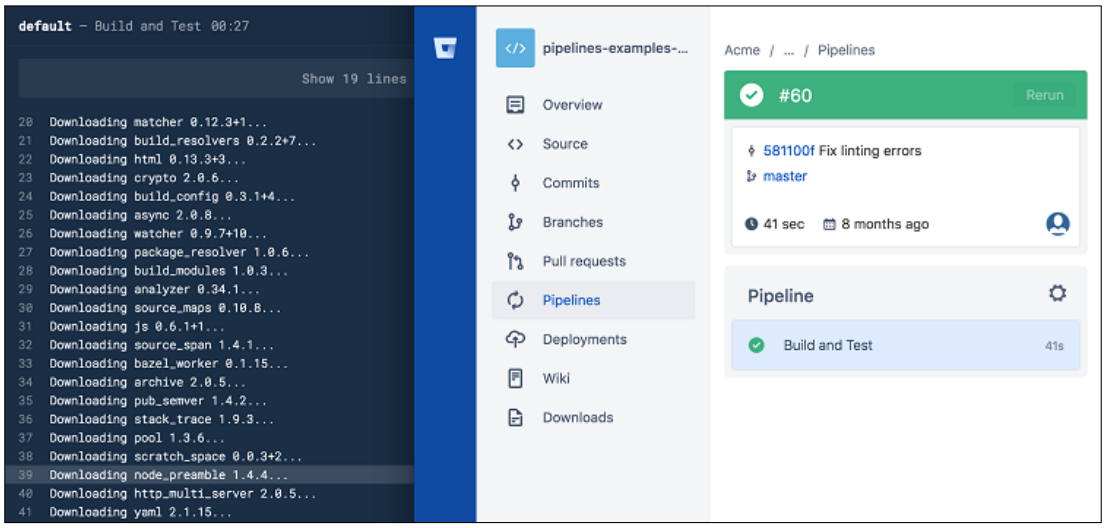


Figure 5.2: An example Bitbucket build pipeline.

pipelines available in the Bitbucket for npm integration (see Fig. 5.2). The developer can deploy the library into the repository and bitbucket takes care of the testing and build process and then deploy the library into npm (see Fig. 5.1). This makes future deployments easier.

5.2 Foundation

Since the basic considerations towards the implementation have been made. React JS is the JavaScript library which is selected to use for the main development purpose. There are also additional dependencies needed for the project to work. As the project is a JavaScript library which entirely works with the client side, there are no additional database requirements or need to implement a REST API. A version control system is used to keep track of the development history. The project is kept online in a repository, so it is accessible for other individuals. So now a basic development flow is finalized (see Fig. 5.1). Next part is to start with the implementation.

5.2.1 React JS

Starting with React JS, it is a JavaScript library for building user interfaces. React is easier to use and helps to make simple views for every state in an application. It will render and update the components automatically when the data changes (see Prog. 5.2). Declarative views makes the code easier to debug. React is highly component based [7]. Because the library project should be component based, it is easier to develop the library in a component based architecture. Using react, components can be made which are encapsulated and manage their own state. Then use these components in another view to make complex user interfaces. Because the component logic is in JavaScript, the rich data can be passed through the app and in the same time the state can be kept out of the DOM.

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }

  tick() {
    this.setState(state => ({
      seconds: state.seconds + 1
    }));
  }

  componentDidMount() {
    this.interval = setInterval(() => this.tick(), 1000);
  }

  componentWillUnmount() {
    clearInterval(this.interval);
  }

  render() {
    return (
      <div>
        Seconds: {this.state.seconds}
      </div>
    );
  }
}

ReactDOM.render(
  <Timer />,
  document.getElementById('timer-example')
);
```

Program 5.2: An example React Timer application, where props are given to *Timer* class and then it is rendered using the *render* method [35].

Components allow the programmer to split the user interface of an application into independent, reusable parts, and work on each part in isolation. components work like normal JavaScript functions. They take inputs called "props" and return elements describing what should appear on the screen. There are also different lifecycle methods available in React, to manage state and run codes in specific time of execution. There are two lifecycles methods used in Prog. 5.2. The `componentDidMount()` is executed after the component output has been rendered to the DOM. The `componentWillUnmount()` is invoked immediately before a component is unmounted and destroyed. This method is mostly used to perform any necessary cleanup.

There are so many different react packages available in the package manager, which will help the developer to easily setup the basic structure. The *create-react-library*[9] contains a package for creating reusable, React libraries using Rollup and *create-react-*

[9] https://www.npmjs.com/package/create-react-library

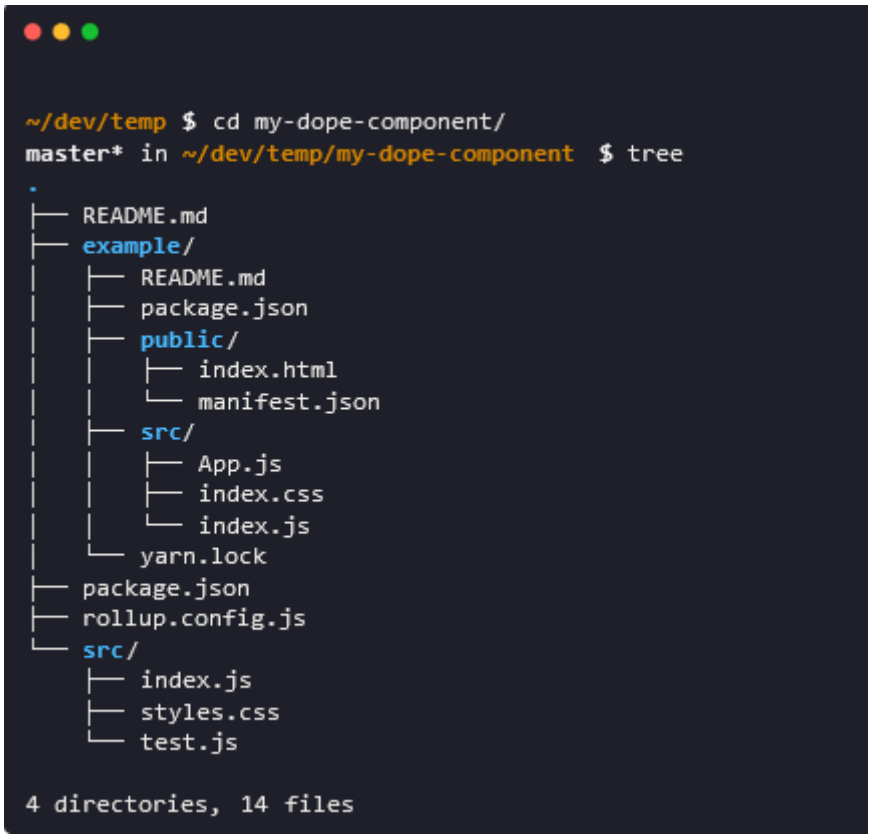


Figure 5.3: Tree structure of an example react library directory.

app[10]. Using this CLI to install and setup the basic project structure will be an enormous help to the developer. By using `npm install -g create-react-library` command, this package is installed globally on the system. After installing the package in the system, using the `create-react-library aria-roles` command, a new project can be setup. There is also provision to choose from different options when the project is setup, like description, author, repository, package manager etc.

The completion of the project setup will give a clean directory structure (see Fig. 5.3). The directory is included of a source folder, for the source files including the test class. A `package.json` file, which contains the information about the project and also the dependencies used for development. Next is the `rollup.config.js` file, this file contains the configuration to build the project and produce output (see Prog. 5.1).

5.2.2 Bitbucket

Since the project will require version control to manage current and future codes, it is suitable to host the library on a git repository hosting website like Bitbucket[11]. Bitbucket provides a platform to store git repositories online. In the current development phase, it

[10] https://github.com/facebook/create-react-app
[11] https://bitbucket.org/product/

is important that the developer keeps track of the changes which is made on the library on every stage of the development. Also there can be different versions of the library, when the ARIA specifications are updated. These new specification changes must be updated on the library also and another version of the library is produced in a timely manner. In the future development of the library, if there is a need for collaboration with another developer, the branching system from the Bitbucket API can be used to split the work load and share it among other developers. Another important advantage in using Bitbucket as an online solution is the availability of build pipelines. Once the coding is finished, it is uploaded to Bitbucket and the build pipelines in the Bitbucket will take care of the deployment of the library into the Node package manager. Thus making it easier for the developer to manage the code, test and deploy the application.

5.2.3 Node Package Manager

Once the project build is successful in the Bitbucket, automatic deployment to the node package manager can be set. This helps the developer to do two tasks in a single step. Node Package Manager (npm) is a package manager for the JavaScript programming language. In just six years since its establishment, npm has become one of the largest software ecosystems, hosting more than 230,000 packages, with millions of package installations every week [11]. It is the world's largest software registry and also the default package manager of the of Node.js, a JavaScript runtime environment. It also consist of a command line client, an online database for public and a paid-for private one for private packages. This database is called npm registry and this can accessed to search and download for available JavaScript packages. Developers use npm to share and borrow packages. When the JavaScript library is being developed, it was necessary that some packages from the npm registry is borrowed for the development process and also the library should be made available for the other developers to use in their own projects. The best way to do this is by using the npm registry.

5.3 Browser Add-on Implementation

The basic and most important step on the implementation process of the browser add-on is to understand the program flow of the Chrome extension API [27], which is used for the implementation process. There are different operations which can be carried out using extension API, like exchanging messages between an extension and its content script or between different extensions. The main parts of an extension is background script, content script and manifest.json file. Using these three files as backbone an extension interacts with the browser and the user. There is another part on the project which is called the popup section, which is used to show a popup window (see Fig. 5.5) to the user when the user clicks on the add-on icon. This popup will have the options inside, within these option user can select the option for the operation the user wants to perform on the page. The background script works on the background coordinating the working of the add-on. Prog. 5.4 shows an example manifest file and in this file different options are given regarding the working of the add-on and the dependent files.

The other two types of file associated with a browser add-on are the background script and the content script. As the name suggests background script works on the

