Università degli Studi di Bologna

Scuola di scienze

Dipartimento di Informatica

Relatore: Prof. Fabio Vitali
Correlatori: Dott.ssa Caterina Morelli, Dott. Vincenzo Rubano

Sessione: III
Anno Accademico: 2022/2023

Raffinamento e Ampliamento di SAHARIAN: Un Contributo allo Sviluppo di Strumenti per l'Accessibilità Web

Anselmo Ferrari

Abstract

Nell'ambito dello sviluppo web, l'accessibilità emerge come una priorità imprescindibile, mirando a garantire che i contenuti digitali siano fruibili da tutti gli utenti, inclusi coloro che si avvalgono di tecnologie assistive. Questa tesi esplora il processo di raffinamento e ampliamento di SAHARIAN, uno strumento sviluppato per facilitare la valutazione dell'accessibilità delle pagine web. Attraverso un'analisi dettagliata, il lavoro si propone di identificare e implementare miglioramenti significativi in SAHARIAN, con l'obiettivo di estendere le sue capacità di analisi e di rendere più efficace il suo contributo nell'ambito dell'accessibilità web.

La ricerca si articola in una serie di fasi operative, a partire dall'analisi delle funzionalità esistenti di SAHARIAN, passando per l'identificazione delle aree suscettibili di miglioramento, fino all'effettiva implementazione di nuove funzionalità e alla verifica della loro efficacia. Un focus particolare viene posto sull'integrazione degli attributi ARIA non ancora supportati da SAHARIAN, considerati fondamentali per la corretta interpretazione dei contenuti da parte delle tecnologie assistive. Inoltre, si esplorano strategie per ottimizzare le prestazioni di SAHARIAN nell'elaborazione di pagine web complesse, affrontando la sfida rappresentata dal tempo di elaborazione necessario per generare versioni accessibili di siti densi di contenuti.

L'approccio adottato nella tesi è fortemente pratico e si basa su un processo iterativo di test e valutazione, attraverso il quale vengono gradualmente introdotte e testate le nuove funzionalità. La validazione delle modifiche apportate a SAHARIAN avviene mediante l'utilizzo di scenari reali di navigazione web, permettendo di valutare l'impatto degli interventi sul miglioramento dell'esperienza utente per persone con disabilità.

Il contributo di questo lavoro di tesi non si limita all'ampliamento delle funzionalità di SAHARIAN, ma si estende alla sensibilizzazione della comunità degli sviluppatori sull'importanza dell'accessibilità web.

In conclusione, il raffinamento e l'ampliamento di SAHARIAN rappresentano un passo avanti significativo nel campo dell'accessibilità web, offrendo agli sviluppatori uno strumento più potente e versatile per garantire che i loro siti siano accessibili a un'ampia gamma di utenti, indipendentemente dalle loro esigenze specifiche.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'accesso alle informazioni è un diritto fondamentale nell'era in cui viviamo. Il web, con la sua abbondanza di dati e servizi, è diventato un pilastro della vita quotidiana per molti. Tuttavia, non tutti possono navigare e utilizzare il web allo stesso modo. Le persone con disabilità visive incontrano ostacoli significativi che limitano la loro partecipazione digitale. Questa tesi affronta il problema dell'accessibilità web, con particolare attenzione alle sfide incontrate dagli utenti non vedenti.

Il problema dell'accessibilità web è ampio e multisfacettato, interessando vari aspetti della progettazione e dello sviluppo di siti web. L'ambito del problema include questioni tecniche, come la conformità agli standard di accessibilità, e questioni pratiche, come la formazione degli sviluppatori e la sensibilizzazione sulle necessità degli utenti con disabilità. Questa tesi si inserisce nel dibattito scientifico e tecnologico attuale, che vede una lacuna tra le linee guida di accessibilità e la loro implementazione pratica nei siti web. Infatti, mentre il consenso sull'importanza dell'accessibilità web cresce, gli strumenti che facilitano tale pratica rimangono complessi e spesso non intuitivi per gli sviluppatori.

Questo lavoro estende le ricerche e le implementazioni presentate in precedenza in ambito di accessibilità web, SAHARIAN è inizialmente concepito dal Prof. Fabio Vitali insieme al Dott. Vincenzo Rubano, è stato oggetto di un'importante fase di sviluppo e implementazione che ha segnato il percorso evolutivo dello strumento nel campo dell'accessibilità web. Successivamente, Luca Morosini ha esteso il lavoro originale attraverso la sua tesi, apportando significativi contributi all'approfondimento delle funzionalità di SAHARIAN, in "SahARIAn: Uno strumento visuale per la progettazione di pagine web accessibili" [Mor18], e successivamente approfondite in "On Making Web Accessibility More Accessible: Strategy and Tools for Social Good" di Vincenzo Rubano del 2022/2023 [Rub23]. Quest'ultima ricerca, oltre a portare avanti lo sviluppo di SAHARIAN, ha incluso l'esplorazione di altri strumenti dedicati all'accessibilità, come l'AX framework e A11A, contribuendo significativamente alla comprensione e alla diffusione delle migliori pratiche nel campo dell'accessibilità web.

In tale contesto, la presente tesi mira ad affinare ulteriormente e ampliare le funzionalità di SAHARIAN, capitalizzando sui solidi fondamenti posti dai lavori di Vitali, Rubano e Morosini. SAHARIAN viene qui presentato come uno strumento avanzato per l'analisi dell'accessibilità delle pagine web, che mediante la rimozione degli stili CSS, offre agli sviluppatori una rappresentazione semplificata del contenuto, così come percepito dalle tecnologie assistive. Questo approccio permette di identificare e correggere gli errori di accessibilità con maggiore efficacia, puntando a una progettazione web che sia inclusiva fin dalle sue fondamenta.

Il valore aggiunto di SAHARIAN è stato verificato attraverso un processo di va-

lutazione che ha coinvolto attivamente gli sviluppatori web nell'uso dello strumento. Il feedback raccolto ha sottolineato l'importanza di SAHARIAN nel rendere l'accessibilità un traguardo più facilmente raggiungibile e nel migliorare complessivamente l'esperienza degli utenti non vedenti sul web.

Per ulteriori dettagli e riferimenti bibliografici relativi ai lavori che hanno gettato le basi per questo studio, si rimanda ai documenti originali delle tesi di Morosini e Rubano.

Nel contesto dell'accessibilità digitale, è essenziale menzionare l'European Accessibility Act (EAA)[Acc23], una direttiva dell'Unione Europea entrata in vigore nell'aprile 2019, con l'obiettivo di migliorare la commercializzazione di prodotti e servizi accessibili rimuovendo le normative specifiche dei singoli paesi. Beneficiando di un insieme comune di regole, le aziende possono facilmente partecipare al commercio transfrontaliero, contribuendo a un mercato più ampio per i prodotti e servizi accessibili. L'EAA, inoltre, mira a garantire che le persone con disabilità e gli anziani possano beneficiare di prodotti e servizi più accessibili sul mercato, con un conseguente aumento della competitività dei prezzi e una riduzione delle barriere all'interno dell'UE. Originariamente proposta nel 2011, questa direttiva si propone di complementare la Web Accessibility Directive dell'UE, diventata legge nel 2016, e riflette gli obblighi della Convenzione delle Nazioni Unite sui Diritti delle Persone con Disabilità, coprendo un ampio spettro di sistemi inclusi dispositivi personali e servizi pubblici

Nell'attuale panorama digitale, l'accessibilità web è diventata un obbligo giuridico oltre che una necessità etica, soprattutto alla luce dell'entrata in vigore dell'European Accessibility Act (EAA) [Eur23]. Questa direttiva, mirando a stabilire standard minimi per l'accessibilità dei prodotti e servizi nell'Unione Europea, sottolinea l'importanza di rendere i contenuti digitali accessibili a tutti gli utenti, indipendentemente dalle loro capacità fisiche. La tesi di questa ricerca si colloca nel contesto di questa sfida in evoluzione, esaminando in modo critico gli standard e gli strumenti esistenti di accessibilità web, con un focus particolare su WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) e WCAG (Web Content Accessibility Guidelines).

Questi standard hanno segnato passi significativi verso la realizzazione di un ambiente digitale inclusivo. Tuttavia, nonostante i loro meriti, presentano limitazioni intrinseche che possono ostacolare la piena realizzazione di un web accessibile. Problemi come la complessità di implementazione, la variabilità del supporto nei diversi browser e dispositivi, l'ambiguità delle linee guida, e la lentezza nell'aggiornamento degli standard, sono solo alcune delle sfide che gli sviluppatori devono affrontare.

Il progetto Saharian si propone di affrontare queste sfide, fornendo uno strumento che aiuta gli sviluppatori a visualizzare e comprendere come i vari dispositivi di accessibilità interpretano le pagine web. Questo approccio permette di identificare rapidamente le aree che necessitano di miglioramenti, facilitando la creazione di siti web più accessibili e inclusivi.

Questa tesi mira a esplorare in profondità le limitazioni degli approcci attuali all'accessibilità web, ponendo le basi per la comprensione del valore aggiunto e dell'innovazione che Saharian intende apportare in questo ambito cruciale.

Questa tesi inizialmente indaga l'attuale stato dell'arte dell'accessibilità web, iniziando con una panoramica sugli aspetti fondamentali quali gli standard ARIA e il loro impiego all'interno di SAHARIAN. Si procederà poi con l'esame comparativo di vari strumenti dedicati al controllo dell'accessibilità, mettendo in luce punti di forza e cri-

ticità. L'obiettivo è fornire una comprensione approfondita delle sfide e delle soluzioni nel campo dell'accessibilità web, evidenziando come SAHARIAN e altri strumenti si posizionano in questo panorama.

Nelle fasi successive, questa tesi approfondisce l'utilizzo di SAHARIAN e la sua architettura, esaminando applicazioni pratiche su pagine web esistenti e campioni creati ad hoc. Attraverso esempi dettagliati, viene esposto il potenziale di SAHARIAN nell'analisi dell'accessibilità web, identificando contemporaneamente i punti deboli dello strumento. Viene fornita una discussione approfondita su come questi limiti possano essere affrontati e superati, includendo una lista precisa degli attributi ARIA da integrare per potenziare ulteriormente le capacità di SAHARIAN nel promuovere lo sviluppo di pagine web accessibili.

L'analisi dell'architettura di SAHARIAN offre uno sguardo approfondito sul suo funzionamento interno, illustrando come modifica le pagine web per migliorarne l'accessibilità. Questo segmento descrive dettagliatamente l'interazione di SAHARIAN con il Document Object Model (DOM) e come utilizza gli attributi ARIA, insieme ad altri elementi chiave, per costruire una versione "pulita" della pagina web. Tale esame è cruciale per i futuri sviluppatori interessati ad espandere SAHARIAN, fornendo loro le basi per un accesso semplificato e comprensivo al codice sorgente e alle sue funzionalità.

La sezione finale della tesi è dedicata all'analisi dei test condotti su SAHARIAN, descrivendo le metodologie adottate e i risultati ottenuti. Viene evidenziato come SAHARIAN abbia significativamente aumentato il numero di errori di accessibilità individuati e ridotto il tempo necessario per la loro identificazione. Questa parte dimostra l'efficacia di SAHARIAN nel supportare gli sviluppatori, evidenziando il suo valore attuale come strumento di assistenza nello sviluppo di pagine web più accessibili e nell'ottimizzazione del processo di verifica dell'accessibilità.

Capitolo 1

L'accessibilità sul web: risultati e limiti

1.1 Analisi degli approcci precedenti all'accessibilità web

Nel campo dell'accessibilità web, gli approcci adottati finora hanno mostrato notevoli progressi ma anche limitazioni significative che necessitano di un'analisi critica. Tradizionalmente, gli sforzi si sono concentrati sulla conformità a standard come le Web Content Accessibility Guidelines (WCAG) [Wor18, Acca, Accb], che forniscono un framework comprensivo per rendere i contenuti web accessibili a persone con varie disabilità. Tuttavia, la mera adesione a tali linee guida non sempre si traduce in un'esperienza utente realmente accessibile e intuitiva, specialmente per gli utenti con disabilità visive.

Un limite fondamentale degli approcci precedenti è la dipendenza eccessiva da controlli manuali e revisioni periodiche. Questo approccio reattivo tende a ignorare il dinamismo dei contenuti web, con il rischio che le pagine diventino rapidamente obsolete in termini di accessibilità non appena vengono aggiornate o modificate. Inoltre, la verifica manuale dell'accessibilità è spesso dispendiosa in termini di tempo e risorse, rendendola impraticabile su larga scala.

Un altro aspetto critico riguarda la comprensione e l'interpretazione delle linee guida WCAG stesse. La loro natura a volte astratta e la vasta gamma di criteri possono essere fonti di confusione per gli sviluppatori web, specialmente per coloro che non hanno esperienza specifica nell'accessibilità. Di conseguenza, molte implementazioni di accessibilità web sono incomplete o mal interpretate, portando a soluzioni che adempiono ai requisiti minimi senza realmente migliorare l'esperienza per le persone con disabilità.

Gli strumenti per testare l'accessibilità web sono essenziali per garantire che i siti rispettino le linee guida e siano fruibili da tutti gli utenti, inclusi quelli con disabilità. Sebbene questi strumenti offrano una valutazione preziosa, spesso si concentrano su aspetti tecnici, trascurando la dimensione dell'usabilità e dell'esperienza utente. Tuttavia, esistono diversi strumenti che, pur con i loro limiti, aiutano gli sviluppatori e i designer a identificare e correggere problemi di accessibilità. Ecco alcuni dei più utilizzati:

- **AXE Browser Extension** [U.S] Integrabile direttamente nei browser come Chrome o Firefox, AXE può analizzare rapidamente le pagine web per identificare violazioni delle WCAG (Web Content Accessibility Guidelines). Fornisce dettagli sugli errori rilevati e suggerimenti su come risolverli, ma si concentra principalmente su aspetti tecnici come l'uso corretto dei tag semantici e le alternative testuali per le immagini.

- **Wave Tool** [Web] Un altro strumento utile è Wave, che offre una visione grafica delle potenziali problematiche di accessibilità direttamente nella pagina web testata. Wave evidenzia errori, avvertimenti e caratteristiche accessibili della pagina, permettendo agli sviluppatori di vedere dove sono necessari miglioramenti. Anche Wave, tuttavia, ha i suoi limiti nell'analizzare la comprensibilità del testo o l'efficacia della navigazione per gli utenti con disabilità.

- **JAWS (Job Access With Speech)**[Hara] Si tratta di un lettore di schermo molto usato da utenti non vedenti o ipovedenti. JAWS permette di navigare il web attraverso comandi vocali e legge ad alta voce i contenuti delle pagine. Questo strumento è utile per testare l'esperienza utente di persone che utilizzano lettori di schermo, ma richiede familiarità con l'uso del software per interpretare i risultati efficacemente.

