ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Scienze

Corso di Laurea in Informatica

Miglioramenti di Saharian: Un Approccio Pratico per l'Accessibilità Web

Relatore:
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Emanuele Pischetola

Sessione marzo 2025
Anno Accademico 2023/2024

Abstract

Nel contesto dello sviluppo web, l'accessibilità rappresenta una priorità imprescindibile, volta a garantire che i contenuti digitali siano fruibili da tutti gli utenti, comprese le persone che utilizzano tecnologie assistive. Questa tesi si concentra sul miglioramento e sull'estensione di Saharian, uno strumento progettato per agevolare la valutazione dell'accessibilità delle pagine web. L'obiettivo del lavoro è completarne l'implementazione, ampliandone le capacità di analisi e rendendolo un supporto ancora più efficace nel campo dell'accessibilità web. L'attività di ricerca si articola in diverse fasi operative: un'analisi approfondita delle funzionalità esistenti di Saharian, l'individuazione delle aree di miglioramento, l'integrazione di nuove caratteristiche e la verifica della loro efficacia. Particolare attenzione è dedicata all'aggiunta del supporto per gli attributi ARIA attualmente non gestiti da Saharian, fondamentali per una corretta interpretazione dei contenuti da parte delle tecnologie assistive. Inoltre, vengono esplorate strategie per ottimizzarne le prestazioni nell'elaborazione di pagine web complesse, affrontando la sfida dei tempi di elaborazione necessari per generare versioni accessibili di siti ricchi di contenuti. L'approccio seguito è di natura pratica e si basa su un processo iterativo di test e valutazione, attraverso cui le nuove funzionalità vengono progressivamente implementate e validate. La verifica delle modifiche apportate a Saharian avviene in scenari reali di navigazione web, consentendo di misurare concretamente l'impatto degli interventi sull'esperienza utente delle persone con disabilità. Il contributo di questa tesi non si limita all'ampliamento delle funzionalità di Saharian, ma mira anche a sensibilizzare la comunità degli sviluppatori sull'importanza dell'accessibilità web. In conclusione, il perfezionamento e l'ampliamento di Saharian rappresentano un passo significativo verso un web più accessibile, offrendo agli sviluppatori uno strumento più potente e versatile per garantire la fruibilità dei loro siti a una vasta gamma di utenti, indipendentemente dalle loro esigenze specifiche.

Indice

Capitolo 1

Introduzione

L'accesso alle informazioni è un diritto fondamentale nell'era digitale in cui viviamo. Il web, con la sua enorme quantità di dati e servizi, è diventato una componente essenziale della vita quotidiana per molte persone. Tuttavia, non tutti possono navigarlo e utilizzarlo con la stessa facilità. 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 difficoltà che gli utenti non vedenti devono affrontare.

L'accessibilità web è un tema complesso e multidimensionale, che coinvolge diversi aspetti della progettazione e dello sviluppo dei siti. Le problematiche spaziano da questioni tecniche, come il rispetto degli standard di accessibilità, a questioni pratiche, come la formazione degli sviluppatori e la sensibilizzazione sull'importanza di progettare per tutti. Questa ricerca si inserisce nel dibattito scientifico e tecnologico attuale, evidenziando la discrepanza tra le linee guida di accessibilità e la loro effettiva applicazione nei siti web. Nonostante l'aumento della consapevolezza sull'importanza dell'accessibilità, gli strumenti disponibili per supportarne l'implementazione risultano spesso complessi e poco intuitivi per gli sviluppatori.

Questo lavoro si basa su precedenti ricerche e implementazioni nel campo dell'accessibilità web, tra cui la tesi "SahARIAn: Uno strumento visuale per la progettazione di pagine web accessibili" di Luca Morosini (2017/2018) [Mor18] e gli approfondimenti successivi contenuti in "On Making Web Accessibility More Accessible: Strategy and Tools for Social Good" di Vincenzo Rubano (2022/2023) [Rub23]. Quest'ultima ricerca ha contribuito allo sviluppo di Saharian ed esplorato altri strumenti dedicati all'accessibilità, come l'AX framework e A11A [Rub25], favorendo una maggiore comprensione e diffusione delle migliori pratiche in questo ambito.

Un ulteriore contributo è stato dato dalla tesi "Raffinamento e Ampliamento di Saharian: Un Contributo allo Sviluppo di Strumenti per l'Accessibilità Web" di Anselmo Ferrari (2023/2024) [Fer24].

In questo contesto, la presente tesi si propone di completare il percorso di sviluppo del tool Saharian, ampliandone le funzionalità già introdotte grazie al lavoro di Morosini, Rubano e Ferrari.

Saharian viene qui presentato come uno strumento avanzato per l'analisi dell'accessibilità delle pagine web. Tramite la rimozione degli stili CSS, fornisce agli sviluppatori una rappresentazione semplificata del contenuto, simile a quella percepita dalle tecnologie assistive. Questo approccio permette di individuare e correggere le problematiche di accessibilità con maggiore efficacia, favorendo una progettazione web realmente inclusiva.

L'efficacia di Saharian è stata verificata attraverso un processo di valutazione che ha coinvolto direttamente gli sviluppatori web nell'utilizzo dello strumento. Il feedback raccolto ha evidenziato il ruolo chiave di Saharian nel rendere l'accessibilità un obiettivo più facilmente raggiungibile e nel migliorare significativamente l'esperienza degli utenti non vedenti sul web.

Nel contesto dell'accessibilità digitale, l'European Accessibility Act (EAA) [acc21; Com19] rappresenta una direttiva fondamentale dell'Unione Europea, entrata in vigore nell'aprile 2019 con l'obiettivo di armonizzare le normative tra gli stati membri e facilitare la commercializzazione di prodotti e servizi accessibili. Grazie a regole comuni, le aziende possono partecipare più facilmente al commercio transfrontaliero, favorendo un mercato più ampio e competitivo. Inoltre, l'EAA si propone di ridurre le barriere digitali, garantendo che persone con disabilità e anziani possano usufruire di tecnologie più accessibili. Questa direttiva si integra con la Web Accessibility Directive del 2016 e riflette gli impegni assunti nella Convenzione delle Nazioni Unite sui Diritti delle Persone con Disabilità, coprendo un'ampia gamma di tecnologie, dai dispositivi personali ai servizi pubblici.

Nel panorama digitale odierno, l'accessibilità web non è solo un'esigenza etica, ma anche un obbligo giuridico che impone standard minimi per garantire che i contenuti digitali siano fruibili da tutti, indipendentemente dalle capacità fisiche degli utenti. In questo contesto, gli standard WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) e WCAG (Web Content Accessibility Guidelines) hanno rappresentato un passo importante verso un web più inclusivo. Tuttavia, nonostante i loro punti di forza, presentano alcune limitazioni, tra cui la complessità di implementazione, la variabilità del supporto tra browser e dispositivi, l'ambiguità di alcune linee guida e la lentezza nell'aggiornamento degli standard. Queste difficoltà costituiscono una sfida concreta per gli sviluppatori.

Il progetto Saharian nasce con l'obiettivo di affrontare le problematiche legate all'accessibilità web, fornendo agli sviluppatori uno strumento in grado di mostrare come le tecnologie assistive interpretano le pagine web. Grazie a questo approccio, è possibile individuare rapidamente le aree che necessitano di miglioramenti, semplificando così il processo di creazione di siti più accessibili e inclusivi.

Allo stato iniziale del progetto, Saharian era in grado di riconoscere solo una parte degli attributi ARIA, senza offrire un supporto completo. Questa limitazione rappresentava un ostacolo per gli sviluppatori, poiché impediva loro di avere una visione chiara e completa dell'accessibilità della propria pagina. In alcuni casi, l'aggiunta di un attributo ARIA sembrava non avere alcun impatto visibile, portando a confusione e possibili implementazioni errate. In particolare, alcuni ruoli ARIA non erano ancora supportati, il che riduceva l'efficacia dello strumento. I ruoli ARIA [W3C23b] sono attributi che assegnano una determinata semantica agli elementi del DOM, specificando il loro comportamento e la loro funzione per le tecnologie assistive. Ad esempio, un ruolo come `alert` segnala ai lettori di schermo la presenza di un messaggio importante, mentre un ruolo `dialog` indica la presenza di una finestra modale. I ruoli mancanti in Saharian erano principalmente secondari, cioè meno comuni nella maggior parte delle pagine web. Tuttavia, anche se meno diffusi, questi ruoli possono essere fondamentali in contesti specifici, ad esempio per applicazioni web complesse o strumenti interattivi avanzati. Oltre ai ruoli, un'altra carenza individuata nel progetto iniziale riguardava la mancanza di supporto per alcuni stati e proprietà ARIA [W3C23a]. Gli stati ARIA servono a definire la condizione attuale di un elemento (ad esempio, `aria-checked="true"` per indicare una casella di selezione attiva), mentre le proprietà ARIA forniscono informazioni aggiuntive sugli elementi (ad esempio, `aria-labelledby` per associare un'etichetta a un componente). L'assenza di questi elementi limitava la capacità di Saharian di offrire un quadro completo dello stato accessibile della pagina.

L'obiettivo principale di questa tesi è stato ampliare il supporto agli attributi ARIA, portandolo il più vicino possibile al 100%. Un obiettivo secondario, ma altrettanto rilevante, è stato lo sviluppo di una modalità avanzata di Saharian, che permette agli sviluppatori di comprendere in maniera più dettagliata il funzionamento di alcuni attributi ARIA più complessi. Questa modalità avanzata era stata inizialmente solo abbozzata, senza una chiara regolazione su quali attributi ARIA dovessero influenzarla e in che modo. Durante lo sviluppo del progetto di tesi, è stato necessario ridefinire e stabilizzare questa funzione, garantendo che l'interazione tra gli attributi ARIA e la pagina web fosse rappresentata in modo coerente e fedele alla realtà.

Un'altra area di intervento ha riguardato la pagina di troubleshooting, che in origine risultava incompleta e priva di contenuti significativi. Questa sezione era stata concepita per offrire esempi pratici di implementazione di componenti web accessibili,

ma inizialmente conteneva soltanto un indice vuoto, senza informazioni dettagliate. Durante il progetto di tesi, la pagina è stata arricchita con contenuti strutturati e approfonditi, rendendola una risorsa utile per gli sviluppatori che vogliono migliorare la conformità delle proprie applicazioni alle linee guida di accessibilità.

Per garantire la correttezza e l'efficacia delle modifiche apportate, erano state create pagine HTML di esempio, ciascuna dedicata a un particolare widget web. Queste pagine includono sia esempi di utilizzo corretto che esempi di errori comuni, permettendo di confrontare i diversi approcci e verificare in che modo Saharian interpreta le varie configurazioni. Tuttavia, all'inizio del progetto, alcuni esempi contenevano errori, oppure Saharian non si comportava come previsto, segnalando problemi di interpretazione degli attributi ARIA.

Grazie alle modifiche implementate, il tool è ora in grado di riconoscere e gestire correttamente tutti i ruoli ARIA, con l'eccezione di quelli astratti. Per ciascun ruolo è stato associato uno stile specifico o uno script che ne modifica il comportamento, in modo da fornire un feedback visivo chiaro agli sviluppatori. Inoltre, sono stati implementati la maggior parte degli stati e delle proprietà ARIA, raggiungendo un elevato livello di copertura, sebbene non ancora completo al 100%. Per ogni nuovo attributo supportato, è stato creato un esempio pratico, inserendolo in una pagina di test o ampliando un esempio esistente. Questo approccio ha permesso di garantire che ogni nuova funzionalità fosse testata e verificata, migliorando così la robustezza dello strumento.

Il lavoro svolto in questa tesi ha contribuito in modo significativo a migliorare le funzionalità di Saharian, trasformandolo in uno strumento più completo e affidabile per gli sviluppatori web. Oltre all'ampliamento del supporto per gli attributi ARIA, il progetto ha portato alla creazione di una modalità avanzata, all'arricchimento della pagina di troubleshooting e alla realizzazione di una serie di esempi pratici per facilitare l'apprendimento e l'adozione di pratiche accessibili. Attualmente, Saharian rappresenta una risorsa utile per chi sviluppa pagine web e desidera testarne l'accessibilità in modo semplice ed efficace.

Grazie a questi sviluppi, il progetto può contribuire a diffondere la cultura dell'accessibilità web, favorendo la creazione di esperienze digitali più inclusive e rispettose delle esigenze di tutti gli utenti.