```
for (let i = 0; i < elements.length; i++) {
    if (elements[i].tagName.indexOf("-") !== -1) {
      elements[i].style.backgroundColor = "#CCCCCC";
      elements[i].classList.add("tooltip");
      const elementsRole = elements[i].getAttribute("role");
      const elementsDescription = elements[i].getAttribute("aria-describedby");
      let feedbackMessage;
      if (elementsRole == null && elementsDescription != null) {
        feedbackMessage = "Aria Description Found " + elementsDescription;
      } else if (elementsDescription == null && elementsRole == null) {
        feedbackMessage = "Aria Description and Role not Found";
      } else if (elementsDescription != null && elementsRole == null) {
        feedbackMessage = "Aria Role Found " + elementsRole;
      } else {
        feedbackMessage = "Aria Description Found " + elementsDescription + "\n" + "
     Aria Role Found " + elementsRole;
      }
      elements[i].title = feedbackMessage;
      elements[i].setAttribute("data-toggle", "tooltip");
   }
}
```

Program 5.3: Loop to find the custom elements in a web page.

```
{
 "manifest_version" : 2,
 "name": "UC Grabber",
 "version": "0.1",

 "permissions": [
 "activeTab"
 ],
 "browser_action": {
 "default_icon": "images/icon.png",
 "default_popup": "html/popup.html",
 "default_title": "uChicago Canvas!"
 },

 "content_scripts": [
 {
    "matches": [
  "*://canvas.uchicago.edu/*"
    ],
    "js": ["scripts/content_canvas.js"]
  }
}
```

Program 5.4: `manifest.json` file example.

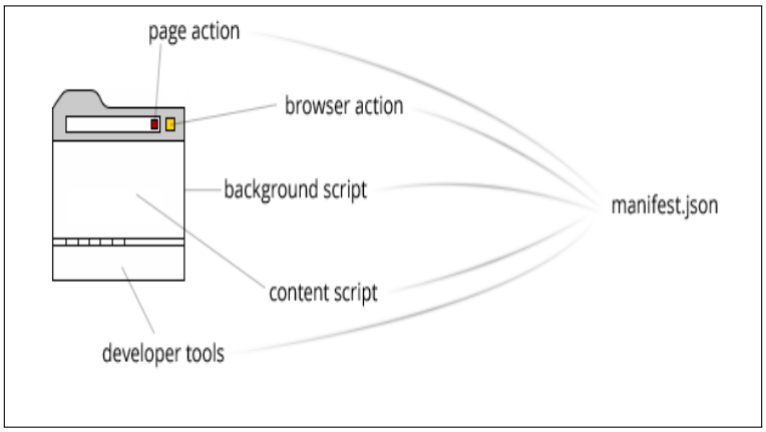


Figure 5.4: Working of a browser extension.

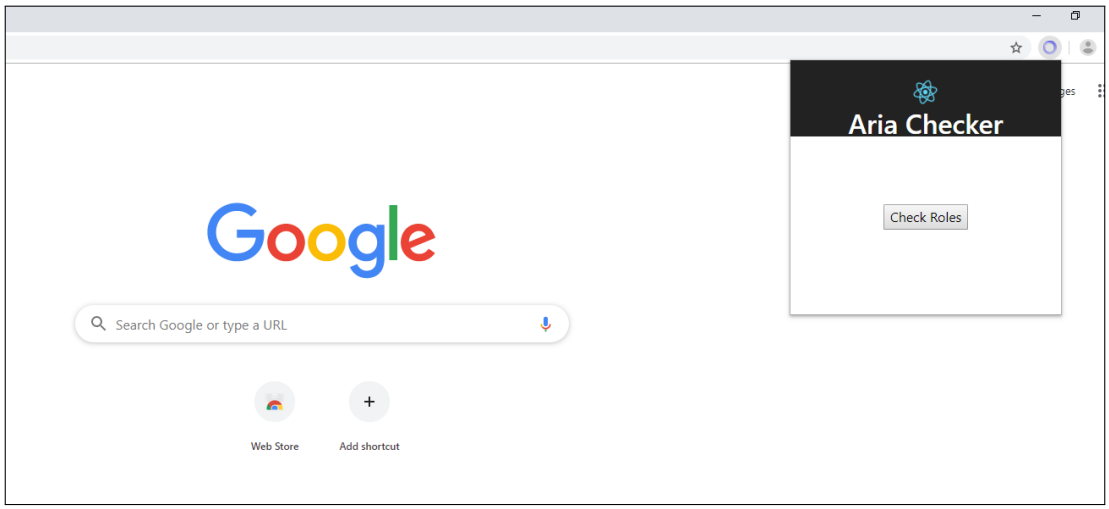


Figure 5.5: Browser Add-on Pop up.

background of the browser and the it helps to check the browser action and pass the message to the content script (see Fig. 5.4). Then the content script is used to make the changes to the content of a web page. Using this method, it is possible to manipulate all the custom HTML tags in the current active page and make changes to these tags, if there is no ARIA roles available to them (see Prog. 5.3). When there are no roles or description found on the element, then the element is highlighted and then a tooltip option is provided for the developer to see which is the element in the page and what are the attributes missing in the element.

If there are no roles and description available in the custom HTML tag, then it is pointed and highlighted. By implementing the browser add-on this way helps the programmer to see what is missing and which ones are the custom elements in the active page.

5.4 Library Implementation

One of the most important step prior to starting the development process is finalising on a structure for the library. It is important that the developer have a clear idea about how the implementation process should go and what are the additional dependencies needed for the project to work on a real world environment. As discussed before, the developer of the project should look into the questions need to be answered and come up with different solutions to a challenge.

Using React JS to develop the library means that, the contents of the library can be created as components. In a project, these components can be imported and bounded to an user interface. This type of approach will give more control to the developer over the elements in that user interface. The developer should be now able to access the HTML elements on that page using different queries or methods. This in turn gives the developer a possible solution to getting a specific element in the page. This is the basic working principle of the library.

5.4.1 Basic setup

The foundation for the development starts with creating a basic project using the `create-react-library` command. Using this command to download a package from the node package manager and running this package will help the developer with creating the library project with ease. The directory structure implemented while using the command to create the project is clean (see Fig. 5.3). It is important to keep a clean directory structure from the very beginning of the project implementation. Having a good directory structure at the start of the development makes it easier for the project maintainers to maintain the project in the long term through asynchronous collaboration.

The source folder (`src`) is where the main body of the project is developed. The source folder consist of mainly three files, `index.js`, `styles.css`, `test.js`. The main JavaScript file in the project is the `index.js`. The `styles.css` is the cascade style sheet file, which can be used to provide styling. And finally the `test.js` file, which is used to write tests to check the working of the project. The code needed for the working of the library is codded in the `index.js` file. There is also an example project available with this package, this project can be used to test the library. If the developer wants to test the library in another test project, there is also provision to do that. There are commands which can be used to link the library package to another project for testing purposes. The most common command used for package linking is `npm-link`[12]. The package linking process is a two step process, using this command in the project folder will create a global symbolic link. This link can be used to connect the library to another project as a module. In the root folder of the project, which the library is to be linked to, use command `npm-link package-name` to connect the library to the project. It is important to notice that, the `package-name` specified in the command is the name of the package on the `package.json` file, this is not the folder name of the library project.

The basic setup of the project also consist of different development dependencies, which helps in the development of the library and also there are dependencies of the

[12] https://docs.npmjs.com/cli/link.html

project, which the project will need to work on other systems. The details about the dependencies of the project is available in the package.json file in the root directory of the project.

5.4.2 Component implementation

To use the library in another project, it is important to create the library as a component. So the basic stage of the development process is the implementation of this component, which is used to import the whole library in another project. As a basic step, the `AriaComponent` is created in the library project. Inside the component, functions can be created that will implement necessary ARIA features into the target page. The next part is the crucial step in the library implementation, this is the about how and what should be the changes made to an element to make it accessible according to ARIA standards. Like, addition of which attributes will make a specific component accessible. For this purpose, WAI-ARIA Authoring Practices[13] is used as a reference. Using these authoring practices, it will be more clear to understand how to create accessible rich internet applications using WAI-ARIA. This document also gives important information about approaches to make widgets, navigation etc. This is the go to guide for web application developers to understand the implementation requirements of ARIA for a specific HTML element.

In the `AriaComponent`, different functions and listeners are created for different HTML elements in target project page. When the library is bound to a page or a component in a project, it will check for the `className` attribute of the HTML elements in the web page. There are already specific classnames described in the library to access specific elements in the target page. So, when the developer of the target project wants to add ARIA features using the library, it is possible to use these specific `className` attributes to get the individual elements attached to specific functions in the library, then the library functions can make necessary changes to these elements to make them accessible.

Sometimes the developer may only want to add some attributes to an HTML element, e.g. `aria-pressed`. Using the library also provides the option to do only a specific task also, if that is what the developer needs. As the requirement of the developer changes for different elements, there is options available to integrate these changes to the elements. For these type of attributes, which needs to be updated, there are different type of listeners available in the library. The library uses these listeners to catch an event happening in the target HTML element. When a click or press event occurs on a button, and this button is attached to the library using that specific classname, then the library will get these updates and make necessary changes and convey it to the end-user.

Table 4.1 from the authoring practices, show the necessary role and attributes needed to make an element a button or a toggle button. This practices are instructions which are followed throughout the development of the library. In the library project, an `ariaButton()` function is created to check the accessibility needs of a button and an `ariaToggleButton()` function is used to check the accessibility of a toggle button in the target HTML page. The developer writing code for the target HTML page can intro-

[13] https://www.w3.org/TR/wai-aria-practices-1.1/