- **NVDA (NonVisual Desktop Access)**[Harb] Un lettore di schermo gratuito e open-source che funziona su Windows. Simile a JAWS, NVDA aiuta a comprendere come gli utenti non vedenti interagiscono con i siti web. Offre una prospettiva sull'accessibilità delle pagine web dal punto di vista dell'utente che fa affidamento su lettori di schermo, ma, come JAWS, richiede conoscenza specifica per l'uso efficace.

- **Google Lighthouse**[Goo] Uno strumento di analisi integrato in Chrome DevTools, Lighthouse fornisce un report sull'accessibilità di una pagina, oltre a metriche su prestazioni, SEO e best practices. Anche se offre consigli utili su come migliorare l'accessibilità, le sue valutazioni possono non coprire completamente l'esperienza utente o l'usabilità da un punto di vista olistico.

Infine, c'è una mancanza di integrazione tra la progettazione dell'accessibilità e lo sviluppo web regolare. Spesso, l'accessibilità viene vista come un'aggiunta postuma piuttosto che come una componente integrata del processo di sviluppo. Questo approccio segmentato porta a soluzioni che sono appiccicate 'sopra' l'interfaccia esistente, piuttosto che essere incorporate organicamente nella progettazione del sito web.

In sintesi, mentre gli approcci esistenti all'accessibilità web hanno gettato le basi per un web più inclusivo, esistono limitazioni sostanziali che impediscono la piena realizzazione di questo obiettivo. Una riflessione critica su queste limitazioni è fondamentale per guidare lo sviluppo di nuove soluzioni più efficaci, efficienti e integrate, come quelle proposte in questa tesi con lo strumento SAHARIAN.

1.2 Presentazione delle tecnologie che hanno influenzato SAHARIAN

1.2.1 Gli standard ARIA

L'Accessible Rich Internet Applications (ARIA) è uno standard cruciale nel panorama dell'accessibilità web, il cui ruolo è diventato sempre più rilevante nel contesto delle applicazioni web moderne. Gli standard ARIA sono stati sviluppati dal Web Accessibility Initiative (WAI) del World Wide Web Consortium (W3C)[Wor23], con l'obiettivo di migliorare l'accessibilità dei contenuti web dinamici e delle applicazioni web complesse per persone con disabilità. In particolare, ARIA fornisce un insieme di attributi specializzati che possono essere aggiunti al codice HTML, migliorando significativamente l'interazione tra il contenuto web e le tecnologie assistive, come i lettori di schermo. Uno degli aspetti fondamentali di ARIA è la sua capacità di descrivere elementi dell'interfaccia utente che non sono facilmente rappresentabili con il solo HTML standard. Ad esempio, elementi di interazione dinamica come dropdown, menù a tendina, barre di navigazione e altri componenti interattivi possono essere difficili da interpretare per le tecnologie assistive. ARIA interviene fornendo attributi aggiuntivi che descrivono il ruolo, lo stato e le proprietà di tali elementi, consentendo ai lettori di schermo di comunicare queste informazioni agli utenti in modo chiaro e comprensibile. ARIA introduce anche concetti come le "landmark regions", che aiutano gli utenti di tecnologie assistive a navigare più facilmente in una pagina, identificando rapidamente le sezioni principali come intestazione, piè di pagina, navigazione principale e contenuto principale. Questo miglioramento nella strutturazione delle pagine web rende l'esperienza utente molto più fluida per persone con disabilità visive.

Tuttavia, l'efficacia di ARIA dipende fortemente dalla sua corretta implementazione. Un uso improprio degli attributi ARIA può infatti portare a maggiore confusione e ostacolare l'accessibilità piuttosto che migliorarla. È quindi cruciale che gli sviluppatori web comprendano e applichino gli standard ARIA con precisione e attenzione. Questo include la necessità di mantenerli aggiornati con le ultime versioni dello standard, poiché l'evoluzione delle tecnologie web e delle tecniche di accessibilità porta continuamente a nuove best practice e raccomandazioni.

In questo contesto, SAHARIAN gioca un ruolo fondamentale nell'assistere gli sviluppatori nell'implementazione corretta di ARIA. Attraverso l'analisi automatica delle pagine web, SAHARIAN è in grado di rilevare l'uso improprio degli attributi ARIA e suggerire correzioni o miglioramenti, contribuendo così a un web più accessibile e inclusivo.

Gli standard ARIA, con la loro vasta gamma di attributi, offrono agli sviluppatori web strumenti potenti per migliorare l'accessibilità dei loro siti. Tra i più utilizzati vi sono gli attributi `role`, che definiscono il ruolo specifico di un elemento nella pagina. Ad esempio, un elemento può avere un role="button", indicando che funziona come un pulsante, o role="navigation", segnalando che l'elemento è parte della navigazione del sito. Questi ruoli sono fondamentali per le tecnologie assistive, poiché forniscono un contesto chiaro sulla funzione degli elementi, migliorando l'orientamento e l'interazione per gli utenti con disabilità visive.

Un altro aspetto importante è l'uso degli attributi aria-label e aria-labelledby. Questi attributi permettono di fornire una descrizione testuale degli elementi, soprattutto

quando il testo visibile non è sufficiente per comprendere il contesto o la funzione di un elemento. Per esempio, un'icona di una lente di ingrandimento usata per la ricerca potrebbe avere un aria-label="search", rendendo chiaro lo scopo dell'icona anche per chi non può vederla.

In aggiunta, gli attributi aria-hidden e aria-live sono cruciali per gestire la visibilità e l'aggiornamento dinamico dei contenuti. aria-hidden="true" può essere usato per nascondere elementi visivi che possono creare confusione o essere irrilevanti per gli utenti di lettori di schermo, mentre aria-live è utilizzato per indicare come e quando gli utenti devono essere informati di aggiornamenti dinamici del contenuto, come i messaggi di stato o le notifiche.

Il tabindex è un altro strumento essenziale per la navigabilità. Permette di controllare l'ordine in cui gli elementi ricevono il focus quando si naviga tramite tastiera. Un tabindex="0" assicura che l'elemento sia raggiungibile tramite la navigazione con tabulazione, mentre tabindex="-1" può essere usato per rendere un elemento focalizzabile programmabilmente ma non tramite la navigazione con tabulazione. Questo è particolarmente utile per creare un flusso di navigazione logico e intuitivo, soprattutto in applicazioni web complesse. Tuttavia, è importante notare che un uso eccessivo o improprio di ARIA e tabindex può complicare o addirittura peggiorare l'esperienza utente. Ad esempio, un tabindex mal gestito può creare un ordine di navigazione confuso, mentre un uso eccessivo di aria-label può sovraccaricare l'utente con informazioni non necessarie. Quindi, è fondamentale che gli sviluppatori adottino un approccio ponderato e testino le loro implementazioni con utenti reali.

In conclusione, gli standard ARIA rappresentano un tassello essenziale nell'ecosistema dell'accessibilità web. Il loro impiego consente di superare numerose sfide poste dalle applicazioni web moderne, garantendo che i contenuti web siano accessibili a un'ampia gamma di utenti, indipendentemente dalle loro abilità. La comprensione e l'adozione corretta di questi standard sono quindi imperativi per sviluppatori web che aspirano a creare esperienze digitali veramente inclusive.

1.2.2 Utilizzo degli standard ARIA da parte di SAHARIAN

SAHARIAN è un avanzato strumento per l'accessibilità web che sfrutta in modo innovativo gli standard ARIA (Accessible Rich Internet Applications) per migliorare l'esperienza utente su pagine web. Al cuore della sua funzionalità vi è un processo sofisticato che si attiva all'avvio dell'estensione: innanzitutto, viene eliminata ogni forma di CSS preesistente sulla pagina web in esame. Questo permette di rimuovere tutte le personalizzazioni stilistiche che potrebbero mascherare o alterare la struttura semantica e l'accessibilità del contenuto. In sostituzione, viene applicato un set di regole CSS specificamente progettato da SAHARIAN, il quale ha il compito di rivelare la struttura accessibile sottostante la pagina web.

Il cuore dell'approccio di SAHARIAN è un'analisi dettagliata del Document Object Model (DOM) della pagina. Durante questo processo, lo strumento esegue una scansione completa degli elementi del DOM, identificando non solo gli elementi HTML standard come input, button, immagini e ancoraggi, ma anche elementi più complessi e personalizzati. Per questi ultimi, SAHARIAN verifica con attenzione la presenza di attributi ARIA pertinenti, che sono fondamentali per garantire l'accessibilità dei contenuti non standard. Uno degli aspetti cruciali di questa verifica è l'attributo "role" di

ARIA, che SAHARIAN utilizza per determinare come ciascun elemento debba essere reso e interpretato dal punto di vista dell'accessibilità.

Oltre al "role", SAHARIAN presta particolare attenzione anche ad altri attributi ARIA come "aria-label", "aria-expanded" e "aria-selected". Questi attributi giocano un ruolo chiave nel definire ulteriormente la semantica e lo stato degli elementi della pagina. Ad esempio, "aria-expanded" e "aria-selected" sono usati per indicare lo stato dinamico di un elemento, come un menu a discesa o un elemento di selezione. SAHARIAN utilizza queste informazioni per applicare un CSS specifico che rende questi stati visivamente distinti e più facilmente comprensibili.

In questo modo, SAHARIAN non solo conforma la pagina web agli standard di accessibilità, ma trasforma anche visivamente la pagina in modo che gli sviluppatori possano comprendere intuitivamente come le modifiche apportate influenzino l'accessibilità. Questo approccio incentrato sulla visualizzazione aiuta a colmare il divario tra le intenzioni di design e l'effettiva esperienza utente.

SAHARIAN non si limita soltanto a migliorare la visualizzazione grafica delle pagine web per renderle più accessibili, sfrutta in modo efficace anche gli attributi ARIA per emulare l'interattività da tastiera e il dinamismo generale della pagina. Un esempio significativo di questa funzionalità è rappresentato dalla funzione "convertMouseEvent", la quale svolge un ruolo cruciale nell'assicurare che gli elementi interattivi siano accessibili tramite tastiera. Questa funzione analizza gli elementi che invocano la funzione, verificando il loro ruolo e assicurandosi che siano tipi di elementi che ci si aspetta siano interattivi, come 'radio', 'tab' o 'option'. Inoltre, controlla che il tabindex sia impostato correttamente, consentendo l'azione solo se questi criteri sono soddisfatti. Questo meccanismo assicura che gli utenti possano navigare e interagire con la pagina web utilizzando solo la tastiera, una caratteristica fondamentale per l'accessibilità.

SAHARIAN estende questo livello di controllo anche agli elementi HTML standard come pulsanti e link, garantendo che la loro interattività sia coerente e accessibile. Questo approccio integrato permette a SAHARIAN di garantire che l'interfaccia utente della pagina web sia non solo visivamente accessibile, ma anche pienamente funzionale per gli utenti che dipendono dalla navigazione da tastiera.

In aggiunta, SAHARIAN implementa controlli sofisticati sugli attributi aria-live e aria-atomic. Questi attributi sono fondamentali per gestire i contenuti dinamici e le loro modifiche all'interno delle pagine web. SAHARIAN esegue un monitoraggio costante della pagina, e quando rileva modifiche in sezioni che utilizzano aria-live, invia notifiche all'utente per segnalare la modifica. Questa funzionalità è particolarmente utile per gli utenti con disabilità visive, poiché consente loro di essere immediatamente informati su aggiornamenti o cambiamenti importanti nel contenuto, migliorando l'esperienza utente e assicurando che le informazioni rilevanti non vengano perse.

In conclusione, SAHARIAN utilizza gli attributi ARIA non solo per ottimizzare la presentazione visiva delle pagine web, ma anche per migliorare significativamente l'interattività e il dinamismo della pagina. Attraverso l'emulazione dell'interattività da tastiera e il monitoraggio attento delle modifiche dinamiche dei contenuti, SAHARIAN si afferma come uno strumento completo per garantire che le pagine web siano accessibili e facilmente navigabili per tutti gli utenti.

Capitolo 2

SAHARIAN: uno strumento per progettare applicazioni accessibili

2.1 Descrizione funzionale di SAHARIAN: obiettivi e funzionalità principali

SAHARIAN è un'innovativa estensione per Google Chrome, concepita inizialmente dal Prof. Fabio Vitali e dal Dott. Vincenzo Rubano[Rub23], per promuovere l'accessibilità nel web. Questo strumento è stato sviluppato per assistere gli sviluppatori web nell'identificare e risolvere problemi di accessibilità, attraverso una rappresentazione visuale intuitiva che evidenzia le aree critiche influenzanti l'esperienza utente di persone con disabilità. Successivamente esteso nella tesi di Luca Morosini[Mor18], il lavoro presente prosegue su questa traiettoria, mirando a perfezionare e ampliare ulteriormente le funzionalità di SAHARIAN.

Il lavoro svolto in questa tesi ha portato all'aggiunta di nuove funzionalità e alla risoluzione di vari problemi in SAHARIAN. Nella tesi si è effettuato un approfondito rework del componente "checkbox mixed", inizialmente completamente incliccabile, ora reso pienamente funzionante e interattivo. Per i collapsible, link-collapsible, tablist e treeview, è stato risolto un problema critico: precedentemente, al click, non venivano nascosti gli elementi non selezionati, rendendo tutti visibili indipendentemente dal focus. Questo fix ha richiesto una revisione dettagliata per garantire che solo l'elemento attivo fosse mostrato, migliorando così l'usabilità. Anche per i listbox, è stato aggiunto un layer di interattività che mancava in SAHARIAN, rendendoli ora cliccabili e più funzionali. Sono stati anche creati esempi per gli slider, sia orizzontali che verticali, e sviluppata una funzione specifica per la loro creazione in SAHARIAN. Un lavoro analogo è stato compiuto per le live region, con lo sviluppo di pagine d'esempio per ogni possibile configurazione e l'adeguamento di SAHARIAN per una corretta visualizzazione. Infine, sono stati rivisti gli esempi relativi agli attributi None, Presentation e Progressbar, rendendoli non visibili in SAHARIAN.

Questo strumento è stato progettato per fungere da guida essenziale per gli sviluppatori web, aiutandoli a identificare, comprendere e risolvere efficacemente i problemi di accessibilità presenti nelle loro pagine web. Attraverso una rappresentazione visuale dettagliata e intuitiva, SAHARIAN non solo evidenzia le aree problematiche, ma mette

anche in luce l'impatto che questi problemi possono avere sulle persone con disabilità. Per raggiungere questo scopo, SAHARIAN segue un metodo sistematico:

- **Rappresentazione visuale** L'estensione genera una rappresentazione visiva delle pagine web, che mira a evidenziare chiaramente i problemi di accessibilità. Questo aiuta gli sviluppatori a comprendere come una pagina web possa essere percepita da un utente con disabilità, ad esempio, da una persona non vedente che utilizza uno screen reader. Questa visualizzazione consente agli sviluppatori di vedere la loro pagina attraverso gli occhi di utenti con diverse necessità e di apportare modifiche consapevoli per migliorare l'accessibilità.