Capitolo 2

Standard e strumenti per un web più accessibile

2.1 Evoluzione dell'accessibilità web, approcci passati e loro limiti

Nel contesto dell'accessibilità web, gli approcci adottati fino a oggi hanno portato a significativi progressi, ma presentano ancora limitazioni che richiedono un'analisi critica. Tradizionalmente, gli sforzi si sono focalizzati sulla conformità a standard come le Web Content Accessibility Guidelines (WCAG)[W3C24b], un framework strutturato per rendere i contenuti web accessibili a persone con diverse disabilità. Tuttavia, attenersi a queste linee guida non garantisce necessariamente un'esperienza utente realmente accessibile e intuitiva, soprattutto per chi ha disabilità visive. Uno dei principali limiti dei metodi precedenti è l'eccessiva dipendenza da controlli manuali e revisioni periodiche. Questo approccio reattivo non tiene conto della natura dinamica dei contenuti web, con il rischio che le pagine diventino rapidamente non conformi agli standard di accessibilità non appena vengono aggiornate o modificate. Inoltre, la verifica manuale dell'accessibilità richiede tempo e risorse considerevoli, rendendola poco sostenibile su larga scala. Un altro aspetto critico riguarda l'interpretazione delle WCAG [W3C24b]: la loro formulazione talvolta astratta e la complessità dei criteri possono risultare difficili da comprendere per gli sviluppatori web meno esperti in materia di accessibilità. Di conseguenza, molte implementazioni risultano incomplete o fraintese, limitandosi a soddisfare i requisiti minimi senza apportare reali benefici all'esperienza utente delle persone con disabilità. Gli strumenti per la verifica dell'accessibilità web svolgono un ruolo essenziale nel garantire che i siti siano conformi alle linee guida e accessibili a tutti gli utenti,

comprese le persone con disabilità. Sebbene questi strumenti offrano un supporto prezioso, tendono a privilegiare gli aspetti tecnici, trascurando talvolta l'usabilità e l'esperienza utente. Nonostante le loro limitazioni, esistono diversi strumenti che aiutano sviluppatori e designer a identificare e correggere problemi di accessibilità. Ecco alcuni degli strumenti più utilizzati [Tea24; W3C25b]:

- **Google Lighthouse** [Goo25]
  Strumento di analisi integrato in Chrome DevTools, Lighthouse genera un report sull'accessibilità di una pagina, oltre a metriche su prestazioni, SEO e best practices. Sebbene offra suggerimenti utili per migliorare l'accessibilità, le sue valutazioni non sempre considerano l'esperienza utente o l'usabilità in modo completo.

- **Wave Tool** [Web25]
  Un altro strumento utile è Wave, che offre una rappresentazione grafica delle potenziali problematiche di accessibilità direttamente nella pagina web analizzata. Wave evidenzia errori, avvertimenti e caratteristiche accessibili, permettendo agli sviluppatori di individuare le aree che necessitano miglioramenti. Tuttavia, presenta limiti nell'analisi della comprensibilità del testo e nell'efficacia della navigazione per gli utenti con disabilità.

- **AXE Browser Extension** [Deq25]
  Integrabile direttamente nei browser come Chrome e Firefox, AXE consente di analizzare rapidamente le pagine web per individuare violazioni delle WCAG (Web Content Accessibility Guidelines). Fornisce dettagli sugli errori rilevati e suggerimenti per risolverli, concentrandosi principalmente su aspetti tecnici, come l'uso corretto dei tag semantici e le alternative testuali per le immagini.

- **JAWS (Job Access With Speech)** [Sci25]
  Si tratta di un lettore di schermo ampiamente utilizzato da utenti non vedenti o ipovedenti. JAWS consente la navigazione web tramite comandi vocali e legge ad alta voce i contenuti delle pagine. È utile per testare l'esperienza utente di chi utilizza lettori di schermo, ma per interpretare correttamente i risultati è necessaria una certa familiarità con il software.

- **NVDA (NonVisual Desktop Access)** [Acc25]
  Un lettore di schermo gratuito e open-source per Windows. Simile a JAWS, NVDA permette di comprendere come gli utenti non vedenti interagiscono con i siti web. Fornisce una prospettiva sull'accessibilità delle pagine dal punto di vista di chi si affida ai lettori di schermo, ma, come JAWS, richiede competenze specifiche per essere utilizzato efficacemente.

Un'altra criticità riguarda la scarsa integrazione dell'accessibilità nella progettazione

e nello sviluppo web. Spesso viene trattata come un'aggiunta successiva, anziché come un elemento essenziale del processo di sviluppo. Questo approccio frammentato porta a soluzioni applicate superficialmente all'interfaccia esistente, invece di essere incorporate in modo organico nella progettazione del sito.

In conclusione, sebbene gli approcci attuali abbiano gettato le basi per un web più inclusivo, persistono limiti significativi che ostacolano il raggiungimento di questo obiettivo. Una riflessione critica su queste problematiche è essenziale per sviluppare soluzioni più efficaci, efficienti e integrate, come quelle proposte in questa tesi con lo strumento Saharian.

2.2 Standard ARIA: Strumento essenziale o ostacolo nascosto?

L'Accessible Rich Internet Applications (ARIA) [W3C22] è uno standard fondamentale nel campo dell'accessibilità web, il cui ruolo è diventato sempre più rilevante con l'evoluzione delle applicazioni web moderne. Sviluppato dal Web Accessibility Initiative (WAI) del World Wide Web Consortium (W3C) [W3C24a], ARIA mira a migliorare l'accessibilità dei contenuti dinamici e delle applicazioni web complesse per le persone con disabilità. In particolare, mette a disposizione un insieme di attributi specifici che possono essere integrati nel codice HTML per facilitare l'interazione con le tecnologie assistive, come i lettori di schermo.

```
<div role="region"
     aria-labelledby="titolo"
     aria-describedby="descrizione"
     aria-live="polite">
    <span id="titolo" role="heading" aria-level="2">
        Sezione Aggiornamenti
    </span>
    <p id="descrizione">
        Contenuto dinamico e aggiornato in tempo reale.
    </p>
</div>
```

Listing 1: Esempio di frammento HTML che utilizza attributi ARIA

Uno degli aspetti chiave di ARIA è la capacità di descrivere elementi dell'interfaccia utente che il solo HTML standard non riesce a rappresentare in modo efficace. Ad esempio, componenti interattivi come menu a tendina, barre di navigazione o altri elementi dinamici possono risultare difficili da interpretare per le tecnologie assisti-

ve. ARIA interviene con attributi che definiscono il ruolo, lo stato e le proprietà di questi elementi, consentendo ai lettori di schermo di comunicarne le funzionalità in maniera chiara e comprensibile. Inoltre, introduce il concetto di "landmark regions", che aiuta gli utenti di tecnologie assistive a orientarsi più facilmente all'interno di una pagina, identificando sezioni fondamentali come intestazione, piè di pagina, navigazione principale e contenuto principale. Questo approccio migliora significativamente l'esperienza di navigazione per le persone con disabilità visive.

Tuttavia, l'efficacia di ARIA dipende dalla sua corretta implementazione [fou25]. Un utilizzo errato degli attributi può generare confusione e compromettere l'accessibilità, invece di migliorarla. È quindi essenziale che gli sviluppatori web conoscano a fondo lo standard e lo applichino con precisione. Inoltre, è importante restare aggiornati sulle nuove versioni e sulle best practice emergenti, poiché il mondo delle tecnologie web e dell'accessibilità è in continua evoluzione.

In questo contesto, Saharian si rivela un supporto prezioso per gli sviluppatori, aiutandoli a implementare correttamente ARIA. Grazie all'analisi automatica delle pagine web, il sistema è in grado di individuare errori nell'uso degli attributi ARIA e suggerire miglioramenti, contribuendo così a rendere il web più accessibile e inclusivo.

Gli attributi ARIA offrono strumenti potenti per migliorare l'accessibilità delle pagine web. Tra i più utilizzati ci sono gli attributi `role`, che definiscono la funzione specifica di un elemento all'interno della pagina. Ad esempio, assegnare `role="button"` a un elemento indica che questo si comporta come un pulsante, mentre `role="navigation"` segnala che fa parte della struttura di navigazione del sito. Queste informazioni sono cruciali per le tecnologie assistive, poiché forniscono un contesto chiaro sull'utilizzo degli elementi, facilitando l'interazione per le persone con disabilità visive.

Un altro aspetto rilevante è l'uso di `aria-label` e `aria-labelledby`, attributi che permettono di fornire descrizioni testuali aggiuntive agli elementi, soprattutto nei casi in cui il testo visibile non sia sufficiente a chiarire il loro scopo. Ad esempio, un'icona a forma di lente di ingrandimento per la ricerca potrebbe avere l'attributo `aria-label="search"`, rendendo evidente la sua funzione anche a chi non può vederla.

Gli attributi `aria-hidden` e `aria-live` sono altrettanto essenziali per gestire la visibilità e l'aggiornamento dinamico dei contenuti. `aria-hidden="true"` viene utilizzato per nascondere elementi visivi irrilevanti per gli utenti dei lettori di schermo, evitando confusione. D'altra parte, `aria-live` serve a indicare quando e come gli utenti devono essere informati su aggiornamenti dinamici della pagina, come notifiche o messaggi di stato.

Anche l'attributo `tabindex` gioca un ruolo chiave nell'accessibilità, permettendo di controllare l'ordine di navigazione tramite tastiera. Impostando `tabindex="0"`, si assicura che l'elemento possa essere raggiunto con il tasto Tab, mentre con `tabindex="-1"` l'elemento rimane accessibile solo tramite script, senza interferire con la navigazione sequenziale. Questa funzionalità è particolarmente utile per creare un flusso di navigazione logico e intuitivo, soprattutto in applicazioni web complesse. Tuttavia, un uso improprio di ARIA o di `tabindex` può compromettere l'usabilità: un ordine di navigazione confuso o un eccesso di `aria-label` possono sovraccaricare l'utente con informazioni superflue. Per questo motivo, è fondamentale che gli sviluppatori adottino un approccio attento e testino le loro soluzioni con utenti reali.

In conclusione, gli standard ARIA rappresentano un elemento essenziale per garantire l'accessibilità delle applicazioni web moderne. La loro implementazione consapevole consente di superare molte delle sfide legate all'accessibilità, assicurando che i contenuti web siano fruibili da tutti, indipendentemente dalle capacità individuali. Per questo, sviluppatori e progettisti dovrebbero impegnarsi a comprenderli e applicarli correttamente, con l'obiettivo di creare esperienze digitali realmente inclusive.

Capitolo 3

Introduzione a Saharian

3.1 Saharian in azione, obiettivi e funzionalità chiave

Saharian è un'estensione per browser compatibile con Google Chrome, Mozilla Firefox e Microsoft Edge, progettata per migliorare l'accessibilità del web. Il suo obiettivo principale è aiutare gli sviluppatori a identificare e comprendere i problemi di accessibilità presenti nelle pagine web attraverso una rappresentazione visiva intuitiva. In questo modo, lo sviluppatore può percepire l'esperienza di navigazione in modo simile a un utente con disabilità visiva.

Per raggiungere questo scopo, Saharian adotta due strategie principali:

- **Visualizzazione:** innanzitutto, disattiva tutti i fogli di stile e gli stili in linea per rimuovere l'aspetto estetico della pagina così come progettato dallo sviluppatore. Successivamente, applica il proprio foglio di stile, appositamente studiato per offrire una visualizzazione chiara e segnalare eventuali problemi di accessibilità. Se un componente della pagina non è adeguatamente supportato, verrà evidenziato attraverso una rappresentazione visiva anomala.
- **Controllo degli input:** oltre alla modifica della visualizzazione, Saharian interviene anche sui metodi di interazione. In particolare, sostituisce gli input da mouse con input da tastiera per simulare l'esperienza di utenti che non possono utilizzare un mouse. Questo consente agli sviluppatori di verificare se la pagina è completamente navigabile e interagibile anche tramite tastiera.

3.1.1 Utilizzo degli standard ARIA da parte di Saharian

Saharian è uno strumento avanzato per l'accessibilità web che sfrutta in modo innovativo gli standard ARIA per migliorare l'esperienza utente sulle pagine web. Il suo funzionamento si basa su un processo sofisticato che si attiva all'avvio dell'estensione: innanzitutto, viene rimossa ogni forma di CSS preesistente dalla pagina in esame. Questo passaggio elimina tutte le personalizzazioni stilistiche che potrebbero alterare la struttura semantica e compromettere l'accessibilità del contenuto. Successivamente, Saharian applica un set di regole CSS specificamente progettato per mettere in evidenza la struttura accessibile della pagina.

Il fulcro dell'approccio di Saharian è un'analisi approfondita del Document Object Model (DOM) della pagina. Durante questo processo, lo strumento esegue una scansione completa degli elementi del DOM, individuando non solo quelli standard come input, pulsanti, immagini e link, ma anche componenti più complessi e personalizzati. Per questi ultimi, Saharian verifica attentamente la presenza di attributi ARIA, fondamentali per garantire l'accessibilità di contenuti non standard. Tra gli elementi chiave di questa verifica vi è l'attributo `role`, che Saharian utilizza per determinare come ciascun elemento debba essere interpretato dal punto di vista dell'accessibilità.

Oltre al `role`, Saharian analizza con particolare attenzione altri attributi ARIA, come `aria-label`, `aria-expanded` e `aria-selected`. Questi attributi sono essenziali per definire la semantica e lo stato degli elementi della pagina. Ad esempio, `aria-expanded` e `aria-selected` indicano lo stato dinamico di un componente, come un menu a discesa o un elemento selezionato. Saharian utilizza queste informazioni per applicare un CSS specifico che rende questi stati visivamente distinguibili e più facilmente comprensibili.

Grazie a questo approccio, Saharian non solo garantisce la conformità della pagina agli standard di accessibilità, ma ne trasforma anche la visualizzazione affinché gli sviluppatori possano comprendere in modo intuitivo l'impatto delle modifiche apportate. Questo metodo basato sulla visualizzazione aiuta a ridurre il divario tra l'intento progettuale e l'effettiva esperienza dell'utente.

