Accessibilità per gli sviluppatori web

Migliorare l'accessibilità rende il tuo sito più utilizzabile per tutti.

Addy Osmani
(https://twitter.com/addyosmani) (https://github.com/addyosmani)

Suggerimento: segui il corso Scopri l'accessibilità (https://web.dev/learn/accessibility?hl=it), un corso di riferimento sull'accessibilità sempre aggiornato per migliorare le tue competenze di sviluppo web.

È importante creare siti inclusivi e accessibili a tutti. Esistono almeno sei aree chiave di disabilità per le quali puoi eseguire l'ottimizzazione: **visiva**, **uditiva**, **mobilità**, **cognitiva**, **linguistica** e **neurologica**. In questo caso, possono essere utili molti strumenti e risorse, anche se non hai mai affrontato l'accessibilità web.

Oltre un miliardo di persone
(https://www.who.int/teams/noncommunicable-diseases/sensory-functions-disability-and-rehabilitation/world-report-on-disability)
vivono con qualche forma di disabilità.

Per essere accessibili, i siti devono funzionare su più dispositivi con schermi di varie dimensioni e diversi tipi di input, come gli screen reader. Inoltre, i siti devono essere utilizzabili dal più ampio gruppo di utenti, inclusi quelli con disabilità.

Ecco alcune disabilità che i tuoi utenti potrebbero avere:

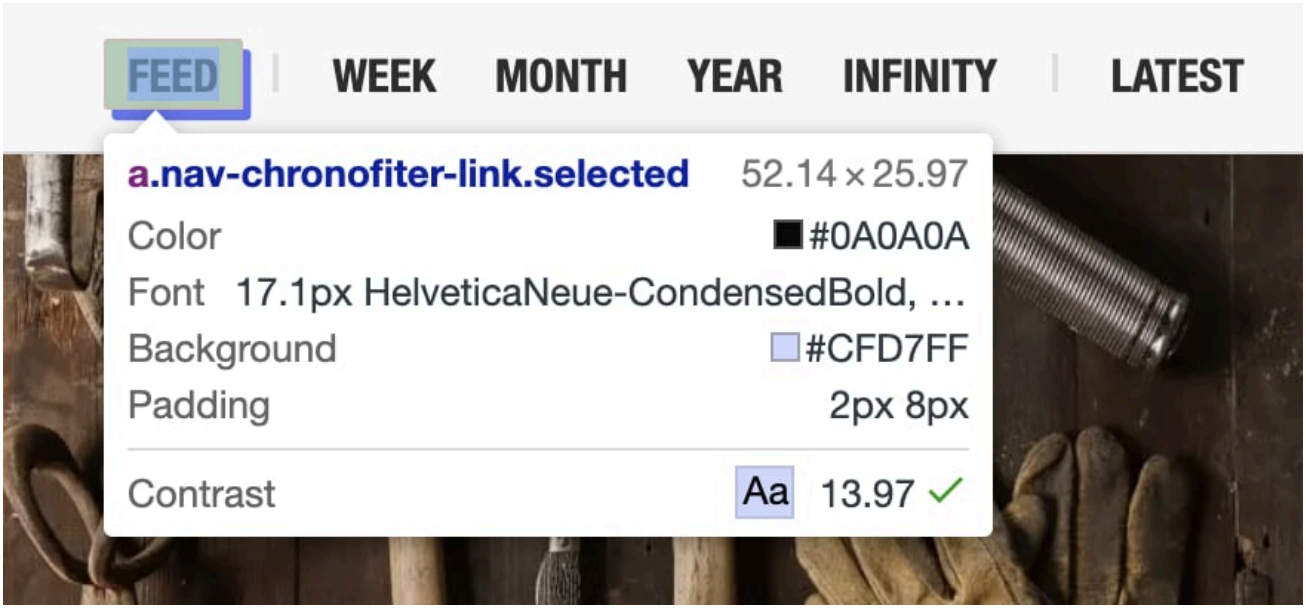
Vision	Audizione	Mobilità
• Ipovedenti	• Disabilità uditiva	• Lesione del midollo spinale