- **Rappresentazione delle operazioni** SAHARIAN illustra anche come una pagina sarebbe funzionale se operata esclusivamente tramite la tastiera. Questo aspetto è di fondamentale importanza, considerando che molti utenti non possono utilizzare un mouse. Assicurando che le pagine siano accessibili attraverso la navigazione da tastiera, SAHARIAN compie un passo importante verso la realizzazione di un web più inclusivo, dove l'accessibilità non è limitata solo alla visualizzazione, ma si estende anche all'interattività.

- **Mapping a concetti più familiari**Infine, SAHARIAN mappa i problemi di accessibilità a concetti ed esempi più familiari per gli sviluppatori. Questo processo di mappatura rende i problemi di accessibilità più comprensibili e consente agli sviluppatori di identificarli e risolverli più facilmente. Attraverso questo approccio, SAHARIAN non solo segnala le aree problematiche, ma fornisce anche soluzioni e strategie concrete, facendo leva sulla familiarità degli sviluppatori con certi concetti di sviluppo.

2.1.1 Installazione

Per installare Saharian basta entrare nel Google Chrome Store e importare un pacchetto esterno, selezionare la cartella contenente il progetto di Saharian e caricarla, poi si può trovare il plugin

2.1.2 Utilizzo

La schermata principale del plug-in

Una volta attivato il plugin SAHARIAN, gli utenti si troveranno di fronte a una schermata iniziale intuitiva e facilmente navigabile. Inizialmente, l'estensione sarà impostata su 'Off', indicando che non è ancora attiva. All'interazione dell'utente con il pulsante 'Min', SAHARIAN inizia un processo di caricamento che dura alcuni secondi. Durante questo intervallo, il programma esegue una serie di operazioni di rendering necessarie per modificare la pagina web. Queste operazioni includono l'adattamento e la modifica di vari elementi della pagina in modo da renderla più accessibile. Una volta completata la trasformazione, la pagina modificata viene presentata all'utente.

Il clic sul pulsante 'Max' innescerà un processo simile. Tuttavia, in questo caso, SAHARIAN applica una trasformazione più profonda della pagina, riducendo drasticamente l'uso del CSS e alterando maggiormente la presentazione visiva per enfatizzare gli

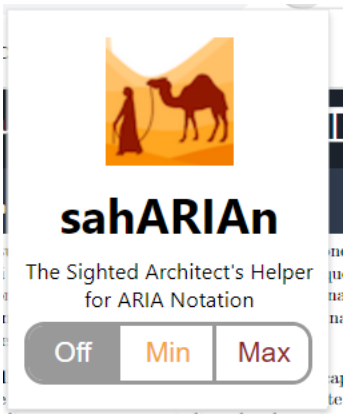

Figura 2.1: Pagina iniziale di SAHARIAN

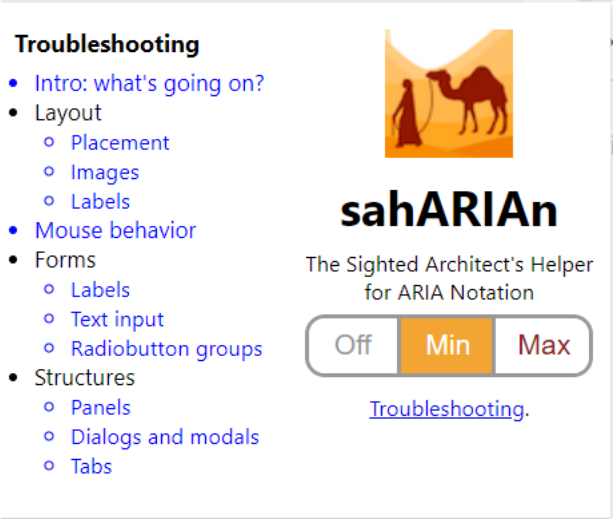

Figura 2.2: SAHARIAN dopo l'attivazione del min

aspetti legati all'accessibilità. Questa modalità 'Max' è particolarmente utile per identificare problemi di accessibilità che potrebbero non essere immediatamente evidenti nella modalità 'Min'.

In entrambe le modalità, 'Min' e 'Max', viene anche introdotto un pulsante di troubleshooting. Questo pulsante, una volta cliccato, rivela un elenco conciso ma informativo dei componenti principali di SAHARIAN. Ogni componente nell'elenco è accompagnato da un insieme di informazioni dettagliate che possono essere visualizzate con un ulteriore clic. Questa funzionalità di troubleshooting è progettata per fornire agli sviluppatori una comprensione più profonda dei vari elementi e aspetti di SAHARIAN, consentendo loro di acquisire una conoscenza più approfondita su come l'estensione modifica e migliora l'accessibilità delle loro pagine web.

2.1.3 Presentazione di esempi pratici, schermate e casi d'uso

Insieme allo sviluppo di SAHARIAN, è stata curata la creazione di 34 pagine di esempio, ciascuna progettata con uno scopo specifico: dimostrare l'efficacia di SAHARIAN nell'interazione con una vasta gamma di elementi HTML e attributi ARIA. Queste

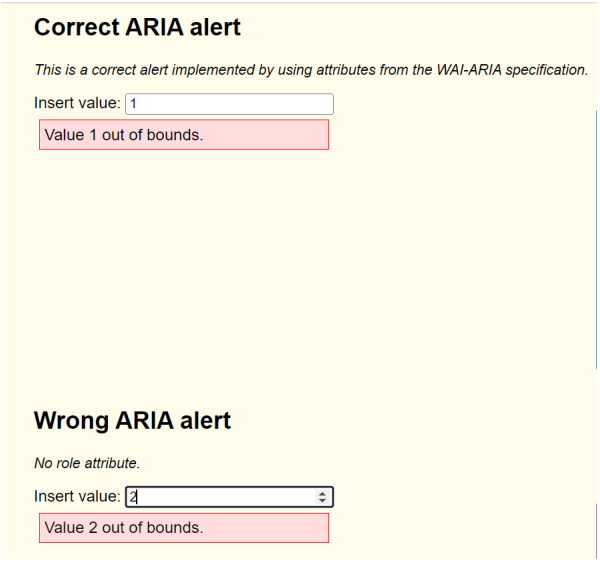

Figura 2.3: Pagina dell'Alert senza Saharian

pagine di esempio rappresentano una risorsa preziosa sia per gli sviluppatori sia per gli utenti interessati a comprendere in modo più concreto come SAHARIAN ottimizzi l'accessibilità delle pagine web.

Ogni pagina di esempio è stata attentamente progettata per mettere in luce specifiche funzionalità o sfide legate all'accessibilità. Questo approccio mirato permette agli sviluppatori di vedere direttamente in azione SAHARIAN, offrendo una comprensione pratica e approfondita delle sue capacità. Inoltre, attraverso questi esempi, gli sviluppatori possono apprendere come applicare tecniche simili nei loro propri progetti web, assicurando che le loro pagine siano accessibili a un pubblico più ampio, inclusi gli utenti con diverse esigenze e abilità.

La lista degli elementi HTML e degli attributi ARIA trattati nelle pagine di esempio include, ma non si limita a, elementi come button, link, form, input di testo, immagini, e navigazione, così come l'uso di attributi ARIA fondamentali come aria-label, aria-hidden, e role. Ogni esempio è stato creato per illustrare come SAHARIAN gestisce e migliora l'accessibilità di questi elementi, fornendo una dimostrazione visiva e funzionale dei cambiamenti apportati.

Di seguito, viene fornita una lista degli elementi e degli esempi pratici, che serve come guida per navigare tra le varie pagine di esempio. Questo elenco non solo funge da indice per le diverse funzionalità di SAHARIAN, ma serve anche come prontuario per gli sviluppatori che desiderano approfondire specifici aspetti dell'accessibilità web.

- **Alert** La pagina di esempio dedicata agli alert di SAHARIAN è un esempio dell'importanza di un'implementazione corretta degli standard di accessibilità. La pagina presenta due versioni di un alert: una etichettata come "corretta" e l'altra come "errata". Entrambe le versioni funzionano allo stesso modo in assenza di SAHARIAN includono un campo di testo per l'inserimento di un valore numerico e, se il numero inserito è fuori dai limiti accettabili, viene visualizzato un messaggio di avviso. Nella versione gestita da SAHARIAN, l'alert correttamente implementato mostra un trattamento visivo distintivo. Il testo dell'alert è evidenziato, segnalando chiaramente all'utente la presenza e l'importanza del-

Figura 2.4: Alert corretto con Saharian attivo

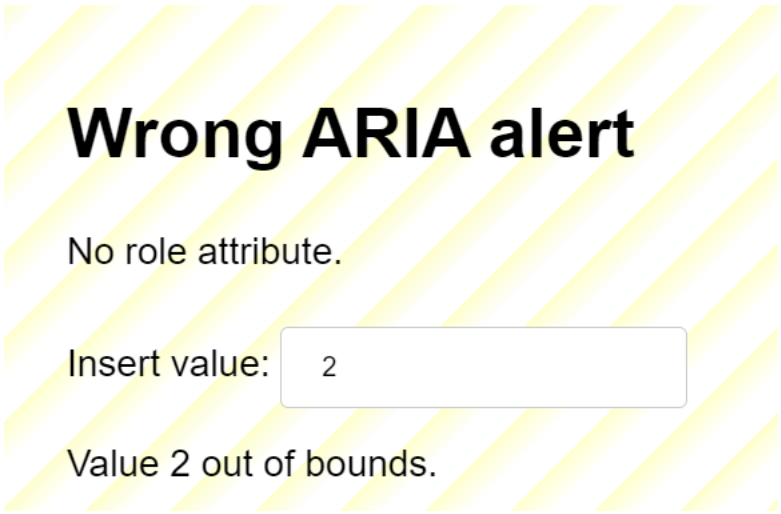

Figura 2.5: Alert innacessibile con SAHARIAN attivo

l'avviso. Questa evidenziazione è fondamentale per distinguere l'alert dal resto del contenuto della pagina e per assicurare che venga riconosciuto come tale dalle tecnologie assistive. Questa implementazione corretta rispecchia l'uso adeguato del ruolo 'alert' in ARIA, che è cruciale per garantire che l'alert sia percepito non solo visivamente, ma anche tramite lettori di schermo o altre tecnologie assistive.

Al contrario, nella versione "errata", dove il ruolo 'alert' non è stato implementato correttamente, il messaggio di avviso appare come semplice testo generico. Anche se questo testo rimane leggibile, non riesce a comunicare efficacemente il suo ruolo e la sua urgenza all'utente. In questa situazione, gli utenti che si affidano a tecnologie assistive potrebbero non riconoscere immediatamente il messaggio come un avviso critico, il che potrebbe portare a fraintendimenti o mancate risposte all'informazione importante.

- **Checkbox** L'esempio dei checkbox nella pagina di esempio di SAHARIAN offre una panoramica su come la corretta implementazione degli attributi ARIA e l'accessibilità da tastiera siano cruciali per la creazione di componenti web accessibili. La pagina presenta sei diversi checkbox, ognuno progettato per illustrare vari aspetti dell'accessibilità.

  I primi due checkbox sono esempi di accessibilità ben implementata. Il primo utilizza HTML di base per creare un checkbox funzionale e facilmente accessibile. Il secondo esempio, invece, si avvale di attributi ARIA corretti per migliorare

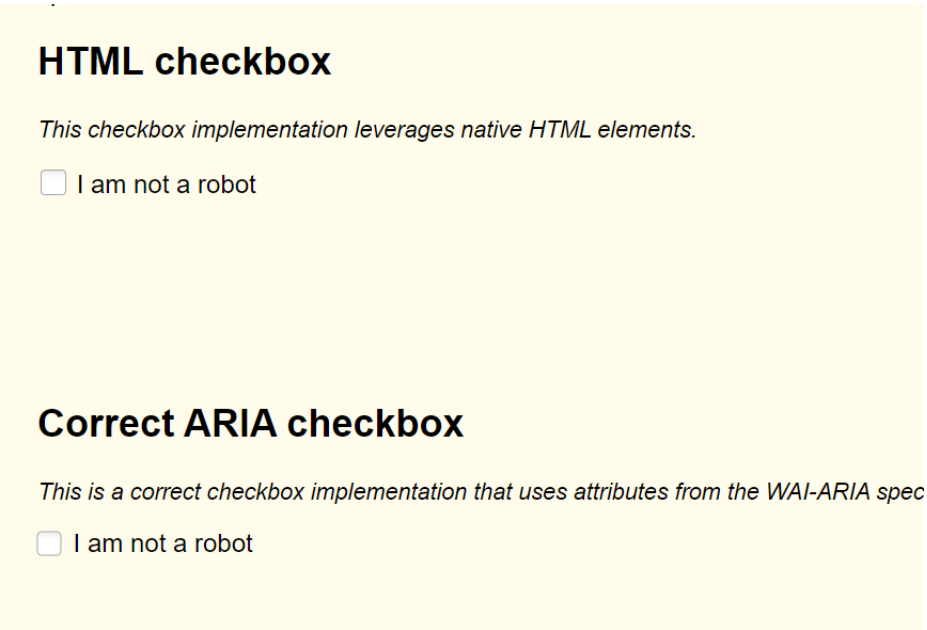

Figura 2.6: La pagina del checkbox senza SAHARIAN

l'accessibilità, dimostrando come un'attenta applicazione degli standard ARIA possa rendere i componenti web più inclusivi.

Gli altri quattro checkbox, tuttavia, rappresentano vari gradi di inaccessibilità:

- **Mancanza di Interazione da Tastiera** Uno dei checkbox errati non supporta l'interazione da tastiera, un aspetto fondamentale per gli utenti che non possono utilizzare un mouse
- **Campo ARIA Errato** In un altro esempio, il click sul checkbox modifica un attributo ARIA non appropriato, compromettendo l'accessibilità.
- **Nessun Aggiornamento dello Stato ARIA** Un altro checkbox errato non mostra alcun aggiornamento dello stato ARIA quando interagito, rendendo incerta la sua funzione.
- **Mancanza dell'Attributo ARIA** L'ultimo esempio manca completamente dell'attributo ARIA necessario per indicare che si tratta di un checkbox, risultando in un componente non riconoscibile e quindi non cliccabile.

Con SAHARIAN attivato, diventa evidente che solo i primi due checkbox sono pienamente funzionali e interattivi. I primi tre checkbox errati, sebbene cliccabili, rimangono inattivi, non riflettendo alcun cambiamento di stato. Questo dimostra l'importanza di un corretto uso degli attributi ARIA e dell'interazione da tastiera per assicurare che i componenti web siano utilizzabili da tutti gli utenti. L'ultimo checkbox, in mancanza del corretto attributo ARIA, non è cliccabile, evidenziando come SAHARIAN efficacemente non generi il componente in assenza di un'adeguata marcatura ARIA.