Oltre a migliorare la resa visiva delle pagine web per renderle più accessibili, Saharian sfrutta in modo efficace gli attributi ARIA per emulare l'interazione da tastiera e il dinamismo della pagina. Un esempio significativo è la funzione `convertMouseEvent`, che garantisce l'accessibilità degli elementi interattivi tramite tastiera. Questa funzione analizza gli elementi che la richiamano, verificando che siano interattivi, come `radio`, `tab` o `option`, e controlla che il `tabindex` sia impostato correttamente. L'azione viene consentita solo se questi criteri sono soddisfatti, assicurando che gli utenti possano navigare e interagire con la pagina esclusivamente tramite tastiera, un requisito essenziale 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. Grazie a questo approccio integrato, lo strumento assicura che l'interfaccia utente di una pagina web sia non solo visivamente accessibile, ma anche completamente funzionale per chi naviga esclusivamente con la tastiera.

Inoltre, Saharian implementa controlli avanzati sugli attributi `aria-live` e `aria-atomic`, essenziali per gestire i contenuti dinamici delle pagine web. Lo strumento monitora costantemente la pagina e, quando rileva modifiche in sezioni che utilizzano `aria-live`, invia notifiche agli utenti per segnalare i cambiamenti. Questa funzionalità è particolarmente utile per le persone con disabilità visive, poiché permette loro di essere immediatamente informate su aggiornamenti o modifiche rilevanti nel contenuto, migliorando l'esperienza d'uso e garantendo che nessuna informazione venga persa.

3.1.2 Installazione

L'installazione di Saharian varia leggermente a seconda del browser, ma segue sempre lo stesso procedimento: il caricamento dell'archivio compresso come estensione per sviluppatori.

Su Google Chrome e i browser basati su Chromium, è necessario accedere alla pagina delle estensioni digitando `chrome://extensions` nella barra degli indirizzi. Da qui, bisogna attivare la modalità sviluppatore e caricare l'archivio premendo su "Load Unpacked".

Su Mozilla Firefox, invece, occorre navigare alla pagina `about:debugging`, selezionare "Questo Firefox" e caricare la cartella compressa cliccando su "Load Temporary Add-on".

3.1.3 Utilizzo

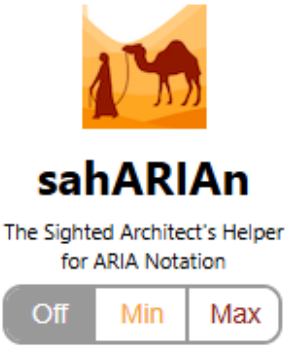

Figura 3.1: Schermata default di Saharian

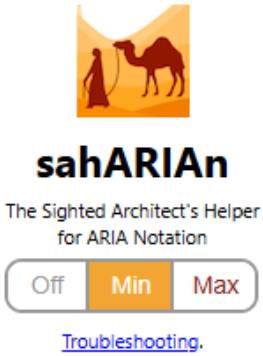

Figura 3.2: Schermata con il livello min attivo

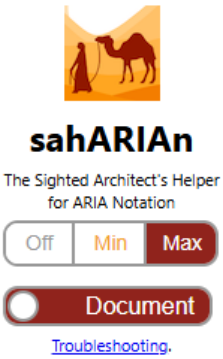

Figura 3.3: Schermata con il livello max attivo

Facendo clic sull'icona dell'estensione, si aprirà l'interfaccia di Saharian, progettata per essere intuitiva e facilmente navigabile. La schermata principale presenta tre pulsanti, ognuno dei quali attiva un diverso livello di funzionamento.

Il pulsante *Off* è impostato come predefinito e mantiene Saharian disattivato. Se si seleziona il pulsante *Min*, viene avviato il livello di accessibilità di base: Saharian rimuove i fogli di stile predefiniti della pagina e carica i propri, ottimizzati per evidenziare le problematiche di accessibilità. Il caricamento richiede solo pochi secondi, dopodiché la pagina modificata diventa navigabile per lo sviluppatore.

Il livello di accessibilità *Max* segue lo stesso principio, ma aggiunge un intervento più mirato su specifici elementi della pagina. In questa modalità è inoltre disponibile un toggle per scegliere tra le modalità *Document* e *Application*, che corrispondono alle modalità di navigazione utilizzate dagli screen reader:

- **Modalità Document:** consente di esplorare la pagina in modo sequenziale, dall'inizio alla fine, come avverrebbe in una lettura lineare.
- **Modalità Application:** è pensata per la navigazione di componenti interattivi, come la compilazione di un form. In questa modalità, gli elementi non interagibili vengono nascosti per simulare con maggiore precisione l'esperienza di un utente che utilizza uno screen reader.

Opzione Troubleshooting

Figura 3.4: Schermata di troubleshooting

Saharian include anche un'opzione *Troubleshooting*, che fornisce un elenco di problemi di accessibilità comuni e di elementi web diffusi, con relative spiegazioni su come implementarli correttamente. Questa funzione offre agli sviluppatori un riferimento rapido e pratico.

Gli argomenti trattati nella guida sono i seguenti:

- **Preferire elementi nativi rispetto a componenti personalizzati:** gli attributi ARIA dovrebbero essere utilizzati solo per colmare le lacune lasciate dai tag HTML standard. Se una funzionalità di accessibilità è già integrata in un elemento nativo, è sempre preferibile utilizzarlo per garantire maggiore compatibilità ed evitare problemi di sviluppo.
- **Landmarks:** rappresentano punti di riferimento della pagina e consentono agli screen reader di navigare rapidamente tra le diverse sezioni.
- **Gerarchia:** è fondamentale che la struttura della pagina sia chiara e coerente. Le intestazioni `H1` devono essere seguite da `H2`, e così via, mantenendo un ordine logico.
- **Immagini:** possono essere decorative o avere un significato informativo. Le immagini decorative devono essere segnalate con il ruolo `presentation` o `none`, oppure con un `alt` vuoto. Se l'immagine trasmette informazioni, è indispensabile fornire una descrizione nell'attributo `alt`.

- **Interazione con il mouse:** gli utenti non vedenti non utilizzano il mouse. Pertanto, gli elementi personalizzati devono supportare la navigazione da tastiera. Gli elementi HTML nativi, invece, gestiscono già questa funzionalità.
- **Campi di input testuali:** devono sempre avere un'etichetta associata, utilizzando l'attributo `for` nel tag `label`, oppure gli attributi `aria-label` o `aria-labelledby`.
- **Gruppi di radiobutton:** è importante raggrupparli utilizzando il tag `fieldset` e fornire una descrizione adeguata tramite il tag `legend`. Inoltre, bisogna assicurarsi che siano navigabili da tastiera.
- **Messaggi di errore:** per indicare che un campo di un form contiene un errore, è necessario impostare l'attributo `aria-invalid` su `true` e fornire un messaggio di errore con l'attributo `aria-errormessage`.
- **Listbox:** se si implementa una `listbox` personalizzata, è essenziale specificare gli attributi ARIA appropriati e indicare l'opzione selezionata con `aria-selected`.
- **Dialog:** i dialoghi devono essere chiudibili tramite tastiera e il focus deve rimanere confinato all'interno della finestra di dialogo. L'elemento deve essere identificato con `role="dialog"`.
- **Tabs:** per garantire l'accessibilità, bisogna utilizzare correttamente i ruoli `role="tablist"`, `role="tab"` e `role="tabpanel"`. È inoltre importante consentire la navigazione da tastiera e indicare la tab attiva con `aria-selected`.

3.2 Saharian applicato a siti reali

- ANSA

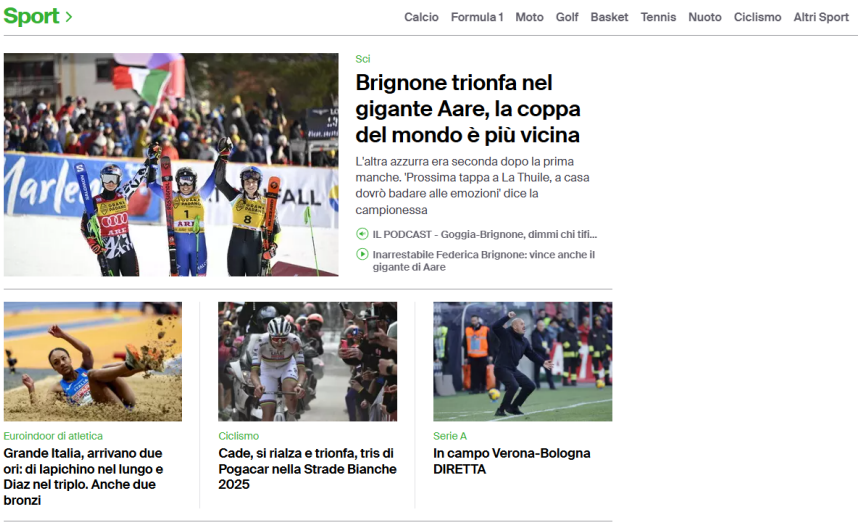

Figura 3.5: Pagina di ANSA con Saharian disattivato

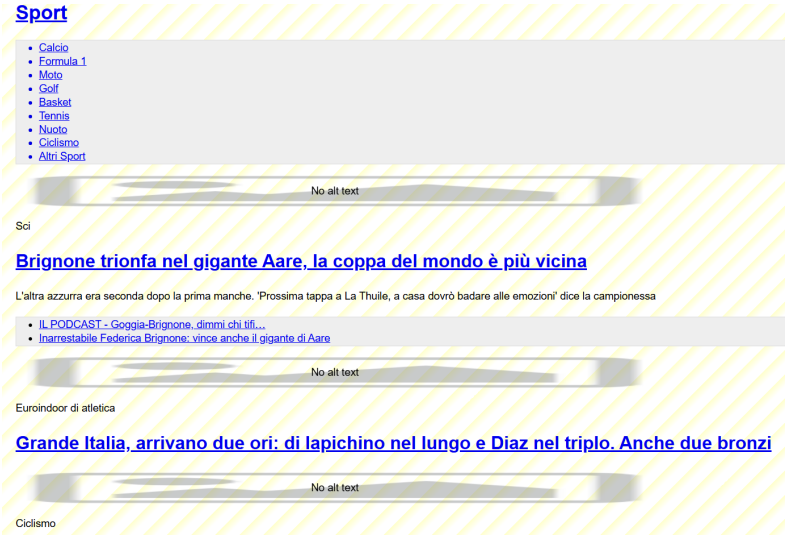

Figura 3.6: La stessa pagina di ANSA che mostra immagini senza testo alternativo

Analizzando il sito di notizie ANSA con Saharian attivo, si può notare un livello di accessibilità sufficiente. Tutti i contenuti rimangono navigabili correttamente e pulsanti e link funzionano come previsto. Il sito, per sua natura,

contiene molte immagini; la maggior parte di esse presenta un testo alternativo, ma alcune immagini nella pagina ne sono prive. Inoltre, la navigazione tra le varie sezioni risulta, grazie a Saharian, leggermente scomoda. Infatti, il sito è composto da varie sezioni, ognuna delle quali raggruppa un tipo di contenuto diverso. Un problema è che non sono presenti modalità per muoversi rapidamente da una sezione all'altra, costringendo l'utente a scorrere a lungo. L'analisi di questo sito con Saharian evidenzia anche un problema legato allo strumento, ovvero il tempo di caricamento. La grande quantità di contenuti e immagini rende il rendering della pagina un po' macchinoso. Saharian potrebbe essere migliorato sotto questo aspetto, ottimizzando i processi di scansione della pagina e la generazione di nuove immagini. Un'alternativa valida potrebbe essere applicare gli effetti di Saharian solo a una porzione della pagina scelta dallo sviluppatore.

- MDN

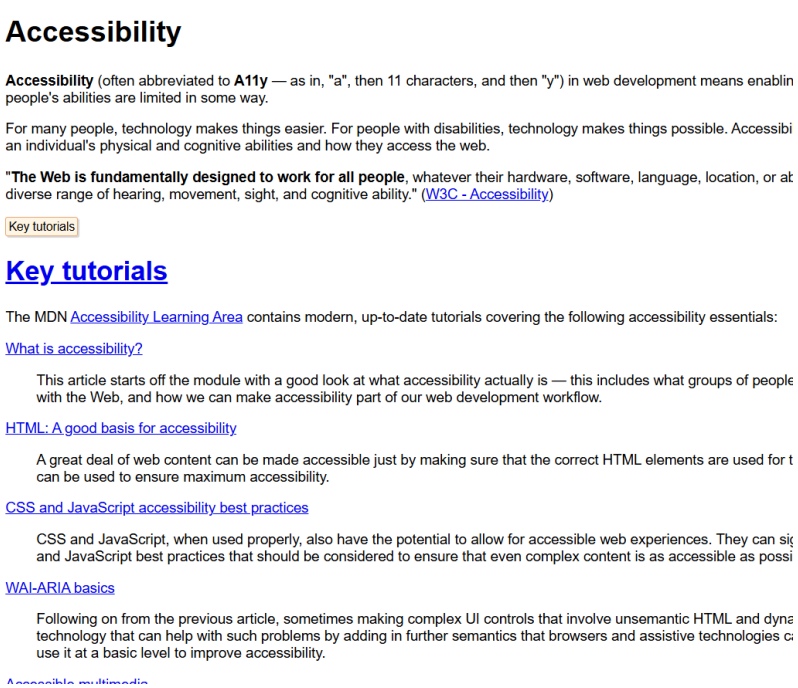

Figura 3.7: Pagina di MDN

Attivando Saharian sul sito Mozilla Developer Network, si può osservare che esso rispetta standard elevati di accessibilità. Il sito è facilmente navigabile, tutti gli elementi della pagina hanno uno scopo chiaro e gli elementi interattivi, come link e pulsanti, funzionano correttamente. In cima alla pagina sono presenti degli "skip links" [W3C25a], una tecnica che semplifica la navigazione,

permettendo agli utenti di saltare direttamente al contenuto principale della pagina, evitando di dover premere ripetutamente il tasto TAB per raggiungere la sezione desiderata. Inoltre, le intestazioni sono correttamente impostate e la gerarchia della pagina è rispettata. Attivando la modalità *Max*, si nota che ogni sezione della pagina ha il suo landmark associato e la semantica dei landmark è utilizzata correttamente.