Vision	Audizione	Mobilità
• Cecità • Daltonismo • Cataratte • Bagliore solare	• Sordità • Rumore • Infezione dell'orecchio	• Destrezza limitata • Mani occupate
Cognitivo	**Voce**	**Neurale**
• Disturbo dell'apprendimento • Sonnolenza o distrazione • Emicrania • Autismo • Attacco epilettico	• Rumore ambientale • Mal di gola • Impedimento alla comunicazione • Impossibile parlare	• Depressione • PTSD • Disturbo bipolare • Ansia

I **problemi visivi** vanno dall'incapacità di distinguere i colori alla cecità totale.

- Assicurati che i contenuti di testo soddisfino una soglia minima del rapporto di contrasto (http://www.w3.org/TR/WCAG20/#visual-audio-contrast-contrast).
- Evita di comunicare informazioni utilizzando solo il colore (http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast-without-color) e assicurati che tutto il testo sia ridimensionabile (http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast-scale).
- Assicurati che tutti i componenti dell'interfaccia utente possano essere utilizzati con tecnologie per la disabilità come screen reader, lenti d'ingrandimento e display braille. Ciò comporta l'assicurarsi che i componenti dell'interfaccia utente siano codificati in modo che le API di accessibilità possano determinare in modo programmatico il *ruolo*, lo *stato*, il *valore* e il *titolo* di qualsiasi elemento.

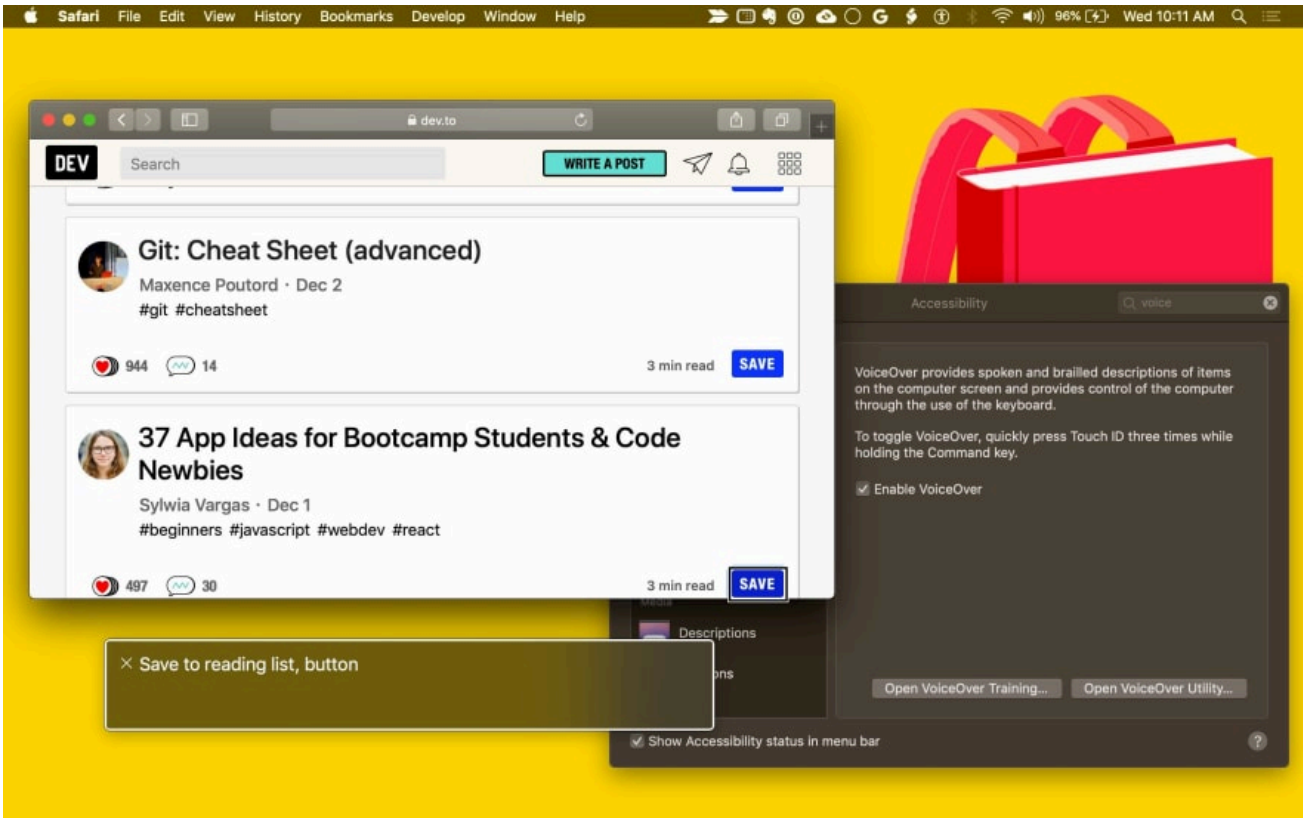
Suggerimento: la funzionalità di ispezione degli elementi nel browser mostra una descrizione comando delle proprietà CSS che include un rapido controllo del rapporto di contrasto del colore.