```
1 import * as React from 'react';
2 import '../index.css';
3 import AriaComponent from 'aria-roles';
```

Program 5.5: Importing React and Aria library into a component.

duce the library to the web page using a specific import statement (see Prog. 5.5) and by adding an extra `className` attribute `className = 'aria-button'` to an HTML element that requires the ARIA accessibility specifications for a button (see Prog. 5.7), the accessibility requirements can be attached to that button element. In the library project, there are functions like `ariaButton()`, which is loaded when the target page is mounted. So, the function will check if there is a `className = 'aria-button'` attribute set on an element in the page. If there is an element with this attribute, then the function is called to make necessary changes on the element. This function works in accordance with the ARIA authoring recommendations for a button.

In Prog. 5.7 the `ariaButton` variable takes the elements with the `className = 'aria-button'` and the function checks if there is already a role attribute present in the element, also what if there is a role attribute then what is the current value of the attribute. If the attribute value is null or something other than button, it this is not correct implementation of an ARIA button, according to the authoring document. So the function will take necessary steps to make these changes on the target element. There is also a option to set the tab sequence of the element, using the `tabindex='0'` attribute. This attribute includes the button element in the tab sequence.

In Prog. 5.8 the `ariaToggleButton` variable takes the element with `className = 'aria-toggle-button'` attribute set in the target HTML page and here also the first and the basic step is to check if there is already a role attribute in the element with the `aria-toggle-button` className attribute. If there are elements with this attributes set, then same as a button element the role attributes is set as button. But the main change that differentiates a button and a toggle button is the `aria-pressed` attribute. When the button role and `aria-pressed` attribute are used together then the assistive technology will know that this a toggle button. The assistive technology uses `aria-pressed` attribute to convey the current state of the toggle button to the end user. When there is a change in the state, this information can be updated to the user including the current state of the toggle button (see Prog. 5.8). As mentioned before, event listeners are used for this specific task. When an element in the target page have the `aria-pressed` attribute, then these event listeners will also check on that element, when the element is active (see Prog. 5.6). Any changes to an active element with `aria-pressed` attribute will reflect on the library and the library functions will make these changes to the `aria-pressed` attribute and update the target element. In Prog. 5.9, the aria-button is used in a span HTMl tag to give semantics of a button to this tag. When the UI is rendered then the span is rendered including the changes which are needed to give semantics of a button to the span element (see Fig. 5.6). It is clearly visible that the attributes that are needed according to the authoring practices, for the element to work as an ARIA button is implemented and recognised in the span element.

The ARIA authoring practices are used throughout the implementation phase as a

```
ariapressedEvent() {
  let activeState = document.activeElement.getAttribute('aria-pressed')
    if (activeState === 'false') {
        document.activeElement.setAttribute('aria-pressed', 'true')
        return
    }
    if (activeState === 'true') {
        document.activeElement.setAttribute('aria-pressed', 'false')
    }
}
```

Program 5.6: Using `ariapressedEvent()` function to convert a button into a toggle button and changing the `aria-pressed` state according to the button state.

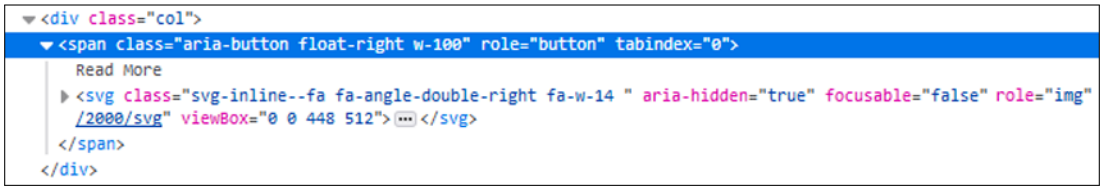


Figure 5.6: change after adding the aria-button className to the span element.

```
ariaButton() {
    let ariaButton = document.getElementsByClassName('aria-button')
    for (let i = 0; i < ariaButton.length; i++) {
      if (!ariaButton[i].getAttribute('role')) {
        ariaButton[i].setAttribute('role', 'button')
      }
      if (!ariaButton[i].getAttribute('tabindex')) {
        ariaButton[i].setAttribute('tabindex', '0')
     }
  }
}
```

Program 5.7: Changes made to a HTML tag with `aria-button` as className.

reference, in order to understand what are the different combinations of attributes to be mixed together to attain accessibility for a specific element. For different elements different combinations of the attributes are used for accessibility, wrong combinations of attribute may seriously affect the accessibility of the whole page. So it is important to review individual elements and their attribute combinations on every stage of the implementation process.

5.5 Deployment

Once the basic development is finished, the next part of the implementation process is to upload the library to an online git repository for managing current and future versions

```
ariaToggleButton() {
    let ariaToggleButton = document.getElementsByClassName('aria-toggle-button')
    for (let i = 0; i < ariaToggleButton.length; i++) {
      if (!ariaToggleButton[i].getAttribute('role')) {
        ariaToggleButton[i].setAttribute('role', 'button')
      }
      if (!ariaToggleButton[i].getAttribute('tabindex')) {
        ariaToggleButton[i].setAttribute('tabindex', '0')
      }
      if (!ariaToggleButton[i].getAttribute('aria-pressed')) {
        if (ariaToggleButton[i].checked) {
          ariaToggleButton[i].setAttribute('aria-pressed', 'true')
        } else {
          ariaToggleButton[i].setAttribute('aria-pressed', 'false')
        }
     }
  }
}
```

Program 5.8: Using `ariaToggleButton()` to make changes to the HTML tag with `aria-toggle-button` as className.