Figura 3.8: Autocomplete nel campo di ricerca di MDN

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

Saharian riesce con successo a evidenziare come l'accessibilità della pagina sia di alto livello; inoltre, arrivare a queste conclusioni è stato molto rapido, mostrando così l'efficacia di Saharian come strumento di valutazione delle pagine web.

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

Per accompagnare l'utilizzo di Saharian, sono state progettate pagine di esempio che offrono agli sviluppatori un ambiente di test per osservare il comportamento dell'estensione con diverse componenti di una pagina web.

Questi esempi includono sia casi di utilizzo corretto che errori comuni. In particolare, ogni scenario presenta:

- un esempio di implementazione accessibile utilizzando solo HTML nativo;
- una variante personalizzata che sfrutta gli attributi ARIA per migliorare l'accessibilità;
- un caso con un'implementazione errata, priva dei corretti attributi ARIA.

Gli sviluppatori possono esplorare queste pagine per comprendere l'uso corretto di tag e attributi HTML, verificare l'impatto degli attributi ARIA e osservare come Saharian interagisce con ciascun caso. Questi esempi possono anche servire come riferimento pratico per integrare componenti accessibili nelle proprie pagine web.

Di seguito, una selezione di esempi utili per guidare gli sviluppatori nella comprensione dell'estensione:

- **Alert**

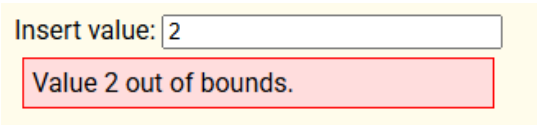

Figura 3.9: Alert corretto con Saharian disattivato

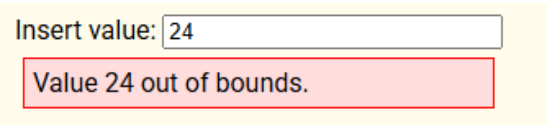

Figura 3.10: Alert errato con Saharian disattivato

Nell'immagine sono mostrati due campi di input testuali identici, entrambi accompagnati da un messaggio di errore. Tuttavia, uno è implementato correttamente, mentre l'altro presenta un utilizzo errato degli attributi ARIA. Visivamente, i due elementi appaiono uguali, ma la differenza diventa evidente quando si analizza il loro comportamento con le tecnologie assistive.

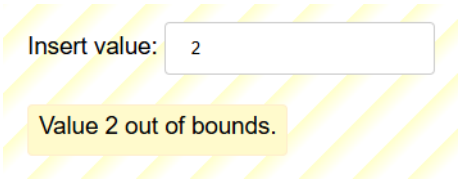

Figura 3.11: Alert corretto con Saharian attivato

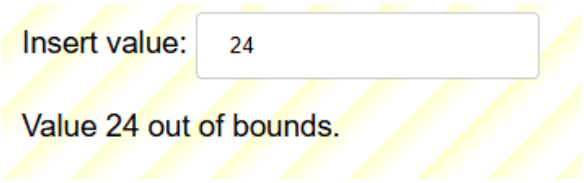

Figura 3.12: Alert errato con Saharian attivato

Gli alert hanno il compito di notificare un errore all'utente, evidenziando l'urgenza e l'importanza del messaggio. Nella versione corretta, Saharian mantiene un bordo e uno sfondo distintivo per l'alert, garantendo una chiara comunicazione visiva e semantica. Al contrario, nella versione errata, il messaggio di errore è rappresentato come semplice testo, facendo capire quanto sia difficile per le tecnologie assistive identificarlo come un avviso importante. Questo esempio evidenzia l'importanza di un corretto utilizzo degli attributi ARIA per migliorare l'accessibilità.

- **Checkbox**

  Questo esempio evidenzia l'importanza della navigazione da tastiera per l'accessibilità e il ruolo fondamentale degli attributi ARIA.

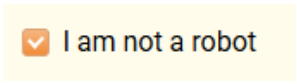

Figura 3.13: Checkbox corretto con Saharian disattivato

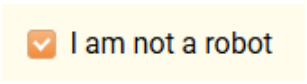

Figura 3.14: Checkbox errato con Saharian disattivato

Nella prima parte della pagina troviamo due checkbox implementate correttamente: la prima utilizza gli elementi nativi HTML, mentre la seconda sfrutta un uso adeguato di attributi ARIA e JavaScript. È importante notare come replicare in JavaScript le funzionalità native richieda un notevole sforzo da parte dello sviluppatore.

A seguire, sono presenti quattro esempi di implementazioni errate, ognuna con un diverso problema di accessibilità:

  - **Nessun supporto da tastiera:** la checkbox non è interagibile tramite tastiera, non può ricevere il focus né essere attivata o disattivata con i tasti *Enter* o *Spacebar*. Questo rappresenta un grave ostacolo per gli utenti con disabilità visive.

- **Attributo ARIA di stato errato:** lo stato della checkbox è segnalato con un attributo errato, causando possibili errori nella lettura da parte degli screen reader e generando confusione nell'utente.
- **Mancato report dello stato:** la checkbox non dispone di alcun attributo che comunichi il suo stato, rendendo impossibile per le tecnologie assistive informare l'utente sulla sua selezione.
- **Mancato utilizzo degli attributi ARIA:** la checkbox è priva di qualsiasi indicazione di accessibilità, risultando per gli screen reader un semplice elemento testuale privo di funzionalità.

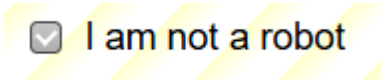

Figura 3.15: Checkbox corretto con Saharian attivato

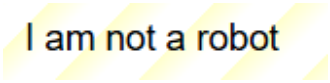

Figura 3.16: Checkbox errato con Saharian attivato

Attivando Saharian, questi problemi vengono messi in evidenza in modo chiaro. Le prime due checkbox continuano a funzionare correttamente, mentre gli ultimi quattro esempi mostrano evidenti anomalie: tre di essi sono cliccabili ma il loro stato non viene aggiornato, mentre l'ultimo non risponde neanche al click. Questo dimostra come l'uso corretto degli attributi ARIA sia indispensabile per garantire un'interazione accessibile e come non sia sufficiente implementare una componente grafica esclusivamente tramite CSS.

- **Immagini**

  Le immagini sono un elemento fondamentale del web e sono presenti in quasi tutte le pagine. Tuttavia, se non vengono implementate correttamente, gli screen reader non possono trasmetterne il significato agli utenti non vedenti. Saharian aiuta gli sviluppatori a distinguere le diverse tipologie di immagini, evidenziando quali sono accessibili e quali no.

  Il primo esempio mostra un'immagine implementata correttamente con HTML nativo, sottolineando l'importanza dell'attributo `alt`, che fornisce un testo alternativo per descrivere il contenuto dell'immagine. Nel secondo esempio troviamo tre immagini decorative, che hanno un puro scopo estetico e non trasmettono informazioni. Per essere considerate accessibili, devono avere un `alt` vuoto oppure gli attributi `role="presentation"` o `role="none"`.

  Il terzo esempio mostra un'immagine accessibile implementata con un corretto utilizzo degli attributi ARIA.

Passando agli esempi errati, il primo riguarda un'immagine priva di testo alternativo. Questo rappresenta un grave problema, poiché l'immagine può trasmettere un significato agli utenti vedenti, ma tale informazione viene completamente persa per chi utilizza uno screen reader. L'ultimo esempio errato è un'immagine realizzata tramite un `div` senza alcun attributo ARIA, rendendola completamente inaccessibile.

Attivando Saharian, possiamo osservare che le immagini implementate correttamente vengono sostituite da un'immagine fittizia con il testo alternativo sovrapposto, evidenziando che questo è l'unico contenuto percepibile da un utente non vedente. Le immagini decorative, invece, scompaiono, poiché non vengono lette dagli screen reader.

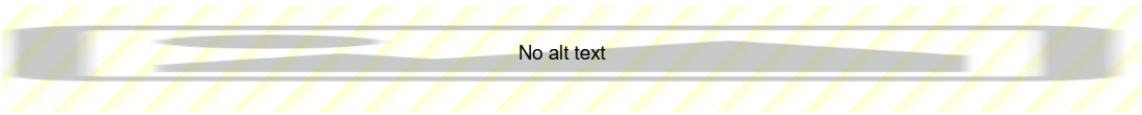

Figura 3.17: Immagine senza testo alternativo con Saharian attivato

L'immagine senza testo alternativo viene accompagnata da un messaggio di errore, segnalando chiaramente la sua mancata accessibilità. L'immagine priva di ruoli ARIA non viene affatto visualizzata, proprio perché non viene riconosciuta come tale.

La seconda parte della pagina è dedicata alle immagini interattive, molto diffuse sul web. Sono presenti due esempi, entrambi basati su un elemento `img` corretto con un testo alternativo. Ciò che li differenzia è l'implementazione dell'interattività. La prima immagine ha un attributo `role="button"` e un `tabindex`, che la rendono identificabile come elemento interattivo e ne consentono l'uso tramite tastiera. La seconda immagine, invece, è priva di questi attributi, risultando non accessibile per chi utilizza tecnologie assistive.

Attivando Saharian, la prima immagine mantiene la sua interattività, mentre la seconda diventa non cliccabile, dimostrando l'importanza di questi attributi per garantire l'accessibilità.

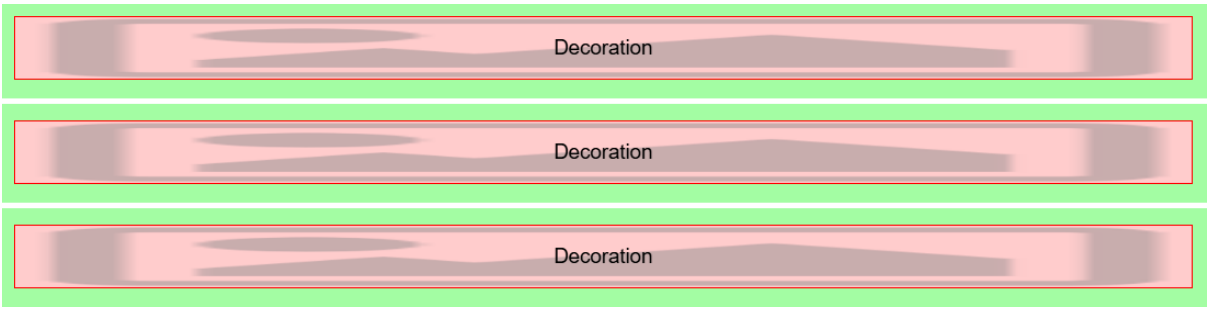

Figura 3.18: Immagine decorativa con livello max attivato

Questo esempio permette inoltre di introdurre una differenza tra le due modalità di Saharian: *Min* e *Max*. Nella modalità avanzata, le immagini decorative vengono evidenziate con un bordo verde e accompagnate dal testo "Decoration". Questo perché, in questa modalità, Saharian fornisce informazioni più dettagliate, indicando esplicitamente allo sviluppatore quali immagini sono presenti sulla pagina e specificando che il loro scopo è puramente estetico.

- **Live Region**

  Le *live region* sono una funzionalità essenziale per le pagine web dinamiche, in cui i contenuti possono aggiornarsi automaticamente. Senza un'adeguata gestione, gli utenti che utilizzano uno screen reader potrebbero non accorgersi delle modifiche o, al contrario, subire interruzioni improvvise che compromettono la navigazione.

  Esistono due principali categorie di *live region*: quelle **assertive**, che interrompono immediatamente la navigazione per segnalare un aggiornamento, e quelle **polite**, che invece aspettano un momento di inattività per notificare il cambiamento. Un'altra distinzione riguarda l'attributo `aria-atomic`, che se impostato a `true` fa leggere solo la parte modificata, mentre con `false` costringe lo screen reader a rileggere l'intero contenuto.

  Nella pagina sono presenti quattro esempi, ciascuno con una configurazione diversa. Per un utente vedente non ci sono differenze visibili, ma attivando Saharian è possibile osservare come le live region **assertive** catturino immediatamente il focus, mentre in quelle **atomic** viene evidenziata solo la parte effettivamente modificata.