- **Immagini**

Figura 2.7: La parte inacessibile del checkbox, senza SAHARIAN

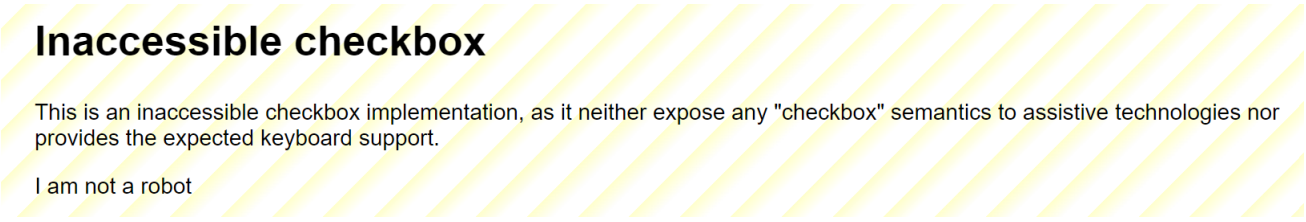

Figura 2.8: Il checkbox inacessibile con SAHARIAN attivo

Le immagini rappresentano un elemento fondamentale nell'accessibilità web, poiché possono fornire informazioni preziose agli utenti, anche se non visibili a tutti. SAHARIAN, attento a questa realtà, è stato opportunamente sviluppato per gestire in modo efficace le immagini, garantendo che siano accessibili anche a utenti con disabilità. La pagina di esempio dedicata alle immagini in SAHARIAN è articolata in due sezioni principali: una con immagini normali e l'altra con immagini interattive, che al click mostrano un alert.

La prima sezione si divide in cinque parti, di cui tre corrette e due errate. Nella prima corretta, è presente una semplice immagine accompagnata da un alt text descrittivo. La seconda sezione corretta è progettata per immagini decorative, che non necessitano di essere percepite dagli utenti con disabilità. Questo è realizzato inserendo un alt text vuoto oppure assegnando all'immagine il ruolo di "presentation" o "none". La terza immagine corretta è invece creata usando un div, al quale viene applicato l'attributo SAHARIAN "image" e un "aria-label", che funge da alt text equivalente per un elemento immagine HTML.

Le due sezioni errate presentano problematiche diverse: una contiene un'immagine HTML senza alt text, mentre l'altra è un'immagine realizzata mediante un div, ma priva di qualsiasi attributo ARIA.

Utilizzando SAHARIAN, è possibile osservare che nelle prime e terze sezioni corrette, le immagini non sono visualizzate; al loro posto, compare una piccola notifica contenente l'alt text e l'aria-label. Questo approccio assicura che le informazioni visive siano trasmesse anche agli utenti che non possono vedere le immagini. La seconda sezione, contenente immagini decorative, non mostra alcuna immagine, il che è coerente con l'intento di renderle non necessarie per gli utenti con disabilità.

Per quanto riguarda le sezioni errate, la prima mostra una notifica simile a quelle delle sezioni corrette, ma in colore rosso e con un messaggio di avviso sulla mancanza dell'alt text. Questo aiuta a sottolineare l'importanza di includere descrizioni testuali per le immagini. L'ultima immagine, quella sbagliata creata con un div, non è visualizzata affatto, evidenziando la mancanza di attributi ARIA appropriati.

La seconda sezione della pagina di esempio di SAHARIAN, dedicata alle immagini interattive, offre un interessante spaccato sull'importanza dell'uso corretto degli attributi ARIA e del tabindex per garantire l'interattività delle immagini. Questa sezione è divisa in due parti distinte: una implementata correttamente e l'altra errata, ma entrambe corrette per quanto riguarda la visualizzazione delle immagini.

Entrambe le parti utilizzano l'elemento HTML 'img' con un alt text appropriato. Ciò garantisce che, quando modificate con SAHARIAN, le immagini siano visualizzabili in entrambi i casi. Tuttavia, la differenza cruciale tra le due parti sta nell'implementazione dell'interattività.

Nella prima parte, correttamente implementata, oltre all'uso dell'elemento 'img', è stato aggiunto l'attributo ARIA "button" e il tabindex. Questa combinazione consente alle immagini di mantenere la loro funzionalità interattiva anche quando SAHARIAN è attivato. Di conseguenza, le immagini in questa sezione restano

cliccabili e, al click, mostrano il messaggio previsto, proprio come in un pulsante interattivo standard.

Al contrario, la seconda parte, pur essendo visualizzata correttamente, manca dell'attributo ARIA "button" e del tabindex. Questa omissione ha un impatto significativo sull'interattività dell'immagine: quando SAHARIAN viene attivato, le immagini in questa sezione non sono cliccabili. Questo esempio dimostra chiaramente come l'assenza degli attributi ARIA e del tabindex appropriati possa influenzare negativamente l'accessibilità e l'usabilità di elementi interattivi in una pagina web.

- **Live region**

  Le live region rappresentano un aspetto sofisticato e cruciale dell'accessibilità web, reso possibile grazie agli attributi ARIA. Anche se possono essere difficili da implementare in una normale pagina HTML, il loro ruolo nell'accessibilità è di vitale importanza. La pagina di esempio creata per illustrare il funzionamento delle live region in SAHARIAN comprende quattro sezioni distinte, ognuna configurata con diversi settaggi di ARIA live e ARIA atomic.

  Nella versione HTML standard, tutti e quattro gli esempi hanno un comportamento identico: includono un pulsante e un campo di testo contenente una lunga versione del "lorem ipsum". Al click del pulsante, un carattere casuale nel testo viene modificato. Tuttavia, una volta attivato SAHARIAN, l'interazione con queste sezioni diventa molto più dinamica e differenziata.

  Le prime due sezioni sono impostate con la live region settata su "assertive". La prima di queste ha l'attributo aria-atomic impostato su true, mentre la seconda lo ha impostato su false. Questa configurazione determina come SAHARIAN gestisce e notifica i cambiamenti nel contenuto. Quando avviene una modifica in una delle live region impostate su "assertive", SAHARIAN reagisce immediatamente, guidando l'utente verso una notifica che segnala il cambiamento. Se aria-atomic è impostato su true, la notifica presenterà l'intero contenuto della sezione modificata. Al contrario, se aria-atomic è false, la notifica mostrerà solo la parte del contenuto che è stata effettivamente modificata.

  Le restanti due sezioni hanno la live region impostata su "polite", con aria-atomic nuovamente settato su true e false rispettivamente. In questi casi, la notifica dei cambiamenti avviene in modo più discreto: SAHARIAN mostra le notifiche di modifica solo quando l'utente non sta compiendo alcuna azione attiva. Questo significa che le notifiche vengono mostrate solo quando l'utente è inattivo, assicurando che le informazioni importanti non interrompano o distraggano l'utente durante altre attività.

- **Slider**

  Per la valutazione dell'accessibilità degli slider, è stata ideata una pagina test strutturata in tre sezioni distinte per esaminare e confrontare differenti approcci nella realizzazione di slider interattivi. La prima sezione ospita uno slider implementato tramite codice HTML standard, fungendo da benchmark per le funzionalità di base attese da un componente slider.

Le seconde e terze sezioni, invece, presentano due slider costruiti utilizzando elementi div e tecniche di styling CSS per emulare il comportamento di uno slider tradizionale. La principale differenza tra questi due slider risiede nell'uso degli attributi ARIA: il secondo slider è arricchito con attributi ARIA pertinenti, come role="slider", oltre a specificare i valori massimo e minimo tramite aria-valuemax e aria-valuemin, mentre il terzo slider è privo di qualsiasi attributo ARIA, affidandosi esclusivamente al CSS per la sua funzionalità e presentazione visiva.

In tutte e tre le sezioni, gli slider sono progettati per modificare il loro valore in risposta all'interazione con il mouse, permettendo agli utenti di trascinare il cursore verso il valore massimo o minimo. Tuttavia, con l'attivazione di SAHARIAN, si osserva un comportamento differenziato: mentre i primi due slider continuano a funzionare come previsto, il secondo slider, grazie all'intervento di SAHARIAN, viene trasformato in uno slider HTML standard. Durante questa trasformazione, SAHARIAN preserva e applica gli attributi ARIA originali, come aria-valuemax e aria-valuemin, assicurando che lo slider mantenga la sua funzionalità accessibile.

Il terzo slider, privo di attributi ARIA, risulta invece non visibile all'attivazione di SAHARIAN, poiché lo strumento elimina tutto il CSS che contribuiva alla sua rappresentazione visiva. Questo evidenzia l'importanza degli attributi ARIA per garantire che componenti interattivi come gli slider siano non solo visibili ma anche funzionali in un contesto privo di stili CSS, in linea con le pratiche di accessibilità web.

Un'esplorazione aggiuntiva riguarda lo slider verticale, per il quale è stata predisposta una pagina dedicata. Il comportamento di questo slider verticale è analogo a quello degli slider orizzontali, ma con l'orientamento modificato per dimostrare come SAHARIAN possa adeguatamente gestire e presentare slider in configurazioni verticali, a condizione che siano implementati correttamente con gli attributi ARIA appropriati. Questa dimostrazione sottolinea ulteriormente la flessibilità di SAHARIAN nell'adattarsi a vari schemi di interazione, promuovendo la creazione di soluzioni accessibili che tengano conto delle diverse modalità di navigazione e interazione da parte degli utenti.

2.2 Esempi su siti esistenti

- **Amazon**

  Nell'esaminare l'interfaccia utente di siti web complessi e densamente popolati di contenuti, come si evince dalla home page di Amazon e dalla pagina del carrello con un articolo inserito, emerge un quadro promettente per quanto riguarda l'accessibilità. Gli elementi interattivi quali pulsanti e link rispondono efficacemente al click, offrendo una rappresentazione visiva adeguata che include anche la navigazione per categorie. Questo dimostra un livello di accessibilità notevole, ulteriormente sottolineato dalla presenza di etichette descrittive accanto ai pulsanti che facilitano l'identificazione delle funzioni per gli utenti che si affidano a tecnologie assistive. Tuttavia, l'analisi mette in luce una sfida significativa nell'utilizzo di SAHARIAN per la valutazione di pagine web di grande dimensione. La

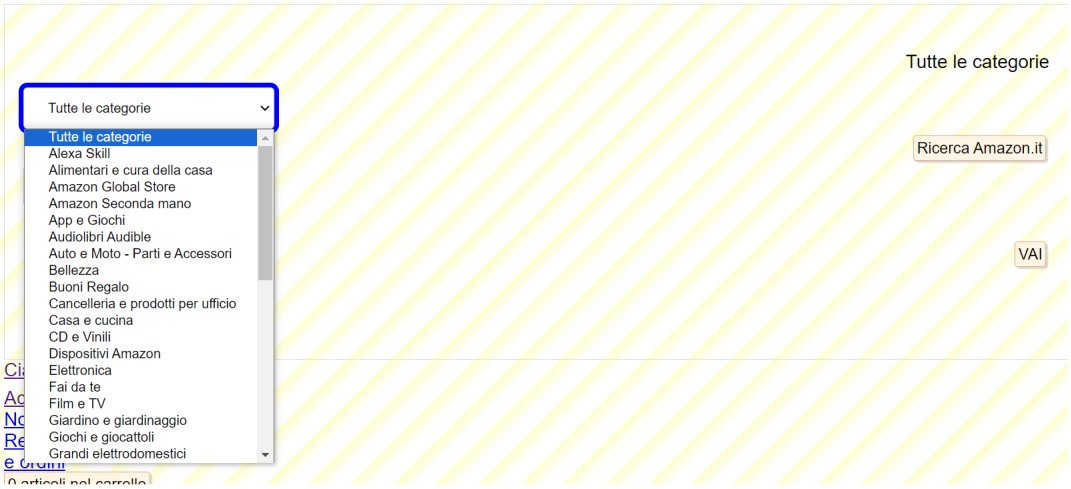

Figura 2.9: Selezione categorie di Amazon con Saharian attivo

Figura 2.10: Un prodotto di Amazon mostrato con Saharian attivo

Figura 2.11: Homepage di MDN

generazione della pagina accessibile, incontra ostacoli notevoli in termini di tempo di elaborazione. La complessità e la ricchezza di contenuti di alcune pagine, come quelle dedicate ai prodotti su Amazon, che includono una moltitudine di link, immagini, descrizioni, prodotti correlati e recensioni, possono sovraccaricare lo strumento. In questi casi, il tempo necessario per elaborare e presentare una versione della pagina ottimizzata per l'accessibilità si estende significativamente, a volte impedendo del tutto la generazione di una pagina fruibile. Questo limite rappresenta un punto critico per SAHARIAN, suggerendo la necessità di ottimizzazioni mirate per gestire pagine web di ampia portata. Un approccio potrebbe includere il miglioramento dell'algoritmo di elaborazione di SAHARIAN per incrementare l'efficienza nella gestione degli elementi della pagina. Tecniche quali l'elaborazione incrementale, dove la pagina viene processata e resa accessibile in segmenti o componenti, potrebbero ridurre il tempo di attesa e migliorare l'esperienza di utilizzo dello strumento. Inoltre, l'introduzione di una funzionalità che permetta agli sviluppatori di selezionare specifiche sezioni della pagina per l'analisi potrebbe offrire un metodo più diretto e veloce per valutare l'accessibilità senza la necessità di processare l'intera pagina in un unico passaggio.

- **MDN** Nell'analisi di SAHARIAN applicato al sito del Mozilla Developer Network, emergono ulteriori conferme dell'efficacia dello strumento nel valutare l'accessibilità web. Attraverso l'esame del sito, si osserva che anche questa piattaforma rispetta elevati standard di accessibilità, caratteristica evidenziata dalla presenza di link dedicati che consentono agli utenti di saltare direttamente alla barra di ricerca o ai contenuti principali. Questa funzionalità di "skip link" è fondamentale per utenti che si avvalgono di tecnologie assistive, poiché permette una navigazione più rapida ed efficiente, bypassando elementi del menu o del layout che altrimenti richiederebbero una navigazione sequenziale prolungata.