```
 <div className="row">
    <div className="col">
        <span className="aria-button float-right w-100">
            Read More <FontAwesomeIcon icon={faAngleDoubleRight} />
        </span>
    </div>
</div>
```

Program 5.9: Adding `aria-button` to a span HTML tag.

of the library. The chosen git repository for the project is Bitbucket. Bitbucket provides pipelines to build, test and publish the library package to the node package manager. Using Bitbucket pipelines takes a large amount of workload from the developers hand. In normal scenario, developer use the git repository for version control and publish the package to the package registry separately. But with Bitbucket there is inbuilt options available to publish the package to different online sources, thus making the job of the developer more simpler.

5.5.1 Bitbucket Pipelines

Using Bitbucket pipelines for deploying the package is a fairly simple process. There are inbuilt options available in Bitbucket, where the developer can give in details about the user account in the npm registry and also token generated from npm registry, to connect the repository to npm. To start with the pipelines, first the developer will have to select the branch in the repository which should be deployed (mostly common deployed branch is the master branch) and add a new file called `bitbucket-pipelines.yml` to

aria-react / bitbucket-pipelines.yml

```
# This is a sample build configuration for JavaScript.
# Check our guides at https://confluence.atlassian.com/x/14UWN for more examples.
# Only use spaces to indent your .yml configuration.
# -----
# You can specify a custom docker image from Docker Hub as your build environment.
image: node:10.15.3

pipelines:
  default:
    - step:
        caches:
          - node
        script: # Modify the commands below to build your repository.
          - npm install
          - npm test
    - step:
        name: Publish
        deployment: production
        script:
          - pipe: atlassian/npm-publish:0.2.0
            variables:
              NPM_TOKEN: "4e79c4d4-5d38-4d18-91e5-46f3a33b91f2"

```

Figure 5.7: `bitbucket-pipelines.yml` file.

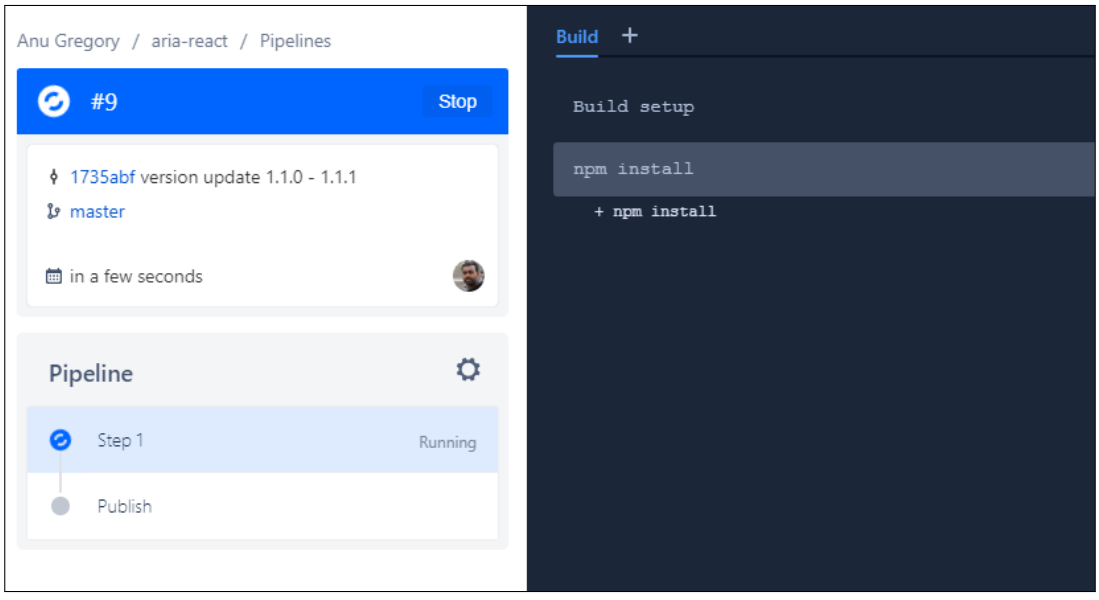


Figure 5.8: Running Bitbucket build pipelines.

this branch (see Fig. 5.7). This file holds the initial deployment steps which informs the Bitbucket API that, this is the branch to be deployed and also the necessary steps to deploy the package is also listed in this file.

In `bitbucket-pipelines.yml` file, developer can specify the scripts to run on the

Figure 5.9: Bitbucket pipeline dashboard showing status.

package to verify it before publishing it to the package manager. The two scripts which runs in pipeline file are `npm install`, to install the node modules and `npm test` to run test classes to see if there is any issues before starting deployment step (see Fig. 5.7). The token variable is got from npm, this in turn helps to connect the package to the npm registry.

Once the necessary information needed for build is set using the bitbucket-pipelines configuration file, in the branch developer wants to deploy to npm, then it becomes simple to do future build, test and deployment processes. For future development of the project, it is mainly done in other branches than the master branch itself. Once the development process is finished and the next version of the package is finalized, then these changes in other branches can be merged into the master branch using pull requests. Whenever there is a new commit in the master branch with a new version the build pipeline is automatically triggered (see Fig. 5.8).

Even though the pipeline is automatically triggered when there is a new commit in the master branch, the developer have the option to stop the pipeline before it is published successfully. If this is the case then the build status is set as 'stopped'. If there is an error in the build process, the status will be set as 'failed' and if the build is successful and the package is deployed to npm successfully, then successful' status is set (see Fig. 5.9). The developer can keep track of the status of different commits in the pipeline dashboard of Bitbucket repository.

5.5.2 Node Package Manager

As mentioned before, npm is one of the largest software ecosystems, there are thousands of packages available with millions of package installations every week. Deploying the package to an online package registry like npm means, this package can be accessed by developers all over the world easily, they can integrate the package to their own projects or download a local copy and make changes of their own to the source code. To install a local copy of the package in a remote system, the developer just need to run `npm i -s react-aria-roles` command. In this way, the package can be distributed among developers from all over the world using the node package manager.

5.5.3 Extensibility

One of the major factors considered in the modern web development is the extensibility of the component that is created. In the library project, there is a possibility to extend the project to include many more available aria roles. Currently the library supports elements like, Button, Toggle Button, Checkbox, Combobox, Link, Slider etc. Many

new roles can be added to the library using the same method and also when there are changes in the ARIA specifications, the library can adapt to these changes without much effort. Version control plays a big part in future development of the library.

Chapter 6

Evaluation

After the implementation of the project, the two phases are tested and evaluated on different scenario's to verify the usability of the project in the real world environment. The two parts are tested separately and the results are noted to see if they fulfill the requirements.

6.1 Browser Extension

To prove the usability of the project, the extension is tested in a web page with custom elements and also in a web page where the custom elements are absent. It is checked to see if the before mentioned requirements are fulfilled and the project is suitable for real world use. The main requirements for the browser extension were, when it is used to check for elements in a web page, the extension should retrieve the custom elements in the web page, then the aria specific attributes of these custom elements should be evaluated and finally the extension should be able to communicate with the user regarding the current state of the aria attributes in the checked custom element.

6.1.1 Results

When the browser extension is tested on a web page with not custom elements, it did show the alert saying "no custom element found" (see Fig. 6.1). This was the intended result and this is being realised without any error. The next part of the evaluation process was to test the extension on web page with custom elements (see Fig. 6.2). The result from this test was also successful, as the custom elements in page was highlighted and on mouse over, there was a tooltip present with the details about the attribute missing on that element.

6.2 JavaScript Library

To prove the project's usability, it has been tested with different accessibility checking tools available. The tools used are browser add-on accessibility checking tools, which also have a single page testing mechanism. Using web components in the development means that when testing, it is possible to test the single component rather than the whole

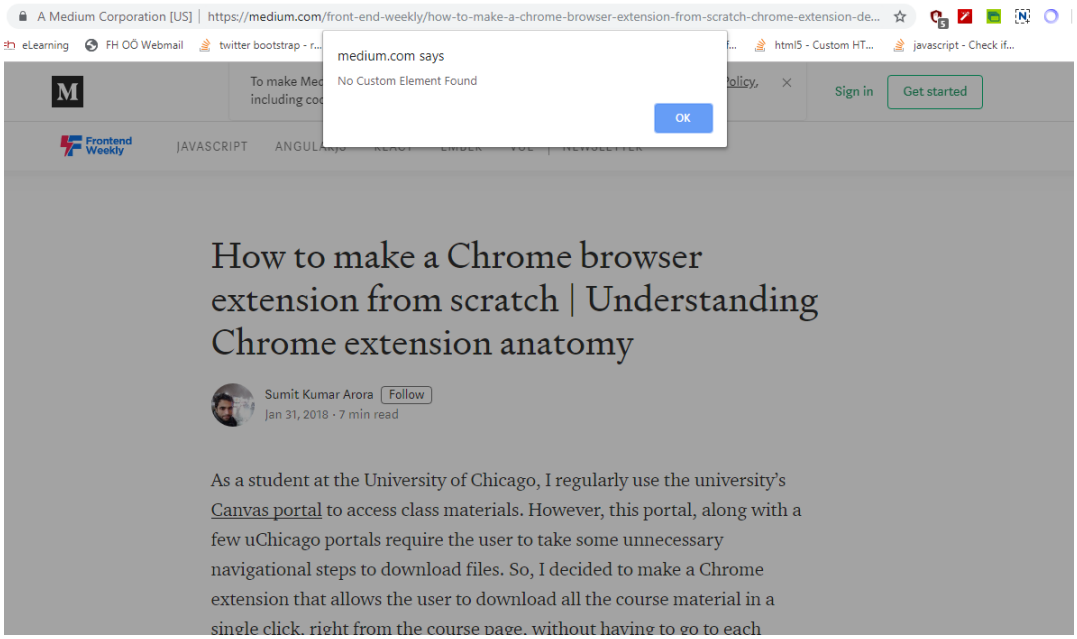


Figure 6.1: Result when there are no custom elements on the page.

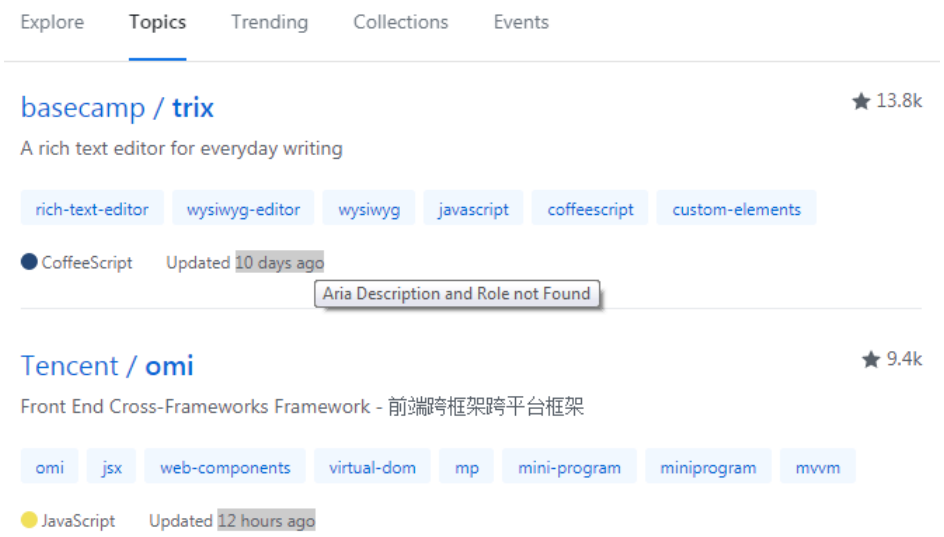


Figure 6.2: Result when custom elements are present on a page.

page. So basically when the components are checked individually, it makes it easier for the developer to quickly find the solution to the problem in hand and also change how the component behaves without affecting other codes in the same project. When the testing is done and the component passes the tests, then this component can be used any number of times in a project without need of another test unless there is some changes