  La modalità *Max* di Saharian enfatizza ulteriormente questi comportamenti: le live region assertive vengono isolate dal resto del contenuto e ricevono il focus in modo forzato, simulando l'esperienza di un utente che viene bruscamente interrotto da un aggiornamento. In modalità *Min*, invece, il loro

funzionamento viene semplicemente evidenziato senza alterare l'esperienza di navigazione.

- **Slider**

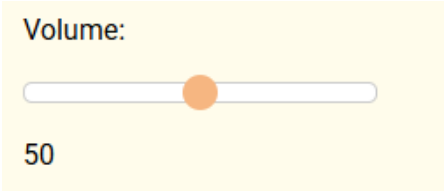

Figura 3.19: Slider corretto con Saharian disattivato

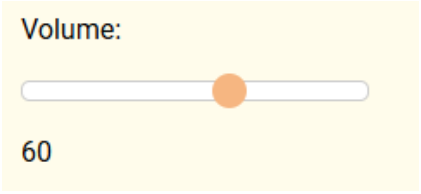

Figura 3.20: Slider errato con Saharian disattivato

Lo slider è un elemento interattivo che permette all'utente di modificare un valore tramite il trascinamento del mouse. Nell'esempio, troviamo uno slider implementato con HTML nativo e due slider creati tramite tag HTML alternativi modificati con i fogli di stile. La differenza principale è che uno dei secondi utilizza correttamente gli attributi ARIA, come `role="slider"` e `aria-valuemin`, `aria-valuemax`, `aria-valuenow`, che permettono allo screen reader di interpretare il valore dello slider. L'altro, privo di questi attributi, risulta inaccessibile.

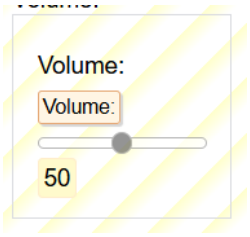

Figura 3.21: Slider corretto con Saharian attivato

Figura 3.22: Slider errato con Saharian attivato

Attivando Saharian, possiamo notare che i due slider correttamente implementati conservano la loro funzionalità, mentre il terzo, errato, viene visualizzato come un numero, senza interazione possibile. Questo accade perché l'elemento grafico, realizzato solo tramite foglio di stile, viene rimosso con l'attivazione di Saharian. L'elemento con ARIA viene sostituito da uno slider HTML per preservare le funzionalità e simulare come esso venga effettivamente riconosciuto dagli screen reader.

In una pagina dedicata agli slider verticali, Saharian è in grado di renderizzarli correttamente, anche con orientamenti diversi, dimostrando la flessibilità dell'estensione nell'affrontare varie tecniche di implementazione.

- **Video**

  I video devono essere accessibili e includere sottotitoli per utenti non udenti. Nell'esempio, troviamo un video corretto, che include l'opzione *CC* per accedere ai sottotitoli, e uno errato, privo di questa opzione. Per aggiungere i sottotitoli, è necessario utilizzare il tag `track` e impostare come sorgente un file `vtt`. Attivando Saharian, il secondo video viene rimosso poiché non è accessibile. Il primo video, invece, rimane visibile.

  In modalità *Max*, il video non supportato viene ancora mostrato, ma con una nota che segnala la mancanza dei sottotitoli, aiutando lo sviluppatore a comprendere l'errore e la sua causa.

- **Landmarks**

  L'esempio dedicato ai landmarks mostra agli sviluppatori come organizzare la pagina per permettere a un utente che utilizza una tecnologia assistiva di navigare facilmente. I landmarks sono punti di riferimento che lo screen reader propone all'utente, consentendogli di saltare rapidamente da una sezione all'altra. In particolare, i landmarks sono i tag HTML come `main`, `footer`, `header`, `section` e `nav`, che rappresentano sezioni specifiche della pagina. Ognuno di questi tag ha un corrispondente ruolo ARIA, ma è importante sottolineare che non è mai consigliato usare il ruolo, se non strettamente necessario. È sempre preferibile utilizzare i tag HTML nativi, quando possibile.

  Nell'esempio sono presenti tutti gli elementi sopra citati, che insieme compongono il layout di una tipica pagina web. Attivando Saharian, si nota che la pagina non subisce cambiamenti evidenti, poiché la modalità `Min` non è progettata per evidenziare il layout e i landmarks. La differenza si osserva invece attivando la modalità `Max`. Ogni landmark avrà un'etichetta che ne indica il tipo, accompagnata da un bordo che evidenzia la zona della pagina occupata. In questo modo, lo sviluppatore può visualizzare come la pagina è suddivisa, dove l'utente verrà portato durante la navigazione tra i landmarks e, soprattutto, verificare se il suo approccio di implementazione viene correttamente riconosciuto dagli screen reader.

Capitolo 4

Architettura di Saharian

4.1 Architettura e implementazione 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 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 JavaScript per analizzare e modificare le pagine web. Con esso 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 siano implementati in maniera acessibile, 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.

Quando Saharian viene installato come estensione del browser, inietta uno script JavaScript. Inizialmente, lo script è disabilitato, ma quando l'utente modifica il livello di attivazione nella schermata dell'estensione, Saharian inizia a eseguire le sue funzioni ogni 500 ms, assicurandosi che ogni modifica al DOM venga rilevata e gestita in tempo reale. Questo ciclo continuo permette all'estensione di rimanere sincronizzata con ogni cambiamento della pagina e di intervenire prontamente con le opportune correzioni. Se viene attivato il livello *Min*, si attivano le funzioni `generateAccessibleNames`, `deactivateImages`, `deactivateStyles`, `deactivateStylesheets` e `generateAriaLiveControls`. Queste costituiscono il cuore di Saharian e implementano le sue funzionalità di base.

- **generateAccessibleName**

  La funzione `generateAccessibleNames()` assegna nomi e descrizioni accessibili [DGC18] agli elementi di una pagina web. Identifica quelli con attributi `aria-label` o `aria-labelledby` e sostituisce gli slider personalizzati con versioni native per migliorarne l'accessibilità. Genera testi alternativi analizzando vari attributi (`aria-labelledby`, `aria-label`, `title`, `alt`) e, se necessario, li inserisce in elementi aggiuntivi per non alterare la struttura. Infine, gestisce `aria-describedby` per le descrizioni accessibili, evitando ricorsioni infinite nel processo.

- **deactivateImages**

La funzione `deactivateImages()` e il metodo di supporto `sideline()` gestiscono la visualizzazione delle immagini in Saharian. `deactivateImages()` sostituisce visivamente le immagini con il loro testo alternativo, se presente, o con un'indicazione predefinita. Identifica gli elementi immagine, verifica se sono già stati elaborati e genera dinamicamente un elemento testuale che, dopo un breve intervallo, viene convertito in un'immagine. `sideline()` memorizza gli attributi originali per permettere il ripristino. Questo approccio garantisce una gestione dinamica senza alterare definitivamente la struttura della pagina.

- **deactivateStyles**

  La funzione `deactivateStyles()` analizza il DOM per individuare elementi con stili inline e modificarli secondo criteri predefiniti, valutando il loro impatto sull'accessibilità. `generateReplacementForRule()` elabora nuove regole CSS per proprietà chiave come `position`, `display` e `visibility`, adattandole senza alterare la struttura semantica della pagina. Le funzioni `sideline()` e `sidelined()` archiviano temporaneamente gli attributi originali e verificano se un elemento è già stato modificato, garantendo il ripristino e prevenendo modifiche ridondanti. Questo approccio assicura un'analisi dinamica senza alterazioni permanenti.

- **deactivateStylesheets**

  La funzione `deactivateStylesheets()` disabilita i fogli di stile non essenziali per l'accessibilità in Saharian. Analizza i fogli di stile del documento, recupera il loro contenuto e lo modifica con `replacedRules()`, che trasforma le regole CSS per migliorarne l'accessibilità. Questa funzione gestisce `@import`, `@media` e le regole di stile, adattandole senza alterare la logica originale. Infine, i fogli di stile originali vengono disattivati e le nuove regole applicate, garantendo una presentazione accessibile del contenuto.

- **generateAriaLiveControls**

  La funzione `generateAriaLiveControls` in Saharian aiuta gli sviluppatori a comprendere come i lettori di schermo annunciano le modifiche dinamiche ai contenuti. Utilizza gli attributi ARIA `aria-live` e `aria-atomic`, applicando stili visivi distinti per notifiche `assertive` e `polite`.

  La funzione monitora le modifiche tramite `MutationObserver`, evidenziando i cambiamenti in base a `aria-live`. Le notifiche **assertive** ricevono maggiore enfasi con il focus automatico, mentre quelle **polite** vengono segnalate senza alterare la navigazione. Inoltre, `aria-atomic` influisce sulla gestione delle modifiche: con `false`, solo le parti modificate vengono evidenziate; con `true`, l'intero contenuto può essere enfatizzato con scrolling e focus automatico.

Successivamente, Saharian procede a modificare il CSS in base al livello di accessibilità scelto, che può essere minimo o massimo. La personalizzazione avviene applicando regole CSS specifiche, pensate per agire solo sugli elementi HTML fondamentali o su quelli che presentano un attributo ARIA. Se un elemento non dispone di un attributo ARIA, la sua visualizzazione potrebbe risultare alterata o addirittura venire omessa. Questo metodo evidenzia l'importanza dell'uso corretto degli attributi ARIA per garantire un contenuto web accessibile.

Quando invece viene attivata la modalità *Max* avviene lo stesso identico procedimento, con l'aggiunta dell'attivazione della stylesheet `SaharianLevelMax` che rappresenta le regole CSS per il livello **Max**. Alcune funzioni modificano il loro comportamento in base al livello indicato. Inolte, quando il livello di accessibilità selezionato è *Max*, compare uno switch che consente di passare tra le modalità *Document* e *Application*. Nella modalità *Document*, la pagina viene visualizzata in modo completo e in linea con la navigazione standard. Invece, in modalità *Application*, vengono identificati e resi visibili solo gli elementi interattivi, cioè quelli che possono ricevere il focus, mentre tutti gli altri vengono nascosti. Questa funzionalità simula il comportamento di uno screen reader, offrendo così un'esperienza d'uso più precisa per testare l'accessibilità.

Infine Saharian ha un meccanismo di disattivazione, quindi ogni funzione che viene usata nel processo di attivazione ha una corrispettiva funzione utilizzata per invertire le modifiche apportate, ripristinando la pagina com'era in precedenza. Questa disattivazione avviene quando l'utente preme sul livello *Off*, ma anche ogni volta che Saharian cambia livello tra *Min* e *Max*, viene sempre eseguita una disattivazione preventiva. Questo assicura un'esecuzione pulita e previene potenziali bug, evitando che permangano modifiche indesiderate tra un livello e l'altro.

4.2 Algoritmi rilevanti

Di seguito spiegherò le modifiche che ho apportato al codice preesistente, precedentemente descritto. In particolare, ho aggiunto piccole funzionalità ad alcuni metodi di Saharian e ho creato una serie di metodi per gestire specifici ruoli ARIA.

4.2.1 Miglioramento dei metodi precedenti

- **generateAccessibleName**

  Una modifica importante che ho apportato a questo metodo consiste nel considerare le etichette `label` con attributo `for` come potenziali accessible name. L'algoritmo ricerca innanzitutto tutti gli elementi associati a una `label`.

Successivamente, quando viene valutato l'accessible name di questi elementi, se non sono presenti gli attributi `aria-label`, `aria-labelledby` o `aria-describedby`, viene utilizzato il contenuto della label.

Nel codice della funzione `generateAccessibleName` ho inoltre aggiunto una gestione specifica per le barre di progresso, sia quelle native (`progress`) sia quelle personalizzate con `role="progressbar"`. L'obiettivo è rendere la percezione delle progress bar più simile alla rappresentazione offerta da uno screen reader. Per questo motivo, vengono sostituite da un valore testuale, che può corrispondere alla percentuale di avanzamento oppure a un messaggio indicante un caricamento in corso.

Il metodo scansiona tutti gli elementi della pagina appartenenti a queste due categorie e, per ciascuno di essi, determina il valore attuale. Se l'elemento è un `progress` nativo, il valore viene recuperato dall'attributo `value`; altrimenti, viene estratto dall'attributo `aria-valuenow`. Se il valore non è definito o è nullo, il codice cerca un'etichetta associata tramite `aria-labelledby` o `aria-label` e la utilizza come testo descrittivo. In assenza di un'etichetta, viene assegnato un valore predefinito di "Loading...".

Se il valore è presente, il codice calcola la percentuale di avanzamento sulla base del valore minimo (`aria-valuemin` o 0 di default) e massimo (`aria-valuemax` o 100 di default). Infine, imposta l'attributo `data-display-text` con la percentuale calcolata o con il valore testuale definito in `aria-valuetext`, fornendo così un'indicazione chiara dello stato di avanzamento. Questo attributo verrà poi utilizzato dal CSS per nascondere la grafica della barra di progresso e mostrare invece un testo con il valore corrispondente.

- **deactivateImages**

  La funzione `deactivateImages` si occupa di individuare il testo alternativo di un'immagine prima di nasconderla. La modifica che ho apportato valuta se l'immagine in questione è decorativa: in tal caso, il testo alternativo viene impostato sulla stringa "Decoration". Questo comportamento è utile quando il livello *Max* è attivo, poiché le immagini decorative verranno visualizzate con questo testo alternativo.