  In aggiunta, la barra di ricerca del sito Mozilla Developer si dimostra pienamente accessibile, integrando funzionalità di autocomplete che sono resi visibili

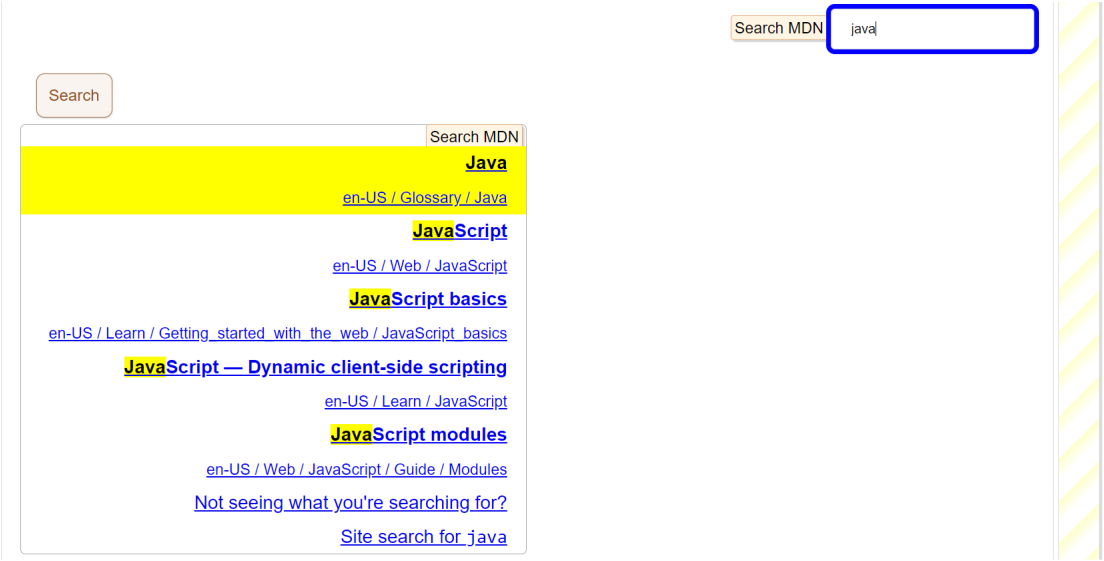

Figura 2.12: Barra di ricerca su MDN, con l'utilizzo del autocomplete

e navigabili anche tramite tecnologie assistive. Questo aspetto è particolarmente rilevante perché garantisce che gli utenti possano beneficiare di suggerimenti di ricerca in tempo reale, migliorando l'usabilità generale del sito per tutti gli utenti, indipendentemente dalle loro modalità di interazione.

L'efficacia con cui SAHARIAN riesce a evidenziare questi aspetti positivi dell'accessibilità su piattaforme ben progettate sottolinea il suo valore come strumento di verifica e sensibilizzazione. Questi risultati non solo confermano l'utilità di SAHARIAN nel riconoscere le pratiche di accessibilità implementate correttamente, ma incoraggiano anche gli sviluppatori a seguire queste best practices nel design e nello sviluppo web. L'esperienza utente migliorata su siti come quello del Mozilla Developer Network evidenzia come l'attenzione all'accessibilità contribuisca significativamente alla creazione di un web più inclusivo e navigabile per tutti.

Il caso di Mozilla Developer, analizzato tramite SAHARIAN, rafforza ulteriormente l'argomento che una progettazione web inclusiva e accessibile non solo è possibile, ma è anche alla portata degli sviluppatori che dispongono degli strumenti giusti per valutare e migliorare le loro creazioni.

Capitolo 3

Architettura di SAHARIAN

3.1 Descrizione di come funziona un browser extension

Le estensioni dei browser rappresentano moduli software che ampliano le funzionalità dei browser web, offrendo agli utenti strumenti aggiuntivi, miglioramenti dell'interfaccia utente, e personalizzazioni delle pagine web. Sviluppate utilizzando tecnologie web familiari come HTML, CSS e JavaScript, le estensioni possono interagire sia con le API web disponibili nelle pagine web che con un set specifico di API JavaScript fornite dalle piattaforme dei browser. Questo consente alle estensioni di eseguire una gamma più ampia di azioni rispetto al codice JavaScript standard eseguito all'interno delle pagine web.

Le componenti principali di un'estensione includono:

- **Manifest file** Un file JSON che definisce metadati basilari, permessi, e punti di ingresso dell'estensione.
- **Background scripts** Script in background che gestiscono eventi e coordinano la logica dell'estensione.
- **Content scripts** Script iniettati nelle pagine web per interagire con il contenuto della pagina e modificare il DOM secondo le necessità.
- **Popup UI** Interfacce utente create con HTML e CSS che forniscono un'interazione diretta con l'utente, solitamente accessibili tramite icone nella barra degli strumenti del browser.
- **Options page** Una pagina di configurazione che permette agli utenti di personalizzare le impostazioni dell'estensione.

Le estensioni possono eseguire una varietà di funzioni, come la modifica del contenuto delle pagine web, il blocco di annunci pubblicitari, la gestione delle password, e molto altro. Tuttavia, operano entro limiti imposti per motivi di sicurezza e privacy, che impediscono loro di accedere a determinate funzionalità e dati sensibili senza espliciti permessi dell'utente.

L'interazione con le API specifiche del browser permette alle estensioni di offrire funzionalità innovative, migliorando significativamente l'esperienza di navigazione degli

utenti. Grazie alla loro versatilità e alla capacità di personalizzare l'esperienza web, le estensioni sono diventate strumenti indispensabili per molti utenti internet.[Moz]

3.2 Struttura concettuale e tecnica di SAHARIAN

La struttura concettuale e tecnica di SAHARIAN emerge da un'innovativa metodologia di test e valutazione dell'accessibilità, introdotta in questo progetto di ricerca per consentire agli sviluppatori di percepire direttamente le problematiche di accessibilità e il loro impatto sugli utenti con disabilità. Questa metodologia si basa su due pilastri fondamentali: il rendering visuale e l'operabilità del mouse, integrati in un'estensione per il browser Google Chrome che trasforma le pagine web in versioni che evidenziano le aree di miglioramento in termini di accessibilità.

- **Architettura e Design Concettuale** SAHARIAN è progettato per agire come una lente attraverso cui gli sviluppatori possono vedere gli ostacoli all'accessibilità, trasformando visivamente le pagine web standard in modo che riflettano direttamente la loro accessibilità. Questo si traduce in una piattaforma che non solo enfatizza la visualizzazione intuitiva dei problemi di accessibilità ma fornisce anche feedback interattivi e immediati, facilitando la comprensione e la risoluzione di tali problemi.

- **Tecnologie e Implementazione** Tecnicamente, SAHARIAN sfrutta le potenzialità di Google Chrome per analizzare e modificare le pagine web. Utilizzando JavaScript e le sue librerie, manipola il DOM delle pagine, identificando e adattando elementi secondo gli standard ARIA e altre pratiche di accessibilità web. Ciò comprende l'adattamento di elementi visivi e la gestione dell'interattività, come la navigazione da tastiera e le live region, assicurando che l'interfaccia utente sia completamente accessibile.

- **Interazione con gli Standard ARIA** Un aspetto chiave di SAHARIAN è l'interpretazione e l'applicazione accurata degli standard ARIA, che consente di garantire che le pagine web non solo rispettino i criteri tecnici di accessibilità ma siano anche funzionalmente accessibili. Analizzando e correggendo gli attributi ARIA, SAHARIAN assicura un'esperienza utente inclusiva.

- **Feedback e Troubleshooting** L'estensione fornisce un feedback dettagliato sugli aspetti di accessibilità, evidenziando problemi e offrendo soluzioni pratiche. Inoltre, grazie a una funzionalità di troubleshooting, facilita una comprensione approfondita di come i vari elementi di accessibilità siano implementati, promuovendo miglioramenti concreti.

- **Privacy e Sicurezza** Operando interamente lato client, SAHARIAN mantiene tutte le operazioni localmente nel browser, evitando trasferimenti di dati sensibili e garantendo la privacy e la sicurezza delle informazioni. Questo approccio consente l'uso sicuro dell'estensione anche su pagine web confidenziali, rispettando le normative sulla protezione dei dati come il GDPR.

In aggiunta, SAHARIAN si adatta alle dinamiche delle pagine web moderne, riflettendo le informazioni veicolate al DOM e monitorando le sue mutazioni per aggiornare

la rappresentazione visiva in risposta ai cambiamenti, utilizzando tecnologie come l'API Mutation Observer.

3.3 Descrizione degli algoritmi rilevanti

All'avvio dell'estensione SAHARIAN, in primo luogo, SAHARIAN effettua un controllo del livello di accessibilità impostato dall'utente. Se il livello è impostato su "Off", SAHARIAN inizia immediatamente una procedura di deattivazione. Questa fase è cruciale, poiché comporta la rimozione di tutte le modifiche precedentemente applicate al CSS e al codice JavaScript durante la sessione iniziale di renderizzazione. Tale processo è essenziale per garantire che lo sviluppatore possa ritornare all'uso convenzionale della pagina web una volta terminato l'impiego di SAHARIAN, preservando l'integrità dell'originale design e funzionalità.

Nel caso in cui il livello di accessibilità sia diverso da "Off", l'estensione inizia a implementare le sue funzionalità avanzate con un meccanismo unico: ogni funzione viene ripetuta a intervalli regolari di 500 millisecondi. Questo ritmo costante di ripetizione assicura che ogni modifica apportata al Document Object Model (DOM) sia accuratamente rilevata e gestita da SAHARIAN, permettendo così una completa sincronizzazione tra le dinamiche di modifica della pagina e le funzionalità di accessibilità introdotte, questa strategia permette a SAHARIAN di adattarsi dinamicamente alle evoluzioni della pagina web, assicurando che le modifiche al DOM siano immediatamente identificate e che le risposte correttive siano applicate senza ritardi. Grazie a questo ciclo di verifica e aggiornamento costante, SAHARIAN riesce a mantenere l'accessibilità della pagina al passo con ogni nuova aggiunta o modifica.

Successivamente, SAHARIAN intraprende la modifica del CSS, un passaggio fondamentale che varia in funzione del livello di accessibilità selezionato, che può essere minimo o massimo. La personalizzazione del CSS avviene attraverso l'implementazione di un insieme di regole CSS specifiche, progettate per agire esclusivamente su elementi HTML basilari o su quegli elementi che includono un attributo ARIA. Ciò significa che se un elemento non possiede un attributo ARIA, la sua visualizzazione sarà influenzata negativamente, risultando in una presentazione non corretta o nella sua completa omissione. Questo approccio selettivo enfatizza l'importanza di utilizzare correttamente gli attributi ARIA per migliorare l'accessibilità del contenuto web.

3.3.1 Funzioni principali

La generazione di SAHARIAN sfrutta cinque funzioni principali, generateAccessibleNames, deactivateImages, deactivateStyles, deactivateStylesheets e generateAriaLiveControls, di seguito una descrizione di tutte

- **generateAccessibleNames** La funzione generateAccessibleNames() si articola in diverse fasi operative, inizialmente, la funzione impiega un metodo di selezione basato sull'utilizzo di querySelectorAll per identificare gli elementi che fungono da slider, specificamente quelli che presentano un attributo ARIA definito come role="slider".

  Nel caso in cui vengano rilevati uno o più slider, viene invocata la funzione replaceCustomSliderWithHTMLSlider(), la quale, come esplicitato dal suo no-

me, sostituisce gli slider rilevati con equivalenti HTML nativi. Questo processo di sostituzione assicura che gli slider siano completamente accessibili, replicando gli attributi essenziali dello slider originale per preservarne la funzionalità e l'integrazione nell'interfaccia utente.

Successivamente, la funzione procede nell'analisi degli elementi che includono attributi ARIA riferiti a labeledby e describedby, intraprendendo un'iterazione accurata su ciascuno di essi. Durante questa iterazione, viene applicata la funzione generateTextAlternative a ogni elemento, rappresentando così l'elemento centrale dell'algoritmo di generazione del testo alternativo. Questo aspetto della funzione è fondamentale per controllare che ogni elemento non testuale della pagina web sia accompagnato da una rappresentazione testuale chiara e comprensibile.

La funzione generateAccessibleNames() si articola in diverse fasi operative, ognuna delle quali è mirata a migliorare la comprensibilità e l'accessibilità di elementi specifici all'interno di una pagina web. Inizialmente, la funzione impiega un metodo di selezione basato sull'utilizzo di querySelectorAll per identificare gli elementi che fungono da slider, specificamente quelli che presentano un attributo ARIA definito come role="slider". Questa fase preliminare è cruciale per l'individuazione di componenti interattivi che potrebbero non essere pienamente accessibili secondo gli standard web moderni.

Nel caso in cui vengano rilevati uno o più slider, viene invocata la funzione replaceCustomSliderWithHTMLSlider(), la quale, come esplicitato dal suo nome, sostituisce gli slider rilevati con equivalenti HTML nativi. Questo processo di sostituzione assicura che gli slider diventino completamente accessibili, replicando gli attributi essenziali dello slider originale per preservarne la funzionalità e l'integrazione nell'interfaccia utente.

Successivamente, la funzione procede nell'analisi degli elementi che includono attributi ARIA riferiti a labeledby e describedby, intraprendendo un'iterazione accurata su ciascuno di essi. Durante questa iterazione, viene applicata la funzione generateTextAlternative a ogni elemento, rappresentando così l'elemento centrale dell'algoritmo di generazione del testo alternativo. Questo aspetto della funzione è fondamentale per garantire che ogni elemento non testuale della pagina web sia accompagnato da una rappresentazione testuale chiara e comprensibile, migliorando significativamente l'accessibilità per gli utenti che si affidano a tecnologie assistive.

La metodologia adottata per determinare il testo alternativo appropriato si basa sull'analisi di una serie di attributi, quali aria-labelledby, aria-describedby, aria-label, e sulle proprietà HTML standard come title e alt. Un'attenzione particolare è rivolta alla visibilità degli elementi, utilizzando la funzione isHidden per assicurare che il testo alternativo venga generato esclusivamente per elementi accessibili. Inoltre, l'implementazione di una variabile di controllo, currentTraversal, evita la possibilità di ricorsione infinita, garantendo che ogni elemento venga valutato una singola volta all'interno del processo di generazione del testo alternativo.

Questa funzione si estende anche alla gestione di elementi specifici, quali campi di input e tabelle, differenziando tra quelli che possono ospitare direttamente il

testo alternativo generato e quelli che, a causa delle loro caratteristiche intrinseche, richiedono un approccio differenziato per l'inserimento del testo alternativo.

- **deactivateimages** La funzione deactivateImages() e il metodo di supporto sideline() svolgono un ruolo significativo nel contesto di SAHARIAN, il cui obiettivo è gestire la renderizzazione delle immagini. La funzione asincrona deactivateImages() è progettata per modificare il modo in cui le immagini vengono presentate nella pagina web, sostituendo visivamente le immagini con testi alternativi laddove appropriato. Inizialmente, la funzione individua tutti gli elementi di immagine, inclusi ¡img¿, ¡source¿, e quelli con un attributo role="img", creando un array di tali elementi. Per ogni immagine identificata, la funzione verifica la presenza di testo alternativo (alt text) o etichette ARIA (ariaLabel) e, in assenza di tali, assegna un testo predefinito che indica la mancanza di testo alternativo o il carattere decorativo dell'immagine. Successivamente, per ciascuna immagine, viene creato dinamicamente un elemento ¡div¿ che contiene il testo alternativo e viene posizionato visivamente dove prima era presente l'immagine. La funzione sidelined() gioca un ruolo critico nel processo di verifica dello stato degli attributi degli elementi immagine gestiti da SAHARIAN. Questa funzione controlla se un elemento ha subito modifiche temporanee attraverso la funzione sideline(), permettendo di identificare se le modifiche apportate sono ancora in atto o se sono state definite come permanenti (marcate con "done"). La funzione opera mediante la verifica della presenza di un attributo hash personalizzato (data-saharian-sideline-[attr]-hash) e confrontando il suo valore con quello previsto, determinando così se l'elemento è stato modificato e se tale modifica è stata completata o meno.