Personalmente ho una disabilità visiva e spesso mi capita di aumentare lo zoom dei siti, delle relative DevTools e del terminale. Sebbene il supporto dello zoom non sia quasi mai in cima alle liste di cose da fare degli sviluppatori, può fare un'enorme differenza per gli utenti come me.

Per **problemi di udito** si intende che un utente potrebbe avere difficoltà a sentire l'audio emesso da una pagina.

- Fornisci alternative testuali (http://www.w3.org/TR/WCAG20/#media-equiv-av-only-alt) per tutti i contenuti che non sono strettamente di testo.
- Verifica che i componenti dell'interfaccia utente siano ancora funzionali senza audio (http://www.w3.org/TR/2008/REC-WCAG20-20081211/#content-structure-separation-understanding).



I **problemi di mobilità** possono includere l'impossibilità di utilizzare un mouse, una tastiera o un touchscreen.

- Rendi i contenuti dei componenti dell'interfaccia utente funzionalmente accessibili da una tastiera (http://www.w3.org/TR/wai-aria-practices/#keyboard) per tutte le azioni per le quali altrimenti si userebbe un mouse.
- Assicurati che le pagine siano contrassegnate correttamente per le tecnologie per la disabilità, tra cui screen reader, software di controllo vocale e controlli degli interruttori fisici, che tendono a utilizzare le stesse API.

Per **problemi cognitivi** si intende che un utente potrebbe aver bisogno di tecnologie per la disabilità per leggere il testo, pertanto è importante assicurarsi che esistano alternative di testo.

- Fai attenzione quando utilizzi le animazioni. Evita video e animazioni che si ripetono (http://www.w3.org/TR/WCAG20/#time-limits) o lampeggiano, perché potrebbero causare problemi (http://www.w3.org/TR/WCAG20/#seizure) per alcuni utenti.