- **generateAriaLiveControls**

  Anche nella funzione `generateAriaLiveControls` ho apportato una modifica attivabile esclusivamente quando Saharian è impostato sul livello *Max*. Quando una live region di tipo `assertive` subisce una modifica, l'utente che utilizza tecnologie assistive viene distolto dalla navigazione in modo brusco. Per simu-

lare questa esperienza utente, Saharian sposta la live region modificata lontano dal contenuto della pagina e le assegna il focus.

Per implementare questa funzionalità, viene applicata la classe CSS `.offscreen`, che imposta `position: absolute` sulla live region.

4.2.2 Metodi nuovi

Di seguito sono elencati i metodi che ho aggiunto per permettere a Saharian di supportare un maggior numero di attributi ARIA.

- **Document/Application mode**

  Questa parte del codice di Saharian serve a evidenziare visivamente gli elementi della pagina che possono ricevere il focus, simulando il comportamento di uno screen reader in due modalità: *application* e *document*.

  Quando la modalità *application* è attiva, il codice esamina tutti gli elementi della pagina e controlla se sono interagibili. Gli elementi non focalizzabili vengono contrassegnati con la classe `.saharian-not-focussable`, che li rende nascosti, mentre quelli focalizzabili ricevono la classe `.saharian-focussable`, che li mantiene visibili. Questo aiuta a mostrare agli sviluppatori quali elementi sono effettivamente navigabili tramite tastiera o screen reader.

  Se invece la modalità *document* è attivata, le classi vengono rimosse, ripristinando la visibilità originale di tutti gli elementi. In questo modo, Saharian permette agli sviluppatori di verificare rapidamente come un utente che usa solo la tastiera o uno screen reader percepirebbe la struttura della pagina.

- **checkVideoAccessibility**

  Questa funzione di Saharian serve a controllare che i video presenti sulla pagina abbiano sottotitoli o descrizioni accessibili. Quando l'estensione è attiva, il codice analizza tutti gli elementi `video` e verifica se contengono almeno una traccia `track` con l'attributo `kind="captions"` o `kind="descriptions"`, che sono fondamentali per garantire l'accessibilità alle persone con disabilità uditive.

  Se un video non ha queste tracce, Saharian aggiunge un messaggio accanto al video per segnalare la mancanza di sottotitoli o descrizioni e applica una classe CSS per evidenziarlo. In questo modo, gli sviluppatori possono individuare facilmente i video non accessibili.

Metodi che implementano ruoli ARIA

- **replaceMeters**

  Questa funzionalità di Saharian si occupa di trasformare gli elementi meter nativi e personalizzati in modo da far comprendere allo sviluppatore come vengono letti da uno screen reader.

  Il metodo `replaceMeters()` identifica tutti i meter presenti nella pagina e li sostituisce con un elemento di testo che riporta il valore percentuale, eventuali descrizioni testuali presenti nell'attributo `aria-valuetext` e una valutazione dello stato rispetto agli intervalli definiti (low, high, optimum). Per ottenere queste informazione viene utilizzata la funzione di supporto `extractMeterInfo(meter)`, che analizza tutti questi campi per restituire una stringa, che rappresenta la descrizione testuale del meter e un valore booleano che è `true` nel caso in cui il meter abbia un valore ottimale. Infine, `updateMeters()` si assicura che i valori testuali visualizzati vengano aggiornati dinamicamente, adattandosi ai cambiamenti nei meter. Questa implementazione aiuta gli sviluppatori a comprendere più facilmente come i meter vengono interpretati dagli screen reader, facilitando l'ottimizzazione dell'accessibilità.

- **findScrollableContent**

  Questa funzionalità di Saharian evidenzia i contenitori associati a barre di scorrimento personalizzate, rendendo più chiara la loro relazione con gli elementi scrollbar definiti tramite `role="scrollbar"`.

  Il metodo `findScrollableContent()` cerca tutti gli elementi con `role="scrollbar"` e, se hanno l'attributo `aria-controls`, individua l'elemento corrispondente e gli assegna una classe. Questa funzionalità da uno stile uniforme a tutte le barre di scorrimento presenti nella pagina, in modo che lo sviluppatore può riconoscere di averle implementate correttamente.

Metodi che implementano attributi ARIA

- **highlightFormErrors**

  Questo metodo identifica gli elementi del DOM che presentano errori di validazione, spesso come elmenti di un form, riconoscibili dall'attributo `aria-invalid="true"` e da un riferimento a un messaggio di errore tramite l'attributo `aria-errormessage`. Se l'elemento in errore sta ricevendo il focus, il metodo evidenzia il messaggio di errore associato applicandogli uno stile di testo in rosso e in grassetto. Se l'elemento non sta ricevendo il focus, rimuove tale evidenziazione ripristinando lo stile predefinito.

- **handleFlowtoHighlight**

  Questo metodo gestisce l'evidenziazione degli elementi referenziati dall'attributo `aria-flowto`, che definisce una relazione di navigazione tra elementi nella pagina. Il funzionamento si basa sull'aggiunta o rimozione di eventi di interazione con il mouse. Quando un utente passa il cursore su un elemento che possiede l'attributo `aria-flowto`, il metodo recupera gli ID degli elementi destinazione e li evidenzia aggiungendo la classe CSS `saharian-flowto-highlight`. Quando il cursore si sposta altrove, l'evidenziazione viene rimossa ripristinando lo stile originale.

  L'obiettivo di questo metodo è migliorare la comprensione del funzionamento dell'attributo `aria-flowto`, fornendo un'indicazione del percorso suggerito da esso.

- **addAriaHasPopupTooltips**

  Questo metodo identifica tutti gli elementi che possiedono l'attributo `aria-haspopup`, il quale indica la presenza di un popup associato. Dopo aver verificato che il valore di `aria-haspopup` sia valido (accettando `menu`, `listbox`, `tree`, `grid` e `dialog`), il metodo genera un tooltip informativo contenente il tipo di popup associato.

  Il tooltip viene creato dinamicamente come un elemento `div`, a cui viene assegnata la classe `saharian-popup-tooltip` e un testo descrittivo. Una volta aggiunto al documento, il tooltip viene posizionato accanto all'elemento che possiede `aria-haspopup`, utilizzando le coordinate fornite da `getBoundingClientRect()`. Per garantire che il tooltip mantenga una posizione corretta anche in caso di ridimensionamento o scorrimento della pagina, vengono registrati eventi resize e scroll che aggiornano dinamicamente la posizione del tooltip.

  Questo metodo aiuta lo sviluppatore a identificare un elemento popup e a comprendere se esso è stato riconosciuto adeguatamente.

- **addAriaModalOverlay**

  Questo metodo crea un overlay semi-trasparente per i modali accessibili, ovvero che hanno l'attributo `aria-modal="true"`. L'overlay viene generato come un elemento `div`, a cui viene assegnata la classe `saharian-modal-overlay`, e inizialmente è nascosto. Infine, viene aggiunto al body del documento, ma rimane nascosto fino a quando un modale non viene rilevato come visibile.

  Questo metodo gestisce la visibilità dell'overlay in base alla presenza di modali accessibili, ovvero quelli che presentano gli attributi ARIA adeguati. Scansiona

tutti i modali della pagina e, se ne trova almeno uno visibile, mostra l'overlay. Se nessun modale è visibile, l'overlay viene nuovamente nascosto.

Per verificare se il modale è visibile, utilizza una funzione di supporto, che controlla che non abbia `display: none`, `visibility: hidden` o `opacity: 0`, e che le sue dimensioni siano maggiori di zero. Se tutte queste condizioni sono soddisfatte, l'elemento è considerato visibile.

L'obiettivo di questi metodi è far notare allo sviluppatore come il modale interagisce con il resto della pagina, ovvero nascondendola completamente e limitando tutte le interazione agli elementi all'interno del modale.

- **addMultilineSymbol**

  Questo metodo individua tutti gli elementi con `role="textbox"` e `aria-multiline="true"`, che rappresentano campi di testo multilinea, e aggiunge un simbolo visivo di ritorno a capo accanto ad essi per indicare esplicitamente la loro natura multilinea. Se non esiste già viene creato un `div` e viene posizionato alla fine del campo di testo. Questa funzionalità aiuta gli sviluppatori a comprendere meglio come un utente con disabilità visive potrebbe percepire il contenuto della pagina, evidenziando in modo chiaro quali campi di input supportano più righe di testo.

- **enableMultiselectableText**

  Il metodo `enableMultiselectableText()` ha lo scopo di rendere evidente agli sviluppatori quali elementi della pagina supportano la selezione multipla. Identifica gli elementi con `aria-multiselectable="true"` applicato a ruoli come `listbox`, `grid`, `tree`, `tablist` e agli elementi `select` con attributo `multiple`. Per ciascuno di essi, aggiunge un'indicazione testuale accanto all'elemento, informando che è possibile selezionare più elementi contemporaneamente. Questa funzionalità aiuta gli sviluppatori a visualizzare in modo immediato quali elementi supportano la selezione multipla, migliorando la comprensione dell'accessibilità della pagina.

- **generateTableHeaders**

  Questa funzione di Saharian si occupa di migliorare la comprensione delle tabelle HTML per gli sviluppatori, evidenziando le intestazioni associate a ciascuna cella di dati.

  Quando in una tabella esistono celle `td` con l'attributo `headers`, il codice estrae i riferimenti alle intestazioni corrispondenti, recupera i loro testi e li combina in un'unica stringa. Questa stringa viene poi memorizzata nell'attributo perso-

nalizzato `data-saharian-an`, che viene usata in `generateAccessibleNames` per visualizzare le intestazioni accanto alla cella stessa.

Capitolo 5

Valutazione

Dopo aver implementato nuove funzionalità in Saharian, è stato realizzato un test con l'obiettivo di valutare in modo approfondito l'efficienza e l'efficacia degli sviluppatori nella creazione di componenti web accessibili, sia con che senza l'ausilio dello strumento. Poiché Saharian è progettato per facilitare lo sviluppo di pagine web più accessibili, il test mira anche a comprendere in che misura l'estensione possa migliorare il flusso di lavoro degli sviluppatori, ridurre gli errori di accessibilità e aumentare la consapevolezza sulle best practice. Inoltre, viene analizzato il livello di soddisfazione nell'utilizzo dello strumento, per identificare eventuali margini di miglioramento e ottimizzazione. Il test è stato svolto nel mese Febbraio 2025 e ha coinvolto la partecipazione di sei studenti di informatica con poca esperienza nello sviluppo web accessibile. La sessione si è svolta interamente online, con i partecipanti suddivisi in due gruppi. L'assegnazione ai gruppi è stata organizzata in base alla disponibilità di ciascun partecipante, garantendo così la massima flessibilità e partecipazione.

5.1 Protocollo di test

Il test è di tipo sommativo e ha una durata prevista compresa tra 20 e 40 minuti. Per il test è stata creata una pagina web che ha automatizzato la raccolta dei risultati. A ogni partecipante è stato assegnato un gruppo, A o B, ed è stato richiesto di selezionarlo nell'interfaccia del sito. Successivamente, è stata presentata un'interfaccia in cui è stato possibile svolgere il test: a sinistra un campo di testo contenente codice HTML, CSS e JavaScript, a destra l'anteprima del codice.

Il test è stato suddiviso in due set di esercizi:

- **Primo set:** i partecipanti hanno analizzato tre componenti HTML con vari errori di accessibilità, che hanno dovuto identificare e valutare senza l'ausilio di Saharian.
- **Secondo set:** i partecipanti hanno affrontato problemi simili, ma questa volta utilizzando Saharian, per valutarne l'efficacia nell'individuazione e correzione degli errori di accessibilità.

Il Gruppo B ha svolto gli stessi esercizi, ma in ordine inverso. Questo approccio comparativo ha consentito di misurare l'impatto dell'utilizzo di Saharian nell'identificazione e correzione degli errori e di valutare la capacità dei partecipanti di riconoscere questi problemi autonomamente.

Nella pratica il test è stato svolto con sei individui organizzati in due gruppi da tre. Il gruppo A e il gruppo B hanno svolto il test in due giorni diversi. Prima il gruppo A e poi il gruppo B.

L'obiettivo del test è stato duplice:

1. **Valutare l'efficacia di Saharian** come strumento di supporto per gli sviluppatori nella creazione di siti web accessibili.
2. **Sensibilizzare i partecipanti** sulle sfide dell'accessibilità web e sull'importanza di adottare pratiche di sviluppo inclusive.

Oltre a migliorare l'usabilità di Saharian, il test ha fornito preziose informazioni sulle competenze e sulla consapevolezza degli sviluppatori in materia di accessibilità web.