  Questa metodologia consente a SAHARIAN di mantenere un controllo dettagliato sullo stato degli attributi degli elementi della pagina web

  Il metodo sideline() gioca un ruolo complementare nella funzione deactivateImages(), gestendo gli attributi degli elementi di immagine in modo da preservare le informazioni originali mentre si applicano modifiche temporanee. Questo metodo salva gli attributi esistenti dell'elemento (come src, srcset, e role) in attributi personalizzati data-saharian-sideline-*, consentendo la reversibilità delle modifiche e assicurando che le informazioni originali non vengano perse.

- **deactivateStyles** deactivateStyles() esegue una scansione del Document Object Model (DOM) per identificare tutti gli elementi che hanno attributi di stile inline. Questa operazione prepara il terreno per un'analisi approfondita delle pratiche di styling dirette che possono influenzare la presentazione del contenuto. Una volta individuati questi elementi, la funzione procede con la modifica degli attributi di stile, applicando una serie di criteri predeterminati per adeguare o rimuovere specifiche proprietà CSS. Questo processo è volto a modificare dinamicamente la pagina, facilitando una valutazione dell'impatto che specifici stili hanno sulla fruibilità e sull'accessibilità del sito. La generateReplacementForRule() dettaglia il meccanismo attraverso il quale le regole CSS degli elementi selezionati vengono analizzate e potenzialmente riscritte. Questa funzione prende in considerazione proprietà CSS chiave come position, display, e visibility, e le adatta secondo logiche che mirano a ottimizzare la presentazione degli elementi. Attraverso l'elaborazione di nuove regole CSS, generateReplacementForRule() interviene sulle

proprietà inline per rendere gli elementi più coerenti con criteri di accessibilità ottimizzati, senza alterare il contenuto o la struttura semantica della pagina. Anche qui le funzioni sideline() e sidelined() giocano un ruolo cruciale nella gestione e nell'analisi delle modifiche agli attributi degli elementi. sideline() è responsabile dell'archiviazione temporanea degli attributi originali degli elementi prima che vengano modificati, consentendo una facile restaurazione dello stato precedente se necessario. Ciò garantisce che le modifiche apportate per testare l'accessibilità possano essere invertite, preservando l'integrità originale del sito web durante il processo di sviluppo e valutazione.

D'altra parte, sidelined verifica se un elemento ha già subito modifiche attraverso sideline(), controllando l'esistenza di attributi personalizzati che indicano modifiche precedenti. Questo meccanismo di controllo permette di evitare ridondanze nelle modifiche e assicura che ogni elemento venga trattato in modo appropriato nel contesto delle operazioni di miglioramento dell'accessibilità.

- **deactivateStylesheets** La funzione deactivateStylesheets() insieme alla funzione di supporto replacedRules() e alla già discussa generateReplacementForRule(), svolgono un ruolo chiave in SAHARIAN. deactivateStylesheets() esegue un'operazione critica: disabilita i fogli di stile esterni e incorporati non essenziali per l'accessibilità, scansionando l'array dei fogli di stile del documento, la funzione identifica e processa ciascun foglio di stile, controllando la presenza di attributi o percorsi che non sono marcati come parte di SAHARIAN. Per i fogli di stile esterni, deactivateStylesheets() recupera il CSS attraverso richieste asincrone, mentre per quelli interni utilizza il CSS già disponibile nel DOM. Il risultato è la generazione di nuove regole CSS, derivanti dalla trasformazione delle regole esistenti, che vengono poi applicate al documento per sostituire i fogli di stile originali disabilitati. replacedRules() elabora il CSS recuperato da deactivateStylesheets(), analizzando ogni regola CSS per determinare se necessita di modifica in base ai criteri di accessibilità definiti. Supporta la gestione di regole importate, regole dei media, e regole di stile, applicando la logica di trasformazione specifica per ogni tipo di regola. Per esempio, le regole dei media vengono preservate ma i loro contenuti (le regole CSS interne) vengono trasformati per riflettere le modifiche desiderate. Questo processo assicura che le modifiche al CSS non solo mantengano la struttura e la logica originale del foglio di stile, ma introducano anche miglioramenti mirati al comprendere meglio l'accessibilità. generateReplacementForRule agisce su singole regole CSS, valutando ciascuna proprietà CSS presente e determinando se e come dovrebbe essere modificata. La funzione prende in considerazione vari aspetti del CSS, come la posizione (position), la visualizzazione (display), e la visibilità (visibility), applicando criteri specifici per ciascuna proprietà

- **generateAriaLiveControls** La funzione generateAriaLiveControls è un meccanismo sofisticato di SAHARIAN, mirato a migliorare la comprensione degli sviluppatori su come le modifiche dinamiche al contenuto delle pagine web sono annunciate ai lettori di schermo, attraverso l'uso degli attributi ARIA aria-live e aria-atomic. generateAriaLiveControls inizia definendo e iniettando stili CSS nel documento per evidenziare le modifiche ai contenuti nei div aria-live. Questi stili sono progettati per catturare l'attenzione dello sviluppatore sulle modifiche

in tempo reale, differenziando tra notifiche assertive e polite mediante colori e bordi distinti.

La funzione procede identificando tutti i div con l'attributo aria-live, applicando loro uno stile che garantisce la loro scorribilità e imposta un'altezza massima, facilitando la visualizzazione delle modifiche all'interno di un contesto limitato.

Per ogni div identificato, generateAriaLiveControls utilizza un MutationObserver per monitorare le modifiche al contenuto. Quando una modifica viene rilevata, la funzione determina se il contenuto modificato dovrebbe essere evidenziato (basandosi sull'attributo aria-atomic e il tipo di aria-live). Le modifiche significative vengono quindi evidenziate visivamente all'interno del div, con comportamenti distinti basati sul valore di aria-live:

- **Assertive**Per le modifiche ritenute urgenti (assertive), oltre all'evidenziazione, il carattere o il contenuto modificato può essere reso il focus della vista, garantendo che tali modifiche catturino l'attenzione immediata del utente.
- **Polite** Le modifiche meno urgenti (polite) vengono segnalate con un'evidenziazione che non induce uno scroll o un focus automatico, ma distingue visivamente la modifica applicando un bordo colorato intorno al div modificato.

Nell'ambito dell'accessibilità web, l'attributo aria-atomic gioca un ruolo fondamentale nella definizione di come le modifiche al contenuto di una regione live ARIA (aria-live) debbano essere presentate agli utenti di tecnologie assistive, come i lettori di schermo. Questo attributo può avere due valori, true o false, che influenzano il comportamento della tecnologia assistiva quando il contenuto della regione live cambia.

- **aria-atomic="true"** Quando aria-atomic è impostato su true (il valore predefinito se l'attributo non è specificato), indica che, in caso di una modifica al contenuto di una regione live, l'intero contenuto della regione deve essere annunciato dalla tecnologia assistiva, non solo la parte modificata. Questo approccio assicura che l'utente riceva il contesto completo, rendendo le modifiche comprensibili all'interno dell'intera porzione di contenuto. È particolarmente utile quando una modifica parziale potrebbe non avere senso senza considerare l'intero contesto del contenuto circostante.
- **aria-atomic="false"** Al contrario, aria-atomic="false" indica che solo le modifiche al contenuto dovrebbero essere annunciate, senza necessariamente riferire l'intero contenuto della regione live. Questo comportamento è utile per gli aggiornamenti frequenti o incrementali dove l'annuncio dell'intero contenuto potrebbe risultare ridondante o eccessivo, permettendo agli utenti di concentrarsi solo sulle nuove informazioni fornite.

La funzione generateAriaLiveControls implementa questa distinzione comportamentale attraverso l'osservazione delle modifiche nei div con attributo aria-live. Quando rileva una modifica:

  - Per i div con aria-atomic="false", la funzione cerca la prima differenza tra il contenuto precedente e quello attuale e la evidenzia, fornendo una visualizzazione focalizzata sulla modifica specifica. Questo approccio mira a simulare come le tecnologie assistive gestirebbero l'annuncio di tali modifiche, facendo attenzione a non sovraccaricare l'utente con informazioni non necessarie.
  - Quando aria-atomic non è specificato o è impostato su true, e aria-live="assertive", la funzione può scegliere di presentare un'interazione più invasiva, come lo scrolling automatico verso il contenuto modificato e il suo focus, riflettendo la necessità di catturare immediatamente l'attenzione dell'utente su queste modifiche.

- **activateDynamics** activateDynamics è una funzione di inizializzazione che attiva specifici listener di eventi su elementi del DOM che presentano caratteristiche dinamiche, quali quelle designate dall'attributo aria-controls. Questa funzione esegue due azioni principali:

  - **Gestione del Focus** Installa un listener di eventi globali per il focus, assicurando che qualsiasi elemento che riceve il focus sia evidenziato in modo distintivo attraverso animazioni CSS. Questo aiuta a simulare visivamente come le tecnologie assistive potrebbero segnalare agli utenti la presenza e l'attivazione del focus su un elemento.
  - **Interattività aria-controls** Aggiunge listener di eventi 'click' agli elementi che utilizzano l'attributo aria-controls, facilitando la visualizzazione di come le azioni sugli elementi controller possano influenzare altri elementi specificati (destinatari del controllo).

  La funzione handleFocus reagisce agli eventi di focus applicando un'animazione CSS agli elementi target. Questa animazione crea un effetto visivo temporaneo di evidenziazione (un glow blu) attorno all'elemento, simulando un forte indicatore visivo del focus che può essere particolarmente utile per gli sviluppatori durante il testing dell'accessibilità. Questa visualizzazione aiuta a comprendere come gli utenti di lettori di schermo o altre tecnologie assistive possano percepire il focus all'interno della pagina. handleAriaControls risponde ai click sugli elementi con l'attributo aria-controls, mostrando l'effetto dell'azione di controllo sull'elemento destinatario. Quando un elemento controller viene attivato, la funzione cerca l'elemento destinatario specificato e ne facilita la visibilità attraverso lo scrolling fluido e un'animazione di evidenziazione (un glow verde), simboleggiando l'attenzione o la modifica dello stato. Questo meccanismo illustra visivamente l'impatto dell'interazione dell'utente con componenti web dinamici, enfatizzando l'importanza di gestire correttamente gli attributi ARIA per l'accessibilità.

- **injectEventBypass** La injectEventBypass è incaricata di impostare un listener globale su tutto il corpo del documento, catturando gli eventi di click e potenzialmente altri eventi del mouse. Questo approccio consente di intercettare e modificare il comportamento degli eventi prima che raggiungano gli elementi target, facilitando la simulazione di come le interazioni basate sul mouse possano essere tradotte in azioni equivalenti accessibili via tastiera o altre tecnologie assistive. La funzione convertMouseEvent è il cuore dell'operazione di bypass degli

eventi. Questa funzione esamina gli eventi catturati da injectEventBypass per determinare se l'elemento target è idoneo per l'azione prevista (come determinato dalla funzione canFocus). Se l'elemento target è considerato rilevante e accessibile, convertMouseEvent tenta di convertire l'evento del mouse nell'azione corrispondente più accessibile (ad esempio, trasformando un click del mouse in un evento di focus, se appropriato). La funzione ausiliaria canFocus identific gli elementi che possono ricevere focus. Questa verifica si basa su criteri specifici, come la presenza di un attributo tabindex positivo o la natura interattiva dell'elemento (ad esempio, link con attributo href, input attivi, aree cliccabili, ecc.). canFocus assicura che solo gli elementi pertinenti e interattivi vengano considerati per la conversione degli eventi

3.4 Lista dei tag ARIA supportati

Questa è la lista di tutti gli elementi ARIA al momento supportati da Saharian:

- **Alert** Serve a indicare agli utenti delle tecnologie assistive la presenza di un messaggio importante che richiede attenzione immediata senza interrompere le attività correnti.
- **Alertdialog** Simile all'alert, ma richiede un'interazione da parte dell'utente per chiudere il dialogo.
- **Autocomplete** Indica se e come i suggerimenti di completamento automatico sono disponibili per un elemento input.
- **Button** Trasforma un elemento in un pulsante, comunicando alle tecnologie assistive la sua funzionalità cliccabile.
- **Checkbox** Identifica un elemento come una casella di controllo che può essere marcata o deselezionata.
- **Checkbox-mixed** Specifica che la casella di controllo ha uno stato intermedio, oltre ai tradizionali stati selezionato e deselezionato.
- **Collapsible** Non direttamente supportato da ARIA come attributo unico, ma può essere realizzato utilizzando combinazioni di aria-expanded per indicare se il contenuto è espanso o collassato.
- **Grid** Definisce un elemento come parte di una griglia complessa, che può contenere righe e colonne come una tabella.
- **Heading** Assegna un livello di intestazione agli elementi per aiutare a strutturare e navigare il contenuto.
- **Image** Indica che un elemento deve essere interpretato come un'immagine.
- **Link** Rende un elemento cliccabile e navigabile, simile a un collegamento ipertestuale.

- **Link-collapsible** Combina le funzionalità di link e collapsible per creare link che espandono o collassano contenuti associati.
- **List** Identifica un gruppo di elementi correlati come un elenco.
- **Listbox** Definisce un elenco di opzioni da cui gli utenti possono selezionare.
- **Listbox-activedescendant** Utilizzato per gestire la focalizzazione all'interno di un listbox senza spostare effettivamente il focus.
- **Live region** Identifica una sezione del contenuto che verrà aggiornata dinamicamente.
- **Log** Specifica una regione che riceve aggiornamenti di log, come messaggi di stato o errore.
- **None e Presentation** Indicano che un elemento non deve essere annunciato dalle tecnologie assistive, spesso utilizzato per nascondere elementi puramente decorativi.
- **Progressbar** Mostra il progresso di un'attività, come il caricamento di una pagina o il download di un file.
- **Progressbar-indeterminate** Indica che il progresso non può essere determinato come una percentuale specifica.
- **Radiogroup** Gruppa un set di controlli radio, consentendo agli utenti di selezionare una sola opzione.
- **Slider** Permette agli utenti di selezionare un valore da un intervallo.
- **Vertical Slider** Una variante dello slider che si estende verticalmente.
- **Status** Fornisce un feedback temporaneo che non richiede l'attenzione dell'utente.
- **Switch** Simula un interruttore che può essere acceso o spento.
- **Table** Organizza i dati in righe e colonne.
- **Tabs** Permette la navigazione tra gruppi di contenuti correlati all'interno dello stesso contesto.
- **Term-definition** Utilizzato per associare una definizione a un termine.
- **Timer** Indica un timer o un conto alla rovescia.
- **Toggle-button** Un pulsante che può essere attivato o disattivato.