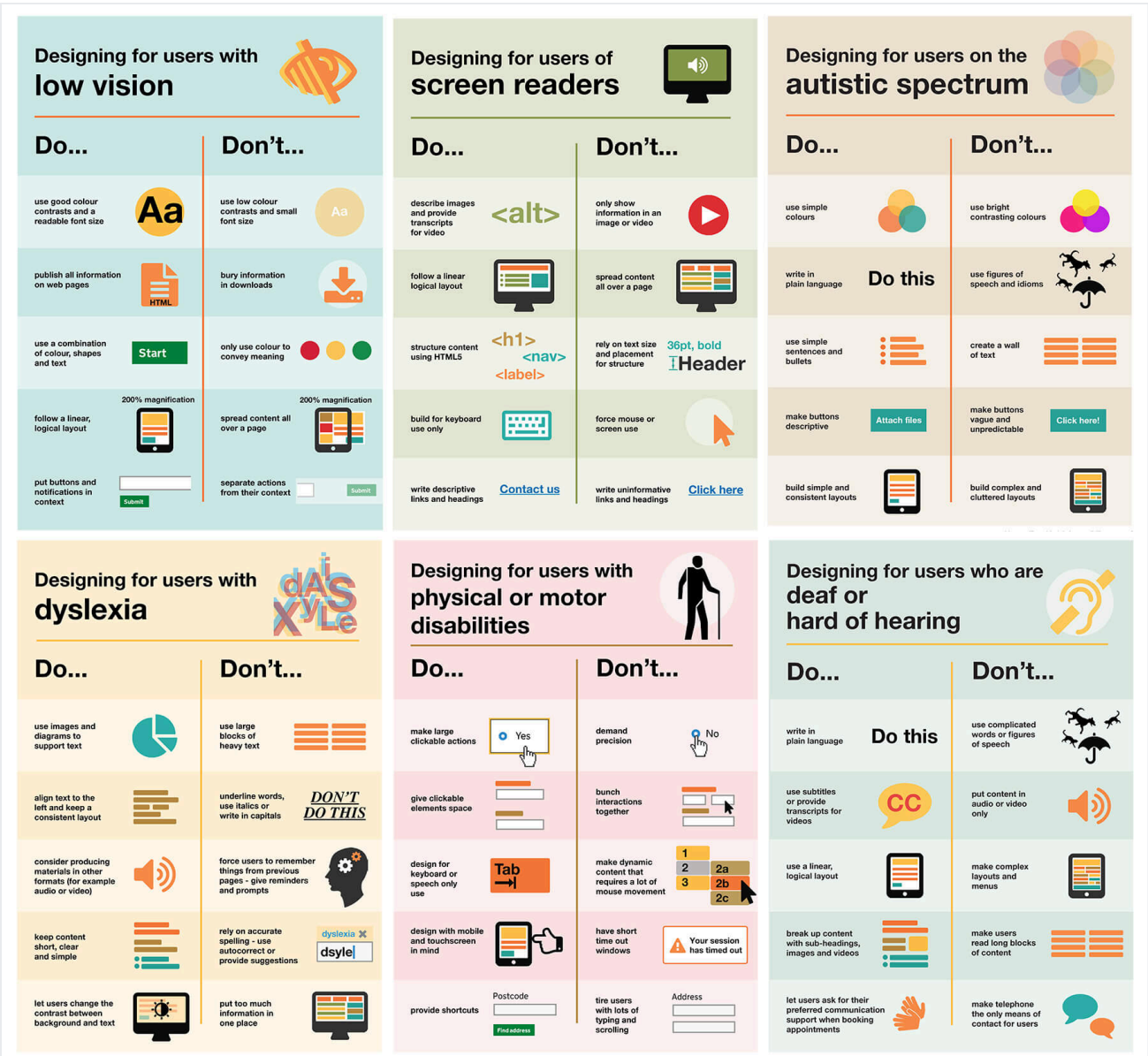
  La query supporti CSS `prefers-reduced-motion` (https://web.dev/articles/prefers-reduced-motion?hl=it#too_much_motion_in_real_life_and_on_the_web) consente di limitare le animazioni e la riproduzione automatica dei video per gli utenti che preferiscono ridurre il movimento:

```
/*
If the user expresses a preference for reduced motion, don't use animations or
*/
@media (prefers-reduced-motion: reduce) {
  button {
    animation: none;
  }
}
```

- Evita le interazioni basate su tempi (https://www.w3.org/WAI/WCAG21/Understanding/no-timing.html).

Potrebbe sembrare che ci siano molti aspetti da considerare, ma ti guideremo attraverso la procedura per valutare e poi migliorare l'accessibilità dei componenti dell'interfaccia utente.