5.1.1 Problemi utilizzati per il test

Di seguito elencherò le pagine utilizzate nel test, spiegando anche che problemi presentava ognuna di esse e come risolverli. Questi primi tre esercizi sono quelli che il gruppo A ha svolto come primi:

- **Form**

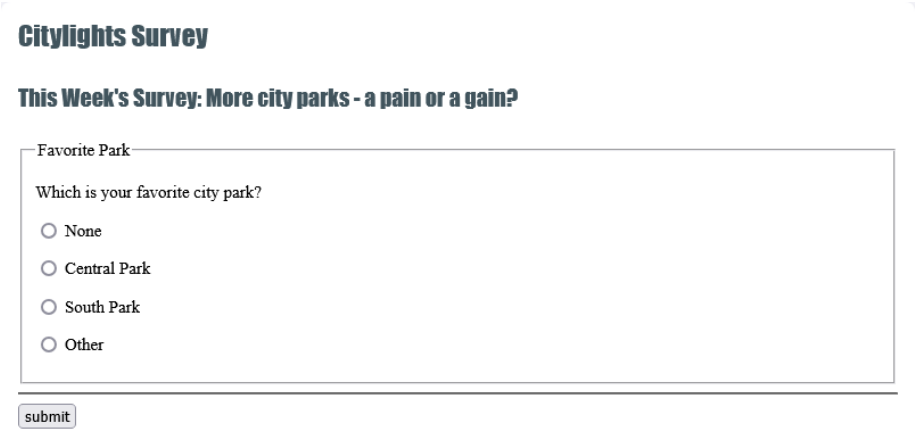

Figura 5.1: Form di tipo A

Si tratta di un form con quattro radiobutton. Il problema di accessibilità in questa pagina è la mancanza di una label associata ad ogni radiobutton. Questo rende difficoltoso per un utente che utilizza tecnologie assistive capire il significato di ognuno dei radiobutton. Per risolvere si può utilizzare l'attributo `aria-label` o `aria-labelledby`, oppure utilizzare un tag `label` con attributo `for` che fa riferimento al corretto radiobutton. Inoltre un altro problema presente è che la gerarchia delle intestazioni è resa corretta visivamente tramite delle regole CSS, ma non utilizza i corretti tag html, e pertanto non è accessibile. Per risolvere questo problema è sufficiente sostituire i due paragrafi con i tag `h1` e `h2` rispettivamente.

- **Table**

Enrollment Statistics

	Last Year			This Year		
	CS	Eng	Eco	CS	Eng	Eco
Undergraduate	1481	400	727	1500	340	777
Graduate	310	84	273	350	64	341

Figura 5.2: Table di tipo A

In questo esercizio troviamo una tabella, a cui manca una sezione di header ben definita dal tag `thead`. Le celle della tabella che fanno da intestazione sono create con una combinazione di `td` e `b`, il che fornisce un adeguato aspetto estetico, ma non è accessibile. Per risolvere questo problema è sufficiente spostare tutte le intestazioni in un tag `thead` adeguato e utilizzare per ognuno di essi il tag `th`. Questa tabella presenta inoltre degli header sulla riga, che vanno indicati con l'attributo `scope="row"`.

- **Tabs**

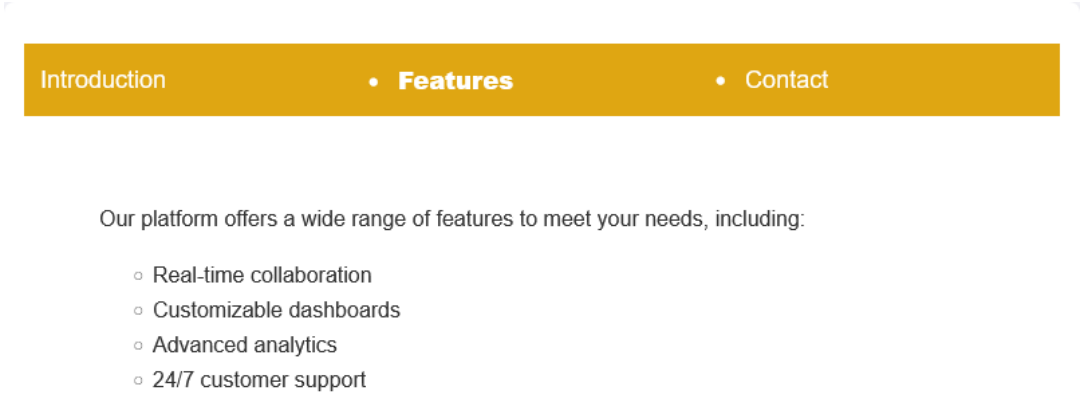

Figura 5.3: Tabs di tipo A

In questa pagina è presente una tablist, è implementata utilizzando il tag `ul`. Di conseguenza è necessario utilizzare i corretti ruoli ARIA, di cui questo componente è sprovvisto. Ogni `li` che funge da pulsante per il cambio tab non ha il ruolo `tab` né l'attributo `aria-controls`, il che impedisce alle tecnologie assistive di comprenderne il funzionamento. Per risolvere, bisogna aggiungere `role="tab"` agli elementi `li` che fungono da pulsanti e collegarli correttamente ai rispettivi pannelli con `aria-controls`. Inoltre, i pannelli stessi non hanno il ruolo `tabpanel`, che deve essere aggiunto per permettere agli screen reader di riconoscerli come contenuto dinamico legato ai `tab`. Infine, il contenitore dei tab non ha `role="tablist"`, che dovrebbe essere aggiunto per segnalare la presenza di una tablist allo screen reader.

Proseguendo, di seguito ci sono i test che il gruppo B ha eseguito per primi. Da notare che i componenti che fanno da soggetto principale sono sempre gli stessi, però il contenuto della pagina presenta delle differenze chiave:

- **Form**

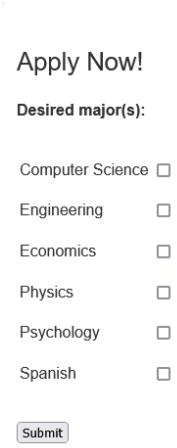

Figura 5.4: Form di tipo B

In questa versione del form, invece dei radiobutton, sono presenti delle checkbox che sono state raggruppate all'interno di una tabella. Tuttavia, l'uso della tabella per il raggruppamento non è semanticamente corretto, in quanto le tabelle dovrebbero essere utilizzate esclusivamente per dati tabulari. La soluzione migliore è racchiudere le checkbox all'interno di un elemento fieldset con una legend adeguata, così da fornire un contesto chiaro agli utenti.

- **Table**

ADULT	Front Seats	Rear Seats
Obelisks - Monkey, Monkey	$20.90	$20.90
Les Garçons de la Plage	$20.90	$27.90
Group (5 or more)		
Obelisks - Monkey, Monkey	$14.90	$14.90
Les Garçons de la Plage	$14.90	$20.90

Figura 5.5: Table di tipo B

Questa tabella presenta una struttura molto particolare, infatti ad ogni cella contente dati corrispondono tre header. Questo fa si che, per uno screen reader, le celle contenenti dati non sono correttamente associate alle relative

intestazioni. Ogni cella dovrebbe avere un attributo `headers` che elenca gli id delle intestazioni corrispondenti, in modo da fornire un chiaro riferimento tra dati e titoli. In secondo luogo, c'è l'assenza di un caption, che servirebbe a fornire una descrizione testuale del contenuto della tabella. Aggiungere un elemento `caption` all'inizio della tabella migliorerebbe la comprensione per chi utilizza tecnologie assistive.

- **Tabs**

Figura 5.6: Tabs di tipo B

Anche in questa pagina la tablist presenta dei problemi di accessibilità molto simili. Il div che contiene il gruppo di `tab` non ha il ruolo `tablist`, che dovrebbe essere presente per facilitare la navigazione. Due dei pulsanti che dovrebbero fungere da `tab` mancano del ruolo `tab`, e solo uno dei pannelli ha il ruolo `tabpanel`, mentre gli altri ne sono sprovvisti. Per garantire un'esperienza accessibile, è necessario assegnare correttamente `role="tab"` ai pulsanti e `role="tabpanel"` a tutti i pannelli associati. Un altro problema è che le immagini utilizzate non hanno un `alt`, il che rende impossibile per un utente non vedente comprenderne il contenuto. Aggiungere un `alt` descrittivo alle immagini risolverebbe il problema.

5.2 Valutazione oggettiva

Grazie alla pagina web sono riuscito a raccogliere i dati sul lavoro svolto dagli utenti del test e il tempo impiegato. Per ognuno di essi ho controllato quanti errori hanno corretto e calcolato una percentuale di completamento di ogni esercizio, mostrata nella seguente tabella:

Senza Saharian			Con Saharian		
Form A	Table A	Tabs A	Form B	Table B	Tabs B
100%	100%	66,67%	100%	50%	100%
100%	100%	100%	100%	100%	100%
100%	100%	100%	100%	50%	75%
Form B	Table B	Tabs B	Form A	Table A	Tabs A
100%	50%	75%	50%	100%	100%
100%	50%	75%	100%	50%	100%
100%	50%	100%	50%	100%	100%
Percentuale di completamento media					
85%			85%		

Tabella 5.1: Efficacia con e senza Saharian

Di seguito, invece, la tabella con i tempi di completamento:

Senza Saharian	Con Saharian
Test A	Test B
40 minuti	36 minuti
35 minuti	27 minuti
37 minuti	35 minuti
Test B	Test A
26 minuti	27 minuti
20 minuti	18 minuti
44 minuti	39 minuti
Tempo medio di completamento	
33 minuti	30 minuti

Tabella 5.2: Tempi di completamento con e senza Saharian

Prestazioni senza Saharian:

- **Efficacia**
  I partecipanti al test hanno risolto in media l'85% dei problemi senza utilizzare Saharian

- **Efficienza**
  I partecipanti hanno risolto i primi tre test senza Saharian in 33 minuti e 45 secondi

Prestazioni con Saharian:

- **Efficacia**
  I partecipanti al test hanno risolto in media l'85% dei problemi usando Saharian
- **Efficienza**
  I partecipanti hanno risolto i secondi tre test con Saharian in 30 minuti e 31 secondi

I risultati del test mostrano che l'uso di Saharian non ha influenzato la capacità dei partecipanti di identificare i problemi di accessibilità. Tuttavia, emerge una differenza nella difficoltà tra i test di tipo A, con una percentuale di completamento del 90%, e quelli di tipo B, completati solo nel 81% dei casi. Analizzando gli errori commessi, si nota che molti sono dovuti a distrazione: alcuni esercizi includevano un problema principale, più evidente, e un problema secondario, spesso più semplice ma meno immediato da individuare. Quest'ultimo è stato frequentemente trascurato, probabilmente perché dimenticato piuttosto che a causa di una reale difficoltà.

Questo suggerisce un possibile limite nella progettazione dei test, che potrebbero essere migliorati rendendo più esplicito il fatto che ogni esercizio contiene più problemi da risolvere, anche non direttamente correlati al nome del test.

Per quanto riguarda i tempi di completamento, la riduzione del 9% potrebbe sembrare marginale a prima vista. Tuttavia, considerando che i partecipanti utilizzavano Saharian per la prima volta, parte del tempo è stata spesa per familiarizzare con lo strumento e analizzare gli esempi forniti. Il fatto che, nonostante questo overhead iniziale, si sia comunque registrata una riduzione dei tempi indica che Saharian ha effettivamente contribuito a velocizzare la risoluzione dei test.

Un'analisi su un campione più ampio e con utenti più esperti nell'uso di Saharian potrebbe fornire risultati ancora più significativi, permettendo di valutare meglio l'impatto dello strumento nel lungo periodo.

5.3 Soddisfazione

Per valutare l'usabilità di Saharian è stata utilizzata la System Usability Scale (SUS) [Bro95], un questionario composto da dieci domande a cui si può rispondere con un punteggio da 1 a 5. Le domande sono standard e organizzate in modo tale che quelle

in posizione dispari siano positive, mentre quelle in posizione pari siano negative. Questo significa che, per le domande dispari, 5 rappresenta la valutazione migliore, mentre per le domande pari il valore più alto è 1.

Una volta raccolte le risposte, è possibile calcolare un punteggio complessivo compreso tra 0 e 100. Per effettuare il calcolo, si applica il seguente metodo: a ogni risposta in posizione dispari si sottrae 1, mentre per ogni risposta in posizione pari si sottrae il valore dalla cifra 5. Successivamente, la somma dei risultati viene moltiplicata per 2.5.

Esempio:

- domande dispari: $(4-1)+(3-1)+(5-1)+(3-1)+(3-1)=13$
- domande pari: $(5-2)+(5-3)+(5-1)+(5-2)+(5-2)=15$
- punteggio totale: $(13+15)*2.5=70$

Il punteggio finale ottenuto permette di valutare il livello di usabilità del progetto. Un risultato inferiore a 50 indica che l'usabilità non è accettabile. Un punteggio compreso tra 50 e 70 suggerisce un livello di usabilità marginale, mentre valori superiori a 70 indicano che l'usabilità è accettabile. In generale, un punteggio inferiore a 68 suggerisce la necessità di miglioramenti, mentre un valore superiore a questa soglia indica un livello accettabile, pur con possibili margini di ottimizzazione.

Al termine del test ogni utente ha valutato Saharian secondo la metrica SUS e qui di seguito sono presentati i risultati di tale valutazione:

Punteggio	**Valutazione**
77.5	accettabile
57.5	marginale
65	marginale
82.5	accettabile
80	accettabile
80	accettabile

Tabella 5.3: Punteggi SUS

5.4 Feedback degli utenti

Sono stati raccolti dei feedback testuali dai partecipanti. Di seguito viene analizzato il loro contenuto per individuare i punti di forza e le aree di miglioramento.