3.5 Informazioni per programmatori interessati a modificare SAHARIAN

: Per quanto SAHARIAN sia già a un ottimo livello e possa essere utilizzato per esaminare l'accessibilità di numerosi attributi ARIA, la sua evoluzione non si è ancora conclusa. Infatti, mancano ancora implementazioni per una serie di attributi cruciali per l'accessibilità, come ad esempio:

- **aria-errormessage** Utilizzare questo attributo per indicare un messaggio di errore che è collegato a un elemento che ha fallito la validazione. SAHARIAN potrebbe visualizzare visivamente questi messaggi di errore quando l'elemento riceve focus o viene interagito, simulando come un utente con tecnologie assistive verrebbe informato dell'errore.

- **aria-flowto** Indica l'ordine di navigazione del focus dopo l'elemento corrente, bypassando il flusso del documento. SAHARIAN può evidenziare la sequenza di navigazione del focus basata su aria-flowto, facilitando agli sviluppatori la comprensione e il test dell'esperienza utente navigazionale.

- **aria-haspopup** Segnala la presenza di un popup collegato, come un menu o una finestra di dialogo. SAHARIAN potrebbe simulare l'attivazione di questi controlli e visualizzare visivamente l'area di popup quando l'elemento collegato viene attivato.

- **aria-invalid** Indica che il valore di un input non soddisfa i criteri di validazione. SAHARIAN potrebbe enfatizzare visivamente gli elementi con aria-invalid="true", aiutando gli sviluppatori a identificare e correggere problemi di validazione.

- **aria-modal** Denota un elemento come parte di una finestra di dialogo modale che limita l'interazione dell'utente al di fuori di essa. SAHARIAN potrebbe oscurare gli elementi fuori dalla modale quando questa è attiva, simulando l'esperienza modale per gli sviluppatori.

- **aria-multiline e aria-multiselectable** Specificano rispettivamente se un campo di testo accetta più righe di input e se più di un'opzione può essere selezionata. SAHARIAN può verificare e segnalare se l'implementazione di questi attributi corrisponde al comportamento previsto degli elementi.

- **aria-placeholder** Fornisce una breve hint che descrive l'input atteso in un campo di testo. SAHARIAN potrebbe assicurare che il placeholder sia accessibile e testare la sua visibilità quando il campo è vuoto.

- **aria-relevant e aria-required** Indicano quali modifiche nella regione live sono rilevanti e se un campo di input è obbligatorio. SAHARIAN potrebbe monitorare le modifiche dinamiche e verificare la presenza di attributi aria-required dove necessario.

Oltre all'integrazione di questi attributi, SAHARIAN potrebbe essere arricchito ulteriormente attraverso l'implementazione delle seguenti funzionalità:

- **Esempi Complessi di Interazione** Integrare scenari dove più componenti ARIA interagiscono tra loro, fornendo agli sviluppatori casi d'uso realistici per testare l'integrazione e l'accessibilità di componenti complessi.
- **Guida agli Errori Comuni** Implementare una funzionalità che riconosca configurazioni errate o incomplete degli attributi ARIA e fornisca suggerimenti o soluzioni specifiche per correggere tali errori.
- **Documentazione e Risorse di Apprendimento** Fornire agli sviluppatori risorse educative integrate, come esempi di codice, best practices, e linee guida per l'utilizzo ottimale degli attributi ARIA e il miglioramento generale dell'accessibilità.

Capitolo 4

Analisi dell'efficacia di SAHARIAN

Per verificare l'efficacia e il funzionamento di SAHARIAN, e per assicurarsi che rispondesse correttamente alle esigenze degli utenti, è stato deciso di condurre una serie di test con un gruppo selezionato di professionisti operanti nel settore informatico. Attraverso questa metodologia di test, è stato possibile valutare in maniera precisa l'usabilità di SAHARIAN, identificare eventuali aree di miglioramento e confermare la sua efficacia nel facilitare lo sviluppo di pagine web accessibili.

4.1 Descrizione dei test effettuati per la valutazione

4.1.1 Descrizione dell'esecuzione dei test

La pianificazione del test prevedeva una durata compresa tra i 15 e i 30 minuti, con la partecipazione attesa di 15 individui. Questa sessione di test fu condotta interamente online, organizzando i partecipanti in vari gruppi composti da almeno 5 persone ciascuno. L'organizzazione dei gruppi tenne conto dei tempi e della disponibilità di ciascun partecipante, per garantire la massima flessibilità e partecipazione.

Per quanto riguarda la descrizione dettagliata del test, i partecipanti furono suddivisi in due distinti gruppi, denominati Gruppo A e Gruppo B. Il Gruppo A fu composto da 7 partecipanti, mentre il Gruppo B ne incluse 8. A entrambi i gruppi furono forniti due set di file: il primo set consistette in 4 file HTML caratterizzati da vari errori di accessibilità che i partecipanti dovettero identificare e valutare senza fare affidamento su SAHARIAN. Il secondo set, invece, richiese ai partecipanti di affrontare e risolvere problemi simili utilizzando esplicitamente il tool SAHARIAN, per valutare l'efficacia dello strumento nel facilitare l'identificazione e la correzione di tali errori.

Il Gruppo B affrontò la stessa serie di esercizi ma seguì un ordine inverso nell'esecuzione dei test: i primi 4 file dovettero essere analizzati utilizzando SAHARIAN, mentre per gli ultimi 4 file i partecipanti dovettero procedere senza l'ausilio dello strumento. Questo approccio incrociato mirò a fornire dati comparativi sull'impatto dell'utilizzo di SAHARIAN nell'identificazione e correzione degli errori di accessibilità, oltre a valutare la capacità dei partecipanti di riconoscere e affrontare tali problemi in modo indipendente.

L'obiettivo di questa metodologia di test è duplice: da un lato, si intende verificare l'efficacia di SAHARIAN come strumento di supporto per gli sviluppatori nel migliorare l'accessibilità dei siti web; dall'altro, si cerca di sensibilizzare i partecipanti riguardo le

Figura 4.1: Prime due sezioni della pagina di test per il gruppo A

sfide comuni nell'accessibilità web e l'importanza di adottare pratiche di sviluppo inclusive. Questo test non solo contribuirà a migliorare l'usabilità di SAHARIAN, ma fornirà anche preziose intuizioni sulle competenze e sulla consapevolezza degli sviluppatori in materia di accessibilità web.

Pagine utilizzate per i test

Questa sezione illustra in dettaglio le caratteristiche e le configurazioni delle pagine testate, fornendo uno sguardo approfondito alle sfide e alle opportunità associate all'accessibilità web.

La prima pagina, destinata al Gruppo A, presenta una serie di elementi interattivi progettati per simulare comuni problematiche di accessibilità che gli sviluppatori potrebbero incontrare. Al suo interno, troviamo:

- **Campo di Input per l'Email** Questo elemento richiede all'utente di inserire un indirizzo email e, sebbene sia dotato di funzionalità di autocomplete per migliorare l'usabilità, risulta non accessibile a causa dell'assenza dell'attributo aria-autocomplete. Questa omissione impedisce alle tecnologie assistive di fornire agli utenti feedback adeguato riguardo la funzionalità di autocomplete

- **Elemento Radio Button** Questo controllo risulta inaccessibile poiché, al momento del click, modifica erroneamente l'attributo aria-checked invece di aria-selected. Questa imprecisione nella gestione degli attributi ARIA può generare confusione nelle tecnologie assistive, compromettendo la capacità dell'utente di comprendere lo stato selezionato dell'elemento.

- **Slider** A differenza degli elementi precedenti, lo slider è implementato correttamente, facendo uso dell'attributo role="slider" di ARIA. Questo consente di comunicare efficacemente alle tecnologie assistive la natura e il funzionamento dello slider

- **Elemento Radio Button**

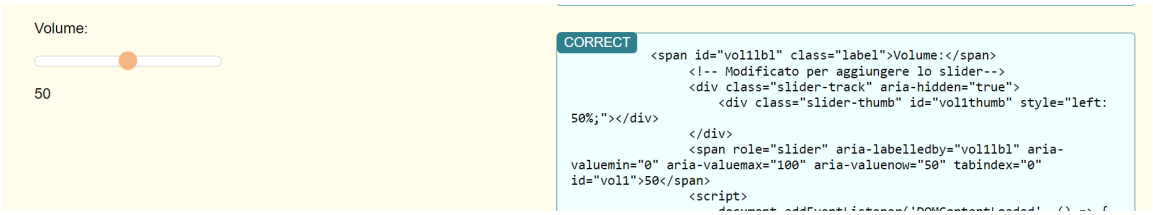

Figura 4.2: Lo slider del test A

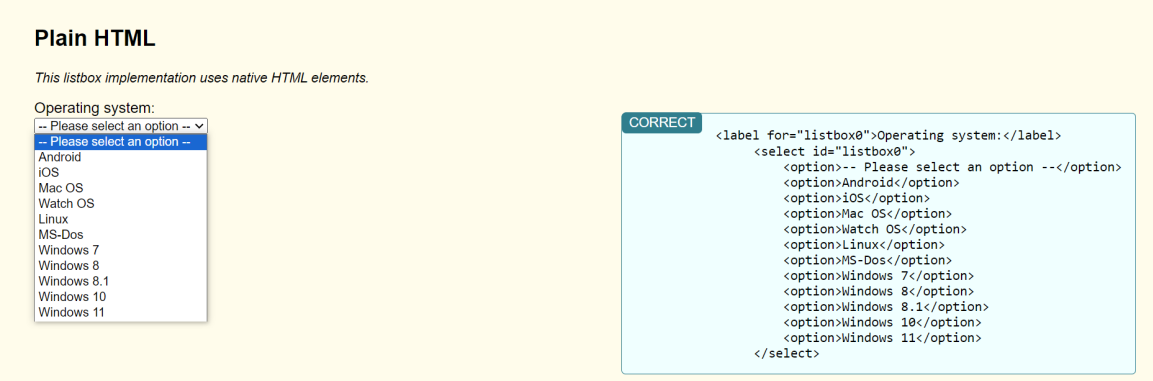

Figura 4.3: La listbox del test A

L'ultimo elemento analizzato è una listbox che, seguendo le migliori pratiche, utilizza gli elementi HTML base select e option per indicare rispettivamente la lista e i suoi elementi. Questa scelta garantisce una compatibilità nativa con le tecnologie assistive.

Proseguendo l'analisi iniziata con il Gruppo A, il Gruppo B estende l'esplorazione sull'uso di SAHARIAN per il debugging dell'accessibilità, focalizzandosi su una diversa serie di elementi web. La selezione di elementi per il Gruppo B segue logicamente il percorso intrapreso con il Gruppo A, mirando a offrire una visione ancora più ampia e dettagliata dell'impiego di SAHARIAN nel rilevamento di varie problematiche di accessibilità. Attraverso questo approccio sequenziale e complementare, si intende fornire un quadro esaustivo dell'efficacia di SAHARIAN come strumento di supporto per gli sviluppatori nell'ottimizzazione delle loro pagine web per tutti gli utenti,

- **Alert** La prima prova per il Gruppo B ha presentato un alert che, nonostante la sua funzione critica nell'avvisare gli utenti su informazioni importanti, è risultato inaccessibile a causa dell'omissione dell'attributo role="alert". Di conseguenza, SAHARIAN ha trattato l'elemento come un semplice div, senza fornire la priorità e l'urgenza tipicamente associata agli alert.

- **Checkbox** La seconda sezione del testing ha evidenziato un checkbox completamente accessibile, grazie all'impiego dell'attributo role="checkbox", della proprietà tabindex per assicurare la navigabilità tramite tastiera, e dell'utilizzo di aria-checked per indicare lo stato del controllo.

- **Link** Il terzo elemento analizzato ha riguardato un link non accessibile, reso tale dall'assenza di un appropriato role. Questa omissione ha reso l'elemento non

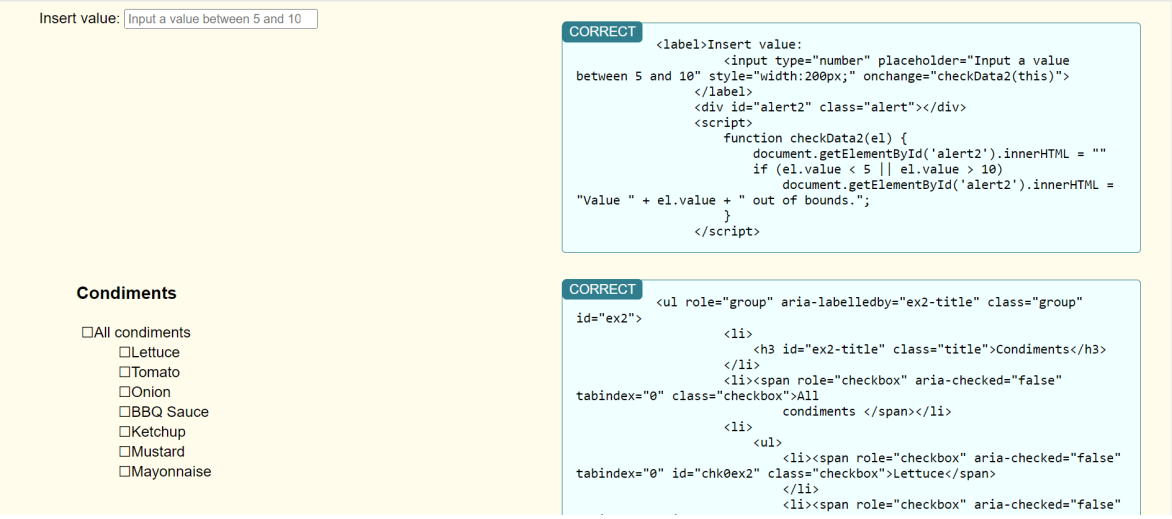

Figura 4.4: L'autocomplete e il checkbox del test B

Figura 4.5: Il link e la tablist del test B

cliccabile e non navigabile tramite tecnologie assistive, limitando la funzionalità del link e l'accesso all'informazione o alla risorsa collegata.

- **Tablist** Infine, è stata testata una tablist accessibile, caratterizzata dall'uso degli attributi role=”tablist”, aria-controls e aria-selected. Questa configurazione ha permesso agli utenti di interagire con le diverse tab in modo intuitivo, con un focus automatico sulla pagina o sulla sezione corrispondente all'attivazione della tab.