Per un'assistenza visiva aggiuntiva, il team di accessibilità di GOV.UK ha creato una serie di poster digitali sui do's e don'ts dell'accessibilità (https://accessibility.blog.gov.uk/2016/09/02/dos-and-donts-on-designing-for-accessibility/) che puoi utilizzare per condividere le best practice con il tuo team.



Sei poster che elencano le best practice di accessibilità. Leggi il testo integrale (https://ukhomeoffice.github.io/accessibility-posters/).

Misurare l'accessibilità dei componenti dell'interfaccia utente

Quando controlli l'accessibilità dei componenti dell'interfaccia utente della tua pagina, chiediti:

- **Puoi utilizzare il componente dell'interfaccia utente solo con la tastiera?**

  Il componente gestisce lo stato attivo ed evita le trappole dello stato attivo? Può rispondere agli eventi della tastiera appropriati?

- **Puoi utilizzare il componente dell'interfaccia utente con uno screen reader?**

  Hai fornito alternative di testo per le informazioni presentate visivamente? Hai aggiunto informazioni semantiche utilizzando ARIA?

- **Il componente dell'interfaccia utente può funzionare senza audio?**

  Disattiva gli altoparlanti ed esamina i casi d'uso.

- **Il componente dell'interfaccia utente può funzionare senza colore?**

  Assicurati che il componente dell'interfaccia utente possa essere utilizzato da una persona che non vede i colori. Uno strumento utile per simulare il daltonismo è un'estensione di Chrome chiamata Colorblindly (https://chromewebstore.google.com/detail/colorblindly/floniaahmccleoclneebhhmnjgdfijgg?hl=it). Prova tutte e quattro le forme di simulazione della daltonismo disponibili. Ti potrebbe interessare anche l'estensione Daltonize (https://chromewebstore.google.com/detail/daltonize/obcnmdgpjakcffkcjnonpdlainhphpgh?hl=it), che è altrettanto utile.

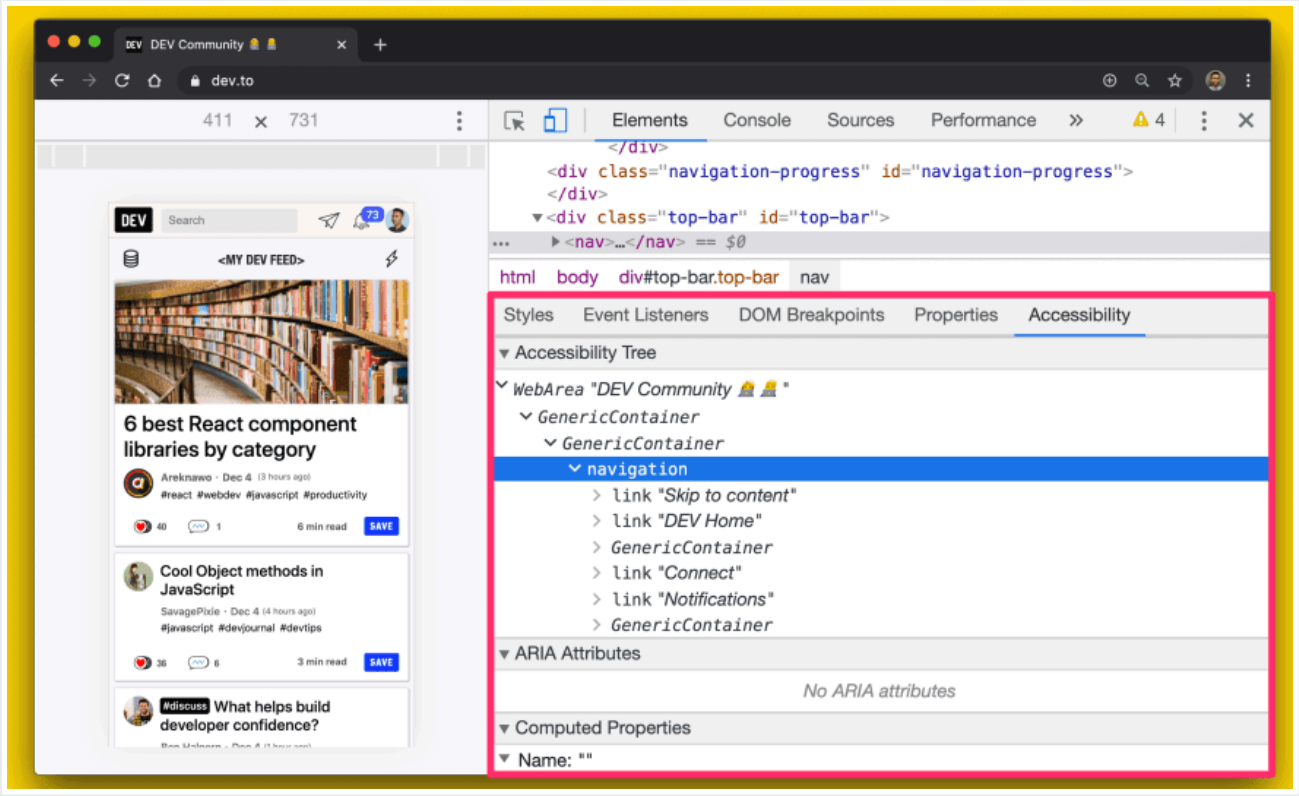
- **Il componente dell'interfaccia utente può funzionare con la modalità ad alto contrasto attivata?**