```
<div className='row'>
    <div className='col'>
        <span className='aria-button float-right w-100 border-dark'>
        Test Button
        </span>
    </div>
</div>
```

Program 6.1: Button element.

made to the component. Even though individual component testing is a good way to test the web components. It is also important to check the whole page once to verify that other codes are not interfering with the accessibility of the individual components in a page. The testing programs will be used for checking the accessibility of the individual components are WAVE Accessibility Extension by WebAIM[1] and Accessibility Insights for Web[2].

6.2.1 Minimum Requirements

When the implementation started the minimum requirements were set as that, the library should be able to add roles to the HTML tags in a page automatically, so the active page is checked to get the HTML elements and then it is verified if the roles are already implemented for the elements present in the active page. If not, then add the aria role attribute to the elements. On the third iteration of the project the requirements were updated and now it was important to check for the aria supporting attributes also, which in turn helps an assistive technology to understand the updates on the page and give the updated information to the end user. To do this, the library should not only check for the aria roles attributes but also the corresponding supporting attributes of that specific role. Then the library should be tested for the correct aria implementation when it is integrated with web components and it should pass the tests with different web accessibility testing tools. Also check with a screen reader to verify the usability of the components.

6.2.2 Results

Here two components are tested with two different accessibility tools and then the results are noted to check if this approach is good for the future development of the accessible components. The two components selected for testing purposes are, a Button component (see Prog. 6.1) and a Toggle Button component (see Prog. 6.2). These components are tested individually and together in a web page to see if it makes a difference to check the components individually or together with other components.

From Fig. 6.3 it is clear that the Accessible rich internet application attributes (ARIA) are correctly implemented for the button component. The WAVE accessibility checking tool will give details about how many ARIA attributes have been detected for

[1] https://wave.webaim.org/

[2] https://accessibilityinsights.io/docs/en/web/overview

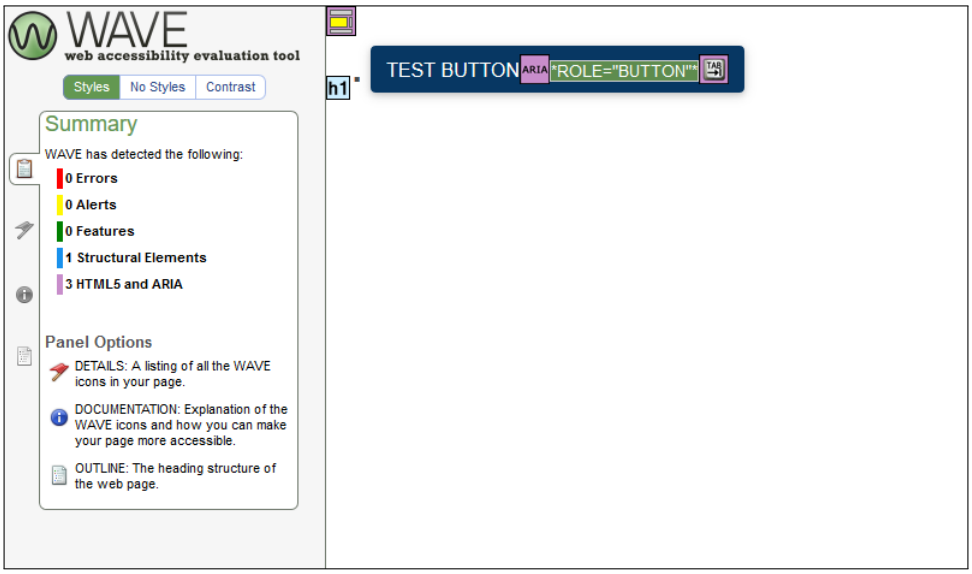


Figure 6.3: Button component test results using WAVE accessibility testing tool.