4.2 Analisi dei risultati:

Gli errori nei test con SAHARIAN per il Gruppo A sono stati principalmente attribuiti alla sezione di autocomplete, evidenziando una certa difficoltà nell'uso corretto degli attributi ARIA per questa funzionalità. Nel Gruppo B, gli errori emersi nell'uso di

SAHARIAN erano legati alla sezione di alert, segnalando un'area di miglioramento per SAHARIAN nella chiarificazione di come rappresentare accessibilmente gli alert.

- **Prestazioni senza SAHARIAN**
  - **Media delle Risposte Corrette** Gli utenti hanno ottenuto in media 1,57 risposte corrette su 4 nei test senza l'uso di SAHARIAN.
  - **Tempo Medio Impiegato** 13,4 minuti è stato il tempo medio impiegato per completare i test senza SAHARIAN.
- **Prestazioni con SAHARIAN**
  - **Media delle Risposte Corrette** Con l'uso di SAHARIAN, la media delle risposte corrette è salita a 3,73 su 4.
  - **Tempo Medio Impiegato** L'introduzione di SAHARIAN ha ridotto il tempo medio a 10,2 minuti, indicando un miglioramento nell'efficienza del processo di debugging
- L'uso di SAHARIAN ha migliorato significativamente le prestazioni dei partecipanti nell'identificare e correggere gli errori di accessibilità, con un incremento medio del 138% nelle risposte corrette.
- La riduzione del tempo medio di completamento del 24% con SAHARIAN sottolinea l'efficacia dello strumento nel rendere il processo di debugging più rapido e intuitivo.
- La discrepanza nelle aree di difficoltà tra i gruppi A e B con l'uso di SAHARIAN evidenzia l'importanza di fornire istruzioni chiare e esempi specifici per l'uso degli attributi ARIA, specialmente per funzionalità complesse come l'autocomplete e gli alert.

4.3 Valutazione complessiva

Tabella 4.1: Risultati test gruppo A

	TEST A NO SAHARIAN	TEMPO		TEST B SAHARIAN	TEMPO
User 1	2 su 4 corrette	15 min	User 1	3 su 4 corrette	11 min
User 2	0 su 4 corrette	13 min	User 2	3 su 4 corrette	9 min
User 3	4 su 4 corrette	15 min	User 3	4 su 4 corrette	9 min
User 4	1 su 4 corrette	15 min	User 4	4 su 4 corrette	15 min
User 5	3 su 4 corrette	13 min	User 5	4 su 4 corrette	10 min
User 6	1 su 4 corrette	14 min	User 6	3 su 4 corrette	9 min
User 7	0 su 4 corrette	13 min	User 7	4 su 4 corrette	8 min

I test sono stati condotti su un campione di 15 individui, con età compresa tra i 21 e i 35 anni. Questo gruppo eterogeneo comprendeva studenti di informatica, professionisti del settore e individui con esperienza nel campo dell'informatica. Tutti i

Tabella 4.2: Risultati test gruppo B

	TEST B NO SAHARIAN	TEMPO		TEST A SAHARIAN	TEMPO
User 8	4 su 4 corrette	12 min	User 8	4 su 4 corrette	5 min
User 9	3 su 4 corrette	14 min	User 9	4 su 4 corrette	9 min
User 10	1 su 4 corrette	10 min	User 10	4 su 4 corrette	8 min
User 11	0 su 4 corrette	15 min	User 11	4 su 4 corrette	15 min
User 12	3 su 4 corrette	15 min	User 12	3 su 4 corrette	14 min
User 13	0 su 4 corrette	15 min	User 13	4 su 4 corrette	13 min
User 14	3 su 4 corrette	12 min	User 14	4 su 4 corrette	8 min
User 15	4 su 4 corrette	13 min	User 15	4 su 4 corrette	9 min

partecipanti hanno descritto la loro conoscenza dello sviluppo web come "almeno nella media" o superiore, mentre la loro familiarità con i principi dell'accessibilità variava da "medio-bassa" a "quasi nulla". Questa diversità di background ha fornito una base solida per valutare l'impatto di SAHARIAN sul miglioramento della comprensione e dell'implementazione dell'accessibilità web, i primi 7 utenti hanno testato la pagina A senza SAHARIAN e la pagina B con SAHARIAN, mentre gli altri 8 utenti hanno seguito l'ordine inverso, offrendo così una visione bilanciata dell'effetto dello strumento.

I risultati del testing evidenziano un'efficacia notevole di SAHARIAN nell'assistere gli sviluppatori nell'identificazione e nella correzione di problemi di accessibilità. Quando SAHARIAN è stato utilizzato, la maggior parte dei partecipanti ha mostrato un miglioramento significativo nella capacità di rilevare e risolvere gli errori di accessibilità, con un aumento delle risposte corrette e una riduzione del tempo impiegato per completare i test.

- **Effetto SAHARIAN sulla Precisione** L'uso di SAHARIAN ha portato a un incremento notevole delle performance dei partecipanti del test, gli unici errori avvenuti sono stati nella pagina B. dove gli errori sono stati principalmente associati alla mancanza di chiarezza nella rappresentazione degli alert, suggerendo un'area di miglioramento per l'interfaccia di SAHARIAN in termini di guidare gli utenti nella corretta interpretazione e implementazione degli attributi ARIA per gli alert.

- **Effetto SAHARIAN sulla Precisione**

- **Tempo di Completamento** In generale, l'impiego di SAHARIAN ha ridotto il tempo necessario per completare i test, indicando che lo strumento non solo facilita l'identificazione degli errori ma anche contribuisce a un processo di debugging più rapido ed efficiente.

- **Comprensione dell'Accessibilità:** L'errore riscontrato nella sezione A durante l'uso di SAHARIAN, associato all'esercizio con l'autocomplete, sottolinea l'importanza di una guida esplicita su come implementare correttamente gli attributi ARIA per funzionalità avanzate come l'autocomplete. Questo evidenzia un'opportunità per SAHARIAN di includere esempi pratici o suggerimenti specifici per migliorare la comprensione degli sviluppatori riguardo l'accessibilità.

- **Miglioramento Generale con SAHARIAN** L'impiego di SAHARIAN ha evidenziato un impatto positivo marcato sull'abilità dei partecipanti di affrontare e risolvere questioni di accessibilità. Analizzando i dati raccolti, si nota che l'uso di SAHARIAN ha significativamente influenzato i risultati dei partecipanti in maniera positiva. In dettaglio, per i test effettuati sulla Sezione B con l'ausilio di SAHARIAN, si è osservato un netto miglioramento delle prestazioni: i partecipanti hanno incrementato notevolmente il numero di risposte corrette, passando da una media preliminare di circa 1 su 4 a una media di 3,73 su 4 risposte esatte grazie all'uso dello strumento. Questo salto qualitativo nelle prestazioni non solo evidenzia l'efficacia di SAHARIAN nel rendere gli standard di accessibilità più comprensibili e applicabili per gli sviluppatori, ma sottolinea anche come il supporto mirato possa facilitare l'identificazione e la correzione di errori di accessibilità in modo più intuitivo e veloce. L'analisi dei tempi di completamento dei test rafforza ulteriormente questo punto, mostrando una riduzione media del tempo impiegato per l'esecuzione dei test quando SAHARIAN è stato utilizzato, segnalando un miglioramento dell'efficienza nel processo di debugging dell'accessibilità web.

- **Diminuzione dei Tempi di Completo** L'introduzione di SAHARIAN ha anche portato a una riduzione dei tempi di completamento dei test. I partecipanti hanno impiegato in media 2-6 minuti in meno per completare le sezioni del test con l'ausilio di SAHARIAN rispetto a quelle senza.

Conclusioni

Attraverso l'analisi dei risultati ottenuti dal testing con SAHARIAN, emerge chiaramente il potenziale di questo strumento nel migliorare significativamente la capacità degli sviluppatori di identificare e correggere le problematiche di accessibilità nelle pagine web. I risultati hanno dimostrato un notevole miglioramento nelle prestazioni dei partecipanti quando hanno utilizzato SAHARIAN, evidenziando un aumento medio delle risposte corrette dal 25% senza SAHARIAN al 93,5% con l'utilizzo dello strumento. Questo incremento sottolinea non solo l'efficacia di SAHARIAN nel facilitare una comprensione più approfondita degli standard di accessibilità ma anche nel rendere il processo di debugging più intuitivo e accessibile. SAHARIAN ha dimostrato di essere uno strumento prezioso per gli sviluppatori, fornendo un mezzo diretto e interattivo per valutare e migliorare l'accessibilità dei loro siti web. Attraverso l'uso di SAHARIAN, gli sviluppatori possono ottenere un feedback immediato sugli errori di accessibilità, permettendo loro di apportare modifiche in tempo reale e vedere gli effetti di quelle modifiche sull'accessibilità complessiva del sito. Questa immediatezza e praticità nel processo di sviluppo non solo aumentano l'efficienza ma anche la qualità del lavoro prodotto, garantendo che i siti web siano progettati con un'attenzione particolare alle esigenze di tutti gli utenti, compresi coloro che utilizzano tecnologie assistive.

Un aspetto fondamentale emerso dall'uso di SAHARIAN è la sua capacità di sfidare e, allo stesso tempo, educare gli sviluppatori sull'importanza dell'accessibilità web. Attraverso il testing e il feedback forniti da SAHARIAN, gli sviluppatori vengono esposti a concetti di accessibilità che potrebbero non essere stati considerati in precedenza, promuovendo una maggiore consapevolezza e comprensione dell'accessibilità come componente cruciale del processo di sviluppo web. Questo processo educativo è vitale per garantire che l'accessibilità diventi una considerazione standard e non un'aggiunta tardiva o un compito da spuntare.

Sebbene SAHARIAN abbia già mostrato un grande potenziale, i risultati del testing suggeriscono anche aree per miglioramenti futuri. Per esempio, l'integrazione di guide più dettagliate o esempi di codice per la gestione di attributi ARIA specifici potrebbe aiutare gli sviluppatori a evitare errori comuni e a implementare pratiche di accessibilità più efficaci. Inoltre, l'espansione del database di errori di SAHARIAN per coprire una gamma più ampia di problemi di accessibilità potrebbe fornire agli sviluppatori una comprensione ancora più completa delle sfide legate all'accessibilità.

In conclusione, SAHARIAN rappresenta un passo significativo verso la creazione di un web più accessibile, offrendo agli sviluppatori uno strumento efficace per identificare e correggere gli errori di accessibilità. I risultati ottenuti dal suo utilizzo sottolineano l'importanza di strumenti di questo tipo nel processo di sviluppo web, evidenziando sia il miglioramento delle competenze degli sviluppatori sia l'aumento dell'accessibilità

complessiva dei siti web. Continuando a sviluppare e perfezionare SAHARIAN, basandosi sui feedback degli utenti e sull'analisi dei dati di utilizzo, possiamo aspettarci non solo un miglioramento delle funzionalità dello strumento ma anche un avanzamento significativo nella realizzazione di un internet accessibile a tutti.

Bibliografia

[AAV19] Julio Abascal, Myriam Arrue, and Xabier Valencia. Tools for web accessibility evaluation. In *Web Accessibility*, pages 479–503. Springer, 2019.

[Acca] Accessibility Guidelines Working Group (AGWG). Techniques for wcag 2.1. World Wide Web Consortium (W3C).

[Accb] Accessibility Guidelines Working Group (AGWG). Understanding web content accessibility guidelines (wcag) 2.1. World Wide Web Consortium (W3C).

[Acc22] Accessibility Guidelines Working Group (AGWG). Essential components of web accessibility. Technical report, World Wide Web Consortium (W3C), 2022.

[Acc23] Accessi. European accessibility act, 2023. https://www.accessi.org/blog/european-accessibility-act/.

[ACD+18] Glenn Adams, Cyril Concolato, Mike Dolan, Sean Hayes, Frans de Jong, Dae Kim, Pierre-Anthony Lemieux, Nigel Megitt, Dave Singer, Jerry Smith, and Andreas Tai. Timed text markup language 2 (ttml2). Technical report, World Wide Web Consortium (W3C), November 2018.

[ACM+22] Chuck Adams, Alastair Campbell, Rachael Montgomery, Michael Cooper, and Andrew Kirkpatrick. Web content accessibility guidelines (wcag) 2.2. Technical report, World Wide Web Consortium (W3C), September 2022.

[Ada22] Paul J. Adam. A11ytools extension for safari, 2022. Last accessed on December 6, 2022.

[AFGM10] Fernando Alonso, José Luis Fuertes, Ángel Lucas González, and Loïc Martínez. On the testability of wcag 2.0 for beginners. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, pages 1–9, 2010.

[Ala22] Nancy Alajarmeh. The extent of mobile accessibility coverage in wcag 2.1: Sufficiency of success criteria and appropriateness of relevant conformance levels pertaining to accessibility problems encountered by users who are visually impaired. *Universal Access in the Information Society*, 21(2):507–532, 2022.

[ALPS15] James Allan, Greg Lowney, Kim Patch, and Jeanne Spellman. User agent accessibility guidelines (uaag) 2.0. Technical report, World Wide Web Consortium (W3C), December 2015. Last accessed December 06, 2022.

[AVA08] Myriam Arrue, Markel Vigo, and Julio Abascal. Including heterogeneous web accessibility guidelines in the development process. In *IFIP International Conference on Engineering for Human-Computer Interaction*, pages 620–637. Springer, 2008.

[BRDC+18] Amelia Bellamy-Royds, Joanmarie Diggs, Michael Cooper, Fred Esch, Rich Schwerdtfeger, and Doug Schepers. Graphics accessibility api mappings. Technical report, World Wide Web Consortium (W3C), October 2018. Last accessed December 06, 2022.

[Eur23] European Commission. European accessibility act, 2023.

[Goo] Google Chrome Developers. Lighthouse: An open-source, automated tool for improving the quality of web pages. Web Page.

[Hara] Harvard University, Information Technology. Jaws. Web Page.

[Harb] Harvard University, Information Technology. Nvda. Web Page.

[Mor18] Luca Morosini. Saharian: Uno strumento visuale per la progettazione di pagine web accessibili, 2017/2018.

[Moz] Mozilla Developer Network (MDN). Webextensions.

[Rub23] Vincenzo Rubano. On making web accessibility more accessible: Strategy and tools for social good, 2022/2023.

[U.S] U.S. Department of Veterans Affairs, Office of Information and Technology. Technology reference model (trm). Web Page.

[Web] WebAIM. Wave web accessibility evaluation tool.

[Wor18] World Wide Web Consortium (W3C). Web content accessibility guidelines (wcag) 2.1, 2018.

[Wor23] World Wide Web Consortium (W3C). Wai-aria overview, 2023.

Ringraziamenti

Ringrazio il Prof Vitali per la disponibilità e per la pazienza, ringrazio i miei correlatori Caterina e Vincenzo per il costante aiuto e supporto, ringrazio soprattutto mia madre, mio padre ed il resto della mia famiglia per essere stati sempre al mio fianco durante questo percorso, sia nei momenti migliori che nei peggiori, infine ringrazio mia moglie, la mia compagna di vita