  Tutti i sistemi operativi moderni supportano una modalità ad alto contrasto. Alto contrasto (https://chrome.google.com/webstore/detail/high-contrast/djcfdncoelnlbldjfhinnjlhdjlikmph?hl=it) è un'estensione di Chrome che può aiutarti.

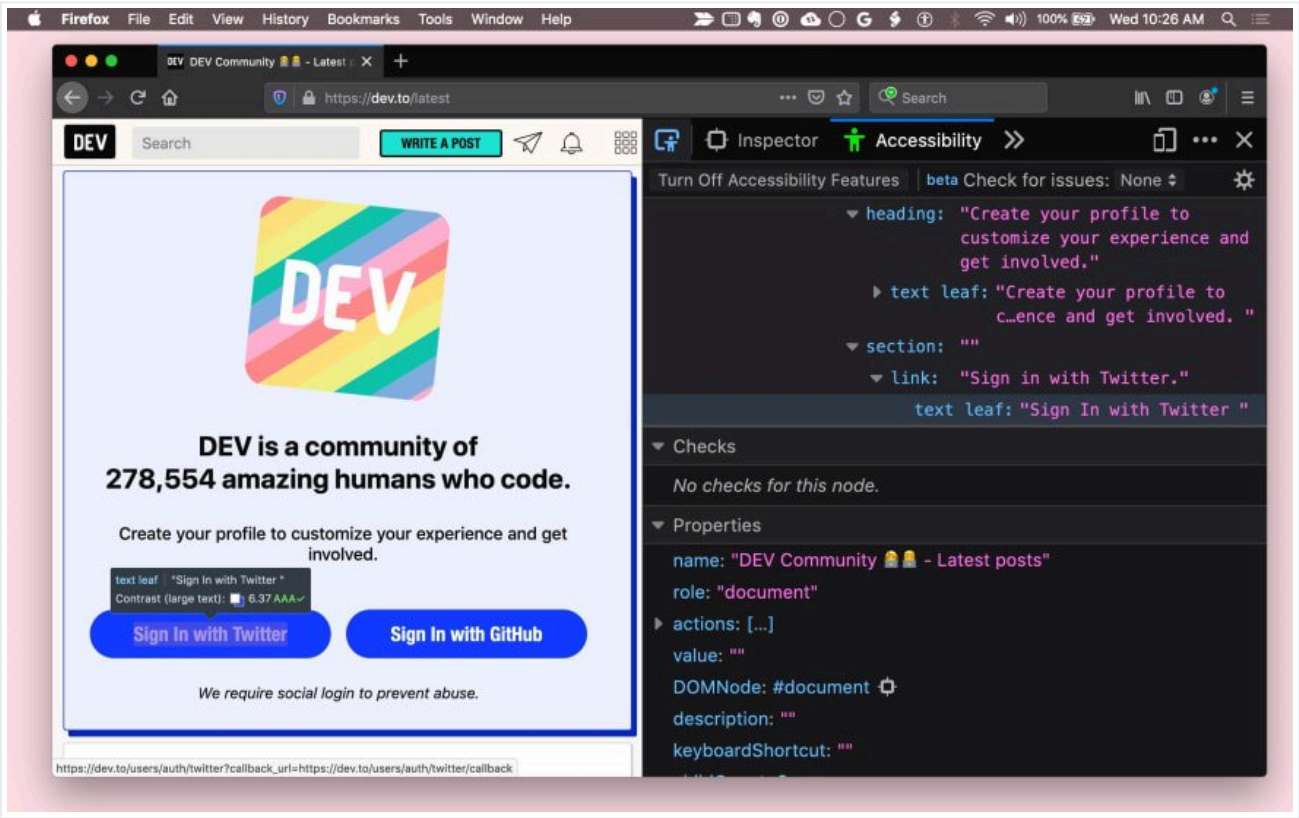
I controlli standardizzati (ad esempio `<button>` e `<select>`) hanno l'accessibilità integrata nel browser. Possono essere selezionati utilizzando il tasto `Tab`; rispondono agli eventi della tastiera (come `Enter`, `Space` e i tasti freccia); e hanno ruoli, stati e proprietà semantici utilizzati dagli strumenti di accessibilità. Inoltre, lo stile predefinito deve soddisfare i requisiti di accessibilità elencati.