```
<div className='row'>
    <div className='col'>
        <span className='aria-toggle-button slider round'>
        Toggle Button Test
        </span>
    </div>
</div>
```

Program 6.2: Toggle Button element.

the checked component and in the component interface, it will show more details about the implementation. Here as the result is shown, there is also visible changes to the tested component showing the features like tabbable and that the button component is inside a header H1 and the header is inside a main aria component. Wave also shows the role of the tested component. It is also possible to click on one of these details and get more information about what is currently behind the component. Here the result shows that the role attribute is correctly recognised and also the tab option is also recognised without any problems. The same button component is then tested with Accessibility Insights for web tool and with this also the component has passed the test without any errors (see Fig. 6.4). The test is done using the Fast Pass option from the Accessibility Insights for web tool, so it gives a more generic evaluation of the component.

The toggle button component is tested using WAVE accessibility tool and the results are shown in Fig. 6.5. More attributes than that of a basic button is available for the toggle button. The main difference of the toggle button and that of a basic button is that the toggle button have `aria-pressed` attribute, so the user know the current state of the toggle. This button is also tabbable. The toggle button is tested using the Accessibility Insights for web tool and in this case also the toggle button component passed the test

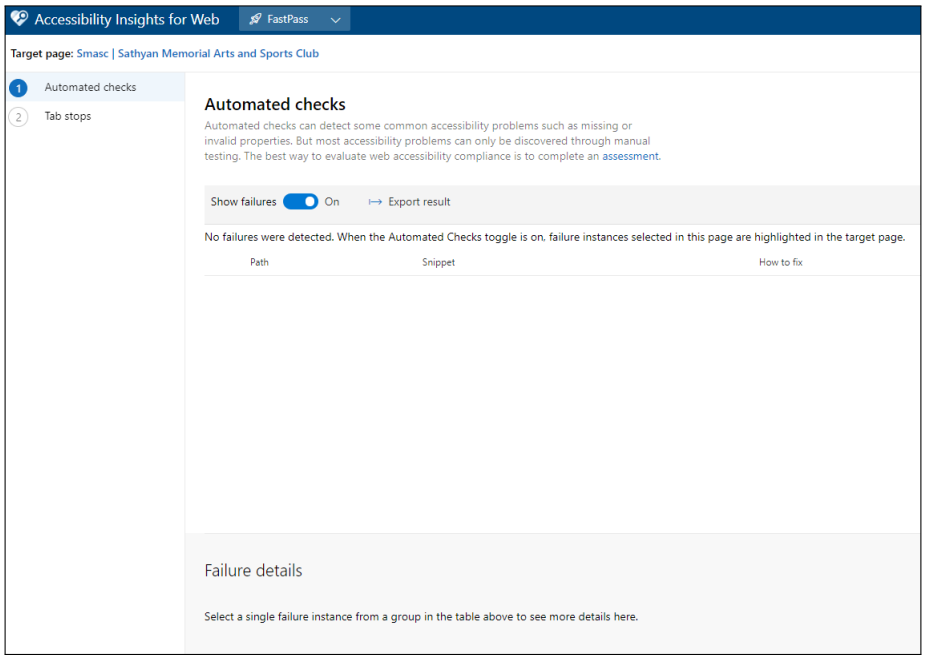


Figure 6.4: Button test results using Accessibility Insights.

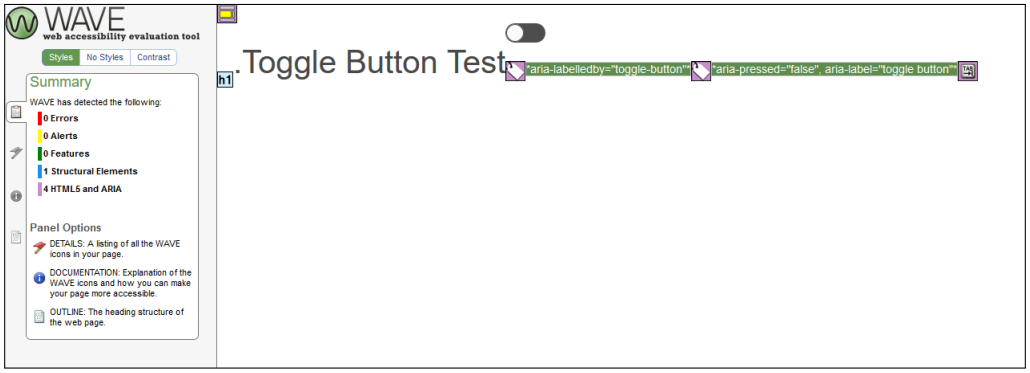


Figure 6.5: Toggle Button test results using WAVE.

without showing any failures (see Fig. 6.6). In the final stage of the testing, the two components are tested together in a single page to see if it shows any failures. WAVE accessibility tool is used in this case to get more feedback. Fig. 6.7 shows the result of testing the two components together and the result of this test was also positive. All the ARIA features are also shown in this result making it sure that these two components is accessible individually and also together in a single web page. Fig. 6.8 shows the result of testing a complete webpage containing different individual components including components using the aria library. The ARIA features are correctly implemented and recognised. In this way these components can be used to build websites. In addition to this, the individual components then where tested with different screen reader software's to verify this results. These tests passed without failures.

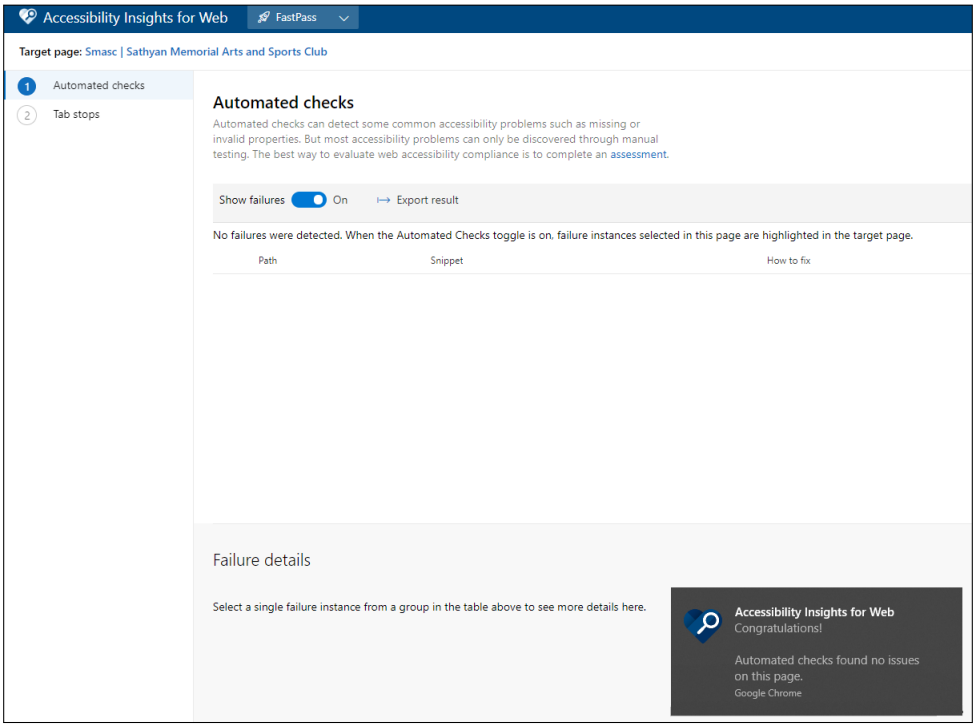


Figure 6.6: Toggle Button test results using Accessibility Insights for web.

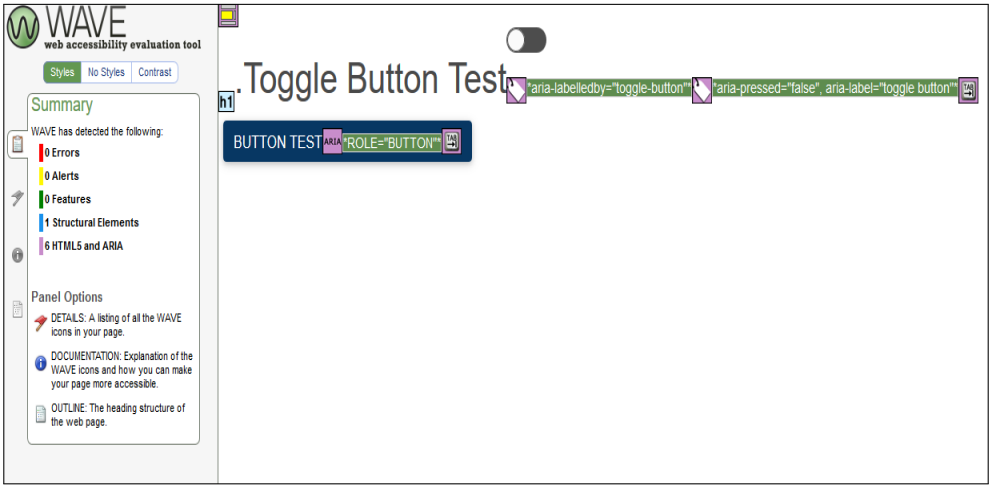


Figure 6.7: Button and Toggle Button test results using Wave.

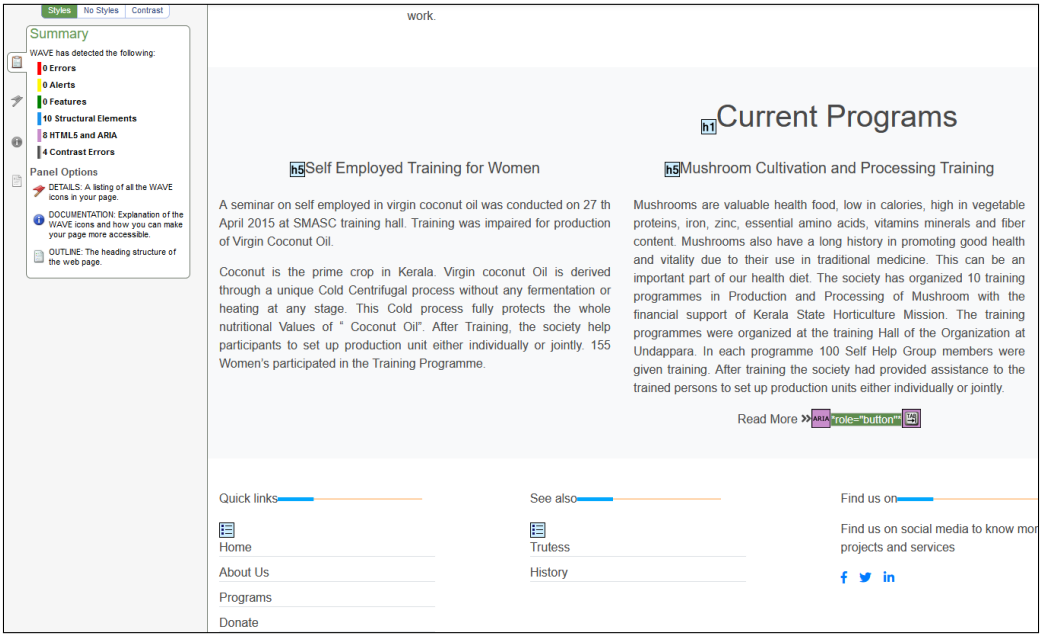


Figure 6.8: Complete webpage test using WAVE, when the library is used.

Chapter 7

Conclusion

The interest in web accessible components evolved when I was working with a personal website project for a social service society in India. On this work, I was looking more into the Accessible Rich Internet Applications (ARIA) attributes to ensure that the HTML page is accessible. This made me realise about the need for web accessibility in modern day web and the need for an easier way to implement the accessibility features on my future projects and be more productive with the time in hand. On my research I found out that, only around 10 - 15% websites are accessible to all the people. Developers don't go much into accessibility programming for various reasons. So what can be a good way to make it happen with less work. Because most of the frameworks I use for web development works with components, I thought that will be a good place to start my research.

The first idea was to implement a library which will automatically assign aria role attributes to the HTML tags in the page, but with more research I found out that native HTML elements have default accessibility attributes from HTML 5, but it is still not possible with custom elements. After that, I wanted to make a change in the way how this library should work and wanted to find a way to provide more support to the developer by also integrating JavaScript functions for updating the aria attributes in the library. This was the next step in the development process. Then after the first aria function is made, which was the button aria role and also the button pressed aria attribute. Then I tested the button component from the main web page, integrating this library function. I used `className` attribute as a selector to point the library that this specific button function should be called on that specific HTML element. When the tests were passed without any aria issues, I moved onto the next component to implement, which was the toggle button. One by one different components are implemented and verified that the overall accessibility is good. This was the way of working I choose for this project.