Tra gli aspetti positivi sottolineati dai partecipanti emergono la semplicità d'uso e il supporto durante lo sviluppo. Molti hanno apprezzato il feedback visivo, che consente di comprendere rapidamente le modifiche effettuate per migliorare l'accessibilità. Da questi feedback sembra che Saharian raggiunga il suo obiettivo di semplificare la vita allo sviluppatore tramite una rappresentazione grafica intuitiva e contribuisca a risolvere il problema della mancata percezione dei problemi di accessibilità di una pagina. Questo aiuta a comprendere quando il livello di accessibilità è sufficiente, fornendo un supporto visivo che guida le correzioni necessarie. Inoltre, la schermata di troubleshooting è stata ritenuta utile, probabilmente perché fornisce una guida rapida in un campo in cui le informazioni sono spesso sparse e frammentate.

D'altra parte, alcuni partecipanti hanno segnalato difficoltà nel riconoscere graficamente un elemento accessibile, il che potrebbe spiegare alcune valutazioni non del tutto positive. Tuttavia, come detto precedentemente, altri utenti hanno trovato gli elementi grafici chiari e intuitivi. Questa discrepanza potrebbe avere due spiegazioni: la prima riguarda una possibile incoerenza visiva, per cui alcuni componenti risultano chiari e intuitivi, mentre altri meno; la seconda ipotesi è legata al tempo di familiarizzazione con lo strumento, ovvero alcuni utenti potrebbero aver dedicato più tempo a osservare gli esempi per capirne il funzionamento.

Un'altra criticità emersa riguarda l'usabilità del livello *Max*, che impatta significativamente sulla struttura della pagina, rendendola difficile da utilizzare. Inoltre, il fatto di dover disattivare frequentemente Saharian dopo ogni modifica ostacola un'esperienza fluida. Una possibile soluzione potrebbe essere l'introduzione di scorciatoie da tastiera per semplificare il passaggio tra le modalità.

Infine, un'ultima osservazione riguarda il ruolo di Saharian rispetto alla conoscenza di ARIA. Anche se il tool facilita l'identificazione degli errori di accessibilità, non può sostituire una conoscenza di base di ARIA. Questo suggerisce che, oltre al miglioramento dello strumento, sarebbe utile fornire risorse per aiutare gli sviluppatori a comprendere meglio le pratiche di accessibilità.

Capitolo 6

Conclusioni

Saharian nasce con l'obiettivo di rendere il web più accessibile, migliorando l'esperienza di lavoro degli sviluppatori. Ancora oggi, l'accessibilità delle pagine web è spesso trascurata e considerata secondaria. Per affrontare questo problema, Saharian offre una rappresentazione visiva intuitiva della pagina, basandosi sullo standard ARIA.

Questa tesi si propone di colmare le lacune di Saharian nel supporto agli standard ARIA, poiché alcuni attributi non erano riconosciuti dallo strumento, riducendone l'efficacia. Inoltre, il livello di attivazione *Max* risultava incompleto: questa modalità mira a fornire una prospettiva alternativa sull'accessibilità, evidenziando problematiche specifiche. Un ulteriore obiettivo è stato il miglioramento della documentazione di Saharian, con l'integrazione di una sezione di troubleshooting, esempi di implementazioni accessibili comuni e una guida all'uso dello strumento.

Infine, Saharian era compatibile esclusivamente con Chrome. Un obiettivo chiave è stato quindi estendere il supporto a più browser, rendendolo accessibile a un numero maggiore di sviluppatori.

È stato aggiunto il supporto per diversi attributi ARIA di tipo *role*:

- `role="term"`
- `role="definition"`
- `role="log"`
- `role="progressbar"`
- `role="menu"`
- `role="menubar"`
- `role="strong"`
- `role="emphasis"`
- `role="meter"`
- `role="marquee"`
- `role="figure"`
- `role="caption"`
- `role="insertion"`
- `role="deletion"`
- `role="scrollbar"`

- `role="toolbar"`
- `role="blockquote"`
- `role="slider"`
- `role="code"`
- `role="note"`

Inoltre, è stato aggiunto il supporto per attributi ARIA di stato e proprietà:

- `aria-invalid`
- `aria-modal`
- `aria-colcount`
- `aria-errormessage`
- `aria-sort`
- `aria-colindex`
- `aria-busy`
- `aria-placeholder`
- `aria-rowcount`
- `aria-flowto`
- `aria-multiline`
- `aria-rowindex`
- `aria-haspopup`
- `aria-multiselectable`
- `aria-required`
- `headers`

Questo ha portato lo strumento a supportare tutti i ruoli ARIA e il 83% degli attributi di stato e proprietà. In particolare delle tre categorie di stato e proprietà mancano:

- Attributi delle Live region: `aria-relevant`
- Attributi di relazione: `aria-colspan`, `aria-details`, `aria-owns`, `aria-posinset`, `aria-readonly`, `aria-rowspan`, `aria-setsize`.

Il livello *Max* è stato migliorato nei seguenti aspetti:

- **Landmark**: evidenzia chiaramente i landmark della pagina.
- **Immagini decorative**: le immagini decorative vengono mostrate e identificate.
- **Video**: i video non accessibili vengono segnalati con un messaggio esplicativo.
- **Live region**: le live region di tipo assertive vengono spostate lontano dal contenuto principale quando aggiornate.
- **Application/Document mode**: aggiunto uno switch per passare dalla modalità Application, che mostra solo gli elementi interattivi, alla modalità Document, che visualizza l'intera pagina.

L'interfaccia dell'estensione include ora una sezione di troubleshooting con spiegazioni, suggerimenti ed esempi per migliorare l'accessibilità dei componenti web basilari. Inoltre, è stata realizzata una pagina web con istruzioni su installazione, utilizzo e test dell'estensione, con esempi pratici su `video`, `landmark`, `flowto` e `scrollbar`.

Sono state anche migliorate le pagine di esempio preesistenti, in particolare `tabs`, `progressbar`, `checkbox` e `term definition`.

Saharian è ora compatibile con tutti i browser basati su Chromium, come Chrome, Edge e Opera, ed è stato esteso anche a Mozilla Firefox. Il supporto per Safari non è ancora stato implementato.

L'analisi dei test condotti con Saharian evidenzia il potenziale dello strumento nell'assistere gli sviluppatori nell'individuare e correggere problematiche di accessibilità. Sebbene non si sia registrato un aumento significativo delle risposte corrette nei test, è stato osservato un miglioramento del 9% nei tempi di completamento, supportato da feedback qualitativi che confermano una maggiore facilità nell'implementazione di pagine web accessibili. I risultati ottenuti secondo la metrica di valutazione SUS, indicano che la soddisfazione nell'utilizzo dello strumento è a un buon livello, però può decisamente essere migliorata. In particolare le valutazioni che hanno evidenziato un livello di soddisfazione marginale attribuiscono la causa al dover capire molte cose prima di poter usare Saharian con facilità. Invece gli aspetti positivi evidenziati dalle valutazione accettabili, indicano una accentuata facilità di utilizzo e comprensione dell'interfaccia dello strumento.

Nonostante i buoni risultati, vi sono margini di miglioramento. Saharian, pur offrendo già un supporto efficace agli sviluppatori, può essere ulteriormente ottimizzato per migliorarne l'efficienza e l'esperienza d'uso. Questi possibili miglioramenti includono:

- **evitare polling**: Saharian utilizza il polling per mantenere la pagina aggiornata quando subisce delle modifiche. Il lato negativo di questa tecnica è che nel caso di grandi contenuti si rischia di effettuare un aggiornamento della pagina ancor prima che quello precedente sia terminato. Potrebbe essere più efficiente modificare la logica e renderla reattiva, tramite dei listener di eventi o tecniche simili.
- **area di influenza**: Per evitare che pagine con molti contenuti influiscano negativamente sull'esperienza si potrebbe implementare una modalità che permette allo sviluppatore di scegliere un'area della pagina e applicare la trasformazione di Saharian solo a quella porzione. Questo potrebbe diminuire drasticamente il numero di elementi del DOM coinvolti, riducendo quindi il tempo di caricamento.

Uno degli aspetti di miglioramento fondamentali riguarda l'usabilità. Attualmente, l'interazione con Saharian avviene principalmente tramite l'interfaccia grafica, ma l'introduzione di shortcut da tastiera potrebbe rendere il processo più fluido, permettendo agli sviluppatori di cambiare rapidamente il livello di accessibilità senza dover

navigare attraverso i menu. Inoltre, una migliore organizzazione delle impostazioni e delle informazioni di feedback potrebbe facilitare la comprensione dei problemi evidenziati. Un altro ambito di miglioramento sarebbe implementare la compatibilità dell'estensione con Safari, così da renderla accessibile al maggior numero possibile di sviluppatori. Infine, Saharian necessita di una documentazione più dettagliata sui suoi meccanismi interni, facilitando il contributo di altri sviluppatori al progetto.

In conclusione, Saharian rappresenta un passo importante verso un web più accessibile, fornendo agli sviluppatori uno strumento efficace per migliorare l'accessibilità delle loro pagine. Continuando a perfezionarlo, basandosi sui feedback degli utenti e sull'analisi dei dati di utilizzo, si potrà contribuire in modo significativo alla creazione di un internet realmente accessibile a tutti.

Bibliografia

[Bro95] John Brooke. «SUS: A quick and dirty usability scale». In: *Usability Eval. Ind.* 189 (nov. 1995).

[DGC18] Joanmarie Diggs, Bryan Garaventa e Michael Cooper. *Accessible name and description computation 1.1*. 18 Dic. 2018. URL: https://www.w3.org/TR/2018/REC-accname-1.1-20181218/.

[Mor18] Luca Morosini. «Saharian: Uno strumento visuale per la progettazione di pagine web accessibili». 2018.

[Com19] European Commission. *European accessibility act*. 2019.

[acc21] accessi. *The Ultimate Guide to Compliance with The European Accessibility Act (EAA)*. 2021. URL: https://www.accessi.org/blog/european-accessibility-act/.

[W3C22] World Wide Web Consortium (W3C). *Using ARIA*. 2022. URL: https://www.w3.org/TR/using-aria/.

[W3C23a] World Wide Web Consortium (W3C). *WAI-ARIA 1.2. Supported States and Properties*. Giu. 2023. URL: https://www.w3.org/TR/wai-aria/#states_and_properties.

[W3C23b] World Wide Web Consortium (W3C). *WAI-ARIA 1.2. The Roles Model*. Giu. 2023. URL: https://www.w3.org/TR/wai-aria/#roles.

[Rub23] Vincenzo Rubano. «On making web accessibility more accessible: Strategy and tools for social good». Tesi di dott. 2023.

[W3C24a] World Wide Web Consortium (W3C). *WAI-ARIA Overview*. 2024. URL: https://www.w3.org/WAI/standards-guidelines/aria/.

[W3C24b] World Wide Web Consortium (W3C). *Web content accessibility guidelines (wcag) 2.2*. 2024. URL: https://www.w3.org/WAI/standards-guidelines/wcag/.

[Fer24] Anselmo Ferrari. «Raffinamento e Ampliamento di SAHARIAN: Un Contributo allo Sviluppo di Strumenti per l'Accessibilità Web». 2024.

[Tea24] accessiBe Team. *The Top Web Accessibility Tools for 2024*. 2024. URL: https://accessibe.com/blog/knowledgebase/top-web-accessibility-tools.

[W3C25a] World Wide Web Consortium (W3C). *Bypass Blocks*. Feb. 2025. URL: https://www.w3.org/WAI/WCAG21/Understanding/bypass-blocks.html (visitato il giorno 02/2025).

[W3C25b] World Wide Web Consortium (W3C). *Web accessibility evaluation tools list*. Feb. 2025. URL: https://www.w3.org/WAI/test-evaluate/tools/list/ (visitato il giorno 02/2025).

[Acc25] NV Access. *About NVDA*. Feb. 2025. URL: https://www.nvaccess.org/about-nvda/ (visitato il giorno 02/2025).

[Deq25] Deque. *axe Browser Extensions for Accessibility Testing*. Feb. 2025. URL: https://www.deque.com/axe/browser-extensions/ (visitato il giorno 02/2025).

[fou25] Access for all foundation. *Bad ARIA practices*. Feb. 2025. URL: https://www.accessibility-developer-guide.com/knowledge/aria/bad-practices/ (visitato il giorno 02/2025).

[Goo25] Google. *Lighthouse Overview*. Feb. 2025. URL: https://developer.chrome.com/docs/lighthouse/ (visitato il giorno 02/2025).

[Rub25] Vincenzo Rubano. *A11A, All about accessibility!* Feb. 2025. URL: https://a11a.disi.unibo.it/ (visitato il giorno 02/2025).

[Sci25] Freedom Scientific. *JAWS Screen Reader*. Feb. 2025. URL: https://www.freedomscientific.com/products/software/jaws/ (visitato il giorno 02/2025).

[Web25] WebAIM. *WAVE Web Accessibility Evaluation Tools*. Feb. 2025. URL: https://wave.webaim.org/ (visitato il giorno 02/2025).

Ringraziamenti

Ringrazio il Prof. Vitali, per avermi guidato e supportato nella fase più importante del mio percorso accademico. Ringrazio i miei amici, con cui ho condiviso tanti momenti divertenti e tanti momenti difficili, per avermi accompagnato lungo questo percorso. Ringrazio i miei genitori per avermi motivato e sostenuto in tutte le difficoltà.