I componenti dell'interfaccia utente personalizzati (ad eccezione dei componenti che estendono elementi standard come `<button>`) non hanno funzionalità integrate, inclusa l'accessibilità, quindi devi fornirle. Un buon punto di partenza per implementare l'accessibilità è confrontare il componente con un elemento standard analogo (o una combinazione di più elementi standard, a seconda della complessità del componente).

La maggior parte degli strumenti per gli sviluppatori del browser supporta l'ispezione dell'albero di accessibilità di una pagina. In Chrome DevTools, questa opzione è disponibile nella scheda **Accessibilità** del riquadro **Elementi**.



Firefox ha anche un pannello **Accessibilità**.



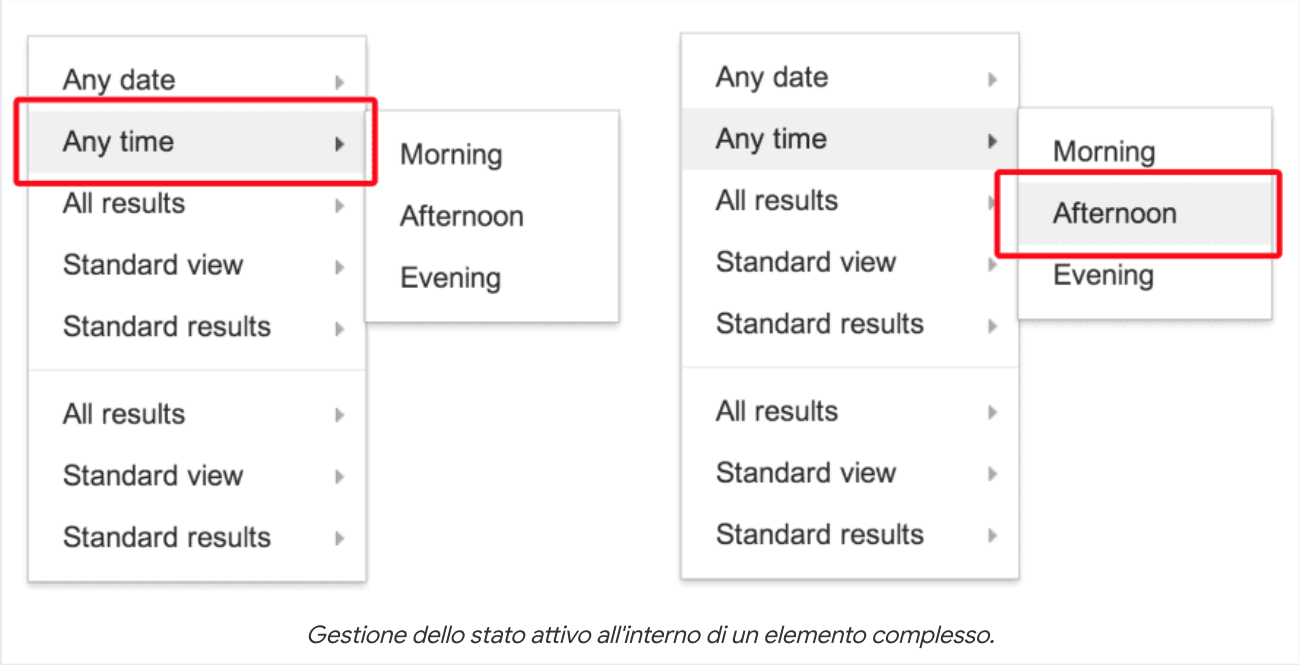
Safari mostra le informazioni sull'accessibilità nella scheda **Nodo** del riquadro **Elementi**.

Di seguito è riportato un elenco di domande che puoi porti quando cerchi di rendere più accessibili i componenti dell'interfaccia utente.

Migliorare lo stato attivo della tastiera

Idealmente, assicurati che sia possibile accedere a tutte le funzionalità del componente dell'interfaccia utente con una tastiera. Quando progetti l'esperienza utente, pensa a come utilizzeresti l'elemento solo con la tastiera e trova un insieme coerente di interazioni con la tastiera.

Innanzitutto, assicurati di avere un target di destinazione ragionevole per ogni componente. Ad esempio, un componente complesso come un menu può essere un target di immissione all'interno di una pagina, ma deve gestire l'immissione al suo interno in modo che l'elemento del menu attivo acquisisca sempre l'immissione.