When the development of the basic roles and attributes in the library was finished, it was time to check how the library works in a live environment. For testing the library I choose different components and went with testing individual components rather than the whole page. I tested the same component with different testing tools available and made sure that the library is doing what it was indented to do. After the individual component testing, the components were added to a single web page and tested again to check if this makes a difference in the accessibility of the components. I found out that

it may happen that as a whole the page may have some accessibility concerns regarding parent or child HTML tag but the components did not show any aria accessibility errors.

The need for more accessible web content is more than ever now. As the opportunities in the web is increasing day by day, more people want to access all these information on the web. It is the duty of the developer to provide better accessible content to the end user. New laws are being passed by different governmental bodies to ensure accessibility of the web pages. There was also a increase in the number accessibility related law-suits over the years. According to UsableNet[1] research team, in 2017 the number of accessibility related law-suits in United States were 814. But in 2018 this number jumped to 2285, an increase of 181% in just a year. In future this number will keep increasing unless we ensure that the contents on web is accessible for everyone.

As a conclusion, I can now say, the most interesting part about my research was learning more about web accessibility, the importance of it in modern day web development, finding a way to implement the library and successfully integrate it into a live project. In future, this should support developers focusing on their core jobs by taking much of the accessibility responsibilities off their hands. To sum everything up, the outcome is quite the initially expected extent; a proof of concept, that the aria attributes and update functions used this way helps with the faster development of more accessible web pages.

[1] https://blog.usablenet.com/2018-ada-web-accessibility-lawsuit-recap-report

Appendix A

Contents of the DVD

Format: DVD-ROM, Single Layer, UDF-Format

A.1 PDF-Files

Path: /

S1710629003_Anitha_Gregory_Thesis.pdf Master's Thesis with instructions (entire document)

Path: /online

Accessibility evaluation tools.pdf
Accessibility Insights overview.pdf
Accessibility Insights for web Fastpass.pdf
Accessibility introduction.pdf
Accessible Brand Colors.pdf
Accessible Brand Colors Check.pdf
American Disability Act section 508.pdf
Assessment in Accessibility Insights for Web.pdf
rollup_js babel guide.pdf
Barrier-Free Information Technology.pdf
browser market share.pdf
Business Case for Digital Accessibility
Chrome extension API.pdf
Chrome extensions overview.pdf
Color Contrast Accessibility Validator.pdf
Disability and Health Overview CDC.pdf
Disability And Types.pdf
react_js.pdf . . . . . .
WAI-ARIA basics.pdf .

WAI-ARIA Browser Support.pdf
WAI-ARIA Screen reader compatibility.pdf
WCAG 2.1 Future.pdf .
Evaluation tools selection.pdf
Web Content Accessibility Guidelines 2.1.pdf
Web Accessibility Evaluation Tools List.pdf
Web Accessibility Initiative.pdf
webaim web accessibility introduction.pdf

A.2 Source Code

Path: /source

v1.0.0.zip . . . . . Source code of the project

A.3 Graphics

Path: /images

*.png . . . . . Rendered images and Screenshots

References

Literature

[1] T. Acosta, P. Acosta-Vargas, and S. Lujan-Mora. "Accessibility of eGovernment Services in Latin America". In: *Proceedings of the 2018 Fifth International Conference on eDemocracy & eGovernment (ICEDEG)*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2018, pp. 67–74 (cit. on p. 13).

[2] P. Acosta-Vargas, T. Acosta, and S. Lujan-Mora. "Framework for Accessibility Evaluation of Hospital Websites". In: *Proceedings of the 2018 Fifth International Conference on eDemocracy & eGovernment (ICEDEG)*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2018, pp. 9–15 (cit. on p. 4).

[3] S. U. Dongaonkar, R. S. Vadali, and C. Dhutadmal. "Accessibility Analyzer: Tool for New Adaptations in Government Web Applications to Improve Accessibility". In: *Proceedings of the 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. Aug. 2017, pp. 1–5 (cit. on pp. 13, 22).

[4] T. Halbach. "Towards Cognitively Accessible Web Pages". In: *Proceedings of the 2010 Third International Conference on Advances in Computer-Human Interactions*. Feb. 2010, pp. 19–24 (cit. on p. 4).

[5] W. A. R. W. M. Isa et al. "Accessibility evaluation using Web Content Accessibility Guidelines (WCAG) 2.0". In: *Proceedings of the 2016 4th International Conference on User Science and Engineering (i-USEr)*. Aug. 2016, pp. 1–4 (cit. on p. 22).

[6] A. Ismail and K. S. Kuppusamy. "Accessibility analysis of North Eastern India Region websites for persons with disabilities". In: *Proceedings of the 2016 International Conference on Accessibility to Digital World (ICADW)*. Dec. 2016, pp. 145–148 (cit. on p. 13).

[7] V. Okanovic. "Web application development with component frameworks". In: *Proceedings of the 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. May 2014, pp. 889–892 (cit. on p. 36).

[8] M. Tollefsen and T. Ausland. "A practitioner's approach to using WCAG evaluation tools". In: *Proceedings of the 2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA)*. Dec. 2017, pp. 1–5 (cit. on p. 22).

[9] Xabier Valencia et al. “User individuality management in websites based on WAI-ARIA annotations and ontologies”. In: *Proceedings of the W4A 2013 - International Cross-Disciplinary Conference on Web Accessibility*. May 2013, 29:1–29:10 (cit. on p. 4).

[10] K. Wille, R. R. Dumke, and C. Wille. “Measuring the Accessability Based on Web Content Accessibility Guidelines”. In: *Proceedings of the 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. Oct. 2016, pp. 164–169 (cit. on p. 15).

[11] E. Wittern, P. Suter, and S. Rajagopalan. “A Look at the Dynamics of the JavaScript Package Ecosystem”. In: *Proceedings of the 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. May 2016, pp. 351–361 (cit. on p. 39).

Online sources

[12] *Accessibility Evaluation Tools - Evaluation*. URL: https://webaim.org/articles/tools/#eval_repair (cit. on p. 16).

[13] *Accessibility Evaluation Tools - Reports*. URL: https://webaim.org/articles/tools/#report (cit. on p. 16).

[14] *Accessibility Insights for Web*. URL: https://accessibilityinsights.io/docs/en/web/overview (cit. on p. 18).

[15] *Accessibility Insights for Web extension*. URL: https://accessibilityinsights.io/docs/en/web/getstarted/fastpass (cit. on p. 20).

[16] *Accessibility Insights for Web extension Automated Checks*. URL: https://accessibilityinsights.io/docs/en/web/getstarted/fastpass#run-the-automated-checks (cit. on p. 20).

[17] *Accessibility Intro*. URL: https://www.w3.org/WAI/fundamentals/accessibility-intro/ (cit. on pp. 4, 11).

[18] *Accessible Brand Colors*. URL: https://abc.useallfive.com/ (cit. on p. 18).

[19] *Accessible Brand Colors Check*. URL: https://abc.useallfive.com/?colors[]=112233,190303 (cit. on p. 19).

[20] *American Disability Act section 508*. URL: https://www.justice.gov/crt/pl-105-220-1998-hr-1385-pl-105-220-enacted-august-7-1998-112-stat-936-codified-section-504 (cit. on p. 13).

[21] *Aria Button Authoring Practices*. URL: https://www.w3.org/TR/wai-aria-practices-1.1/examples/button/button.html (cit. on p. 29).

[22] *Assessment in Accessibility Insights for Web*. URL: https://accessibilityinsights.io/docs/en/web/getstarted/assessment (cit. on p. 18).

[23] *Babel*. URL: https://rollupjs.org/guide/en/%5C#babel (cit. on p. 35).

[24] *BITV*. URL: https://www.einfach-fuer-alle.de/artikel/bitv_english/ (cit. on p. 13).

[25] *Browser Market Share*. URL: http://gs.statcounter.com/browser-market-share (cit. on p. 33).

[26] *Business Case*. URL: https://www.w3.org/WAI/business-case/#groups (cit. on p. 5).

[27] *Chrome Extension Introduction*. URL: https://developer.chrome.com/extensions/extension (cit. on p. 39).

[28] *Chrome Extension Overview*. URL: https://developer.chrome.com/extensions/overview (cit. on p. 27).

[29] *Color Contrast Accessibility Validator Tool*. URL: https://color.a11y.com/ (cit. on p. 17).

[30] *Commercial Evaluation Tools*. URL: https://webaim.org/articles/tools/#free (cit. on p. 14).

[31] *Disability and Health Overview*. URL: https://www.cdc.gov/ncbddd/disabilityandhealth/disability.html (cit. on p. 3).

[32] *Evaluation Tools*. URL: https://webaim.org/articles/tools/#platform (cit. on p. 15).

[33] *Evaluation Tools classification*. URL: https://webaim.org/articles/tools/#stand_guide (cit. on p. 14).

[34] *Modes of Disability*. URL: https://medium.com/fbdevclagos/why-web-accessibility-is-important-and-how-you-can-accomplish-it-4f59fda7859c (cit. on pp. 3, 4, 11).

[35] *React - A JavaScript Library for Building User Interfaces*. URL: https://reactjs.org/ (cit. on p. 37).

[36] *Scope of Accessibility Evaluation Tools*. URL: https://webaim.org/articles/tools/#scope (cit. on p. 15).

[37] *WAI-ARIA Basic Tackling Problems*. URL: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics#A_whole_new_set_of_problems (cit. on p. 5).

[38] *WAI-ARIA Basics*. URL: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics (cit. on pp. 9, 11).

[39] *WAI-ARIA Browser Support*. URL: https://caniuse.com/#feat=wai-aria (cit. on p. 6).

[40] *WAI-ARIA Main Features*. URL: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics#Enter_WAI-ARIA (cit. on p. 5).

[41] *WAI-ARIA Screen reader compatibility*. URL: https://www.powermapper.com/tests/screen-readers/aria/ (cit. on p. 7).

[42] *WAI-ARIA Usability*. URL: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics#When_should_you_use_WAI-ARIA (cit. on pp. 8–10).

[43] *WCAG 2.1 Future*. URL: https://www.w3.org/TR/WCAG21/#later-versions-of-accessibility-guidelines (cit. on p. 12).

[44] *WCAG Tools Selection*. URL: https://www.w3.org/WAI/test-evaluate/tools/selecting/ (cit. on p. 14).

[45] *WCAG2.1*. URL: https://www.w3.org/TR/WCAG21/ (cit. on p. 14).

[46] *WCAG2.1 Color Contrast Accessibility Validator*. URL: https://www.w3.org/WAI/ER/tools/?q=wcag-21-w3c-web-content-accessibility-guidelines-21#a11y-color-contrast-accessibility-validator (cit. on p. 17).

[47] *Web Accessibility Initiative*. URL: https://www.w3.org/WAI/ (cit. on p. 13).

[48] *Web Content Accessibility Guidelines Background*. URL: https://www.w3.org/TR/WCAG21/#background-on-wcag-2 (cit. on p. 11).

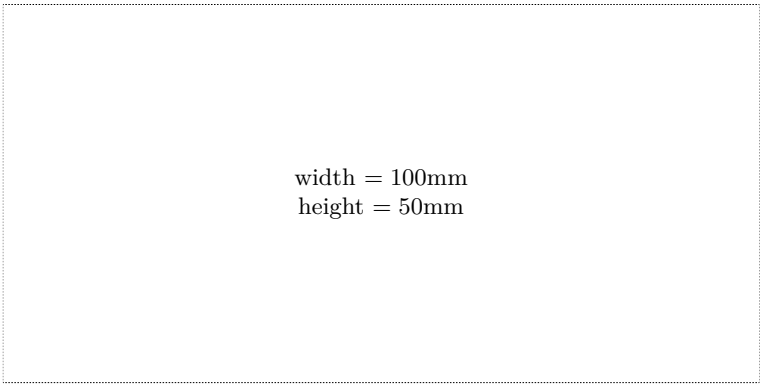
[49] *Web Content Accessibility Guidelines Layers*. URL: https://www.w3.org/TR/WCAG21/#wcag-2-layers-of-guidance (cit. on p. 11).

[50] *Web Content Accessibility Guidelines Requirements*. URL: https://www.w3.org/TR/WCAG21/#requirements-for-wcag-2-1 (cit. on p. 12).

[51] *Webaim Intro*. URL: https://webaim.org/intro/ (cit. on p. 4).

Check Final Print Size

— Check final print size! —



— Remove this page after printing! —