Gestione dello stato attivo all'interno di un elemento complesso.

Utilizzare tabindex

Puoi aggiungere il fuoco della tastiera per elementi e componenti dell'interfaccia utente con l'attributo `tabindex`. Gli utenti che utilizzano solo la tastiera e le tecnologie per la disabilità devono essere in grado di posizionare il fuoco della tastiera sugli elementi per interagire con loro.

Gli elementi interattivi integrati (come `<button>`) sono attivabili per impostazione predefinita, quindi non richiedono un attributo `tabindex`, a meno che non sia necessario modificare la loro posizione nell'ordine di tabulazione.

Esistono tre tipi di valori `tabindex`:

- `tabindex="0"` è il più comune e posiziona l'elemento nell'ordine naturale delle schede (definito dall'ordine DOM).
- Un valore `tabindex` uguale a -1 fa sì che l'elemento sia acquisibile tramite programmazione, ma non nell'ordine di tabulazione.
- Un valore `tabindex` maggiore di 0 inserisce l'elemento in un ordine di tabulazione manuale. Tutti gli elementi della pagina con un valore `tabindex` positivo vengono visitati in ordine numerico, prima degli elementi nell'ordine naturale delle schede.

Avviso: come regola generale, evita gli indici di tabulazione positivi. Interrompere l'ordine normale di messa a fuoco potrebbe confondere e frustrare gli utenti. È quindi probabile che causi un mancato rispetto del criterio di successo 2.4.3 (https://www.w3.org/TR/WCAG22/#focus-order).

Puoi trovare alcuni casi d'uso per `tabindex` nell'articolo Utilizzare tabindex (https://web.dev/articles/using-tabindex?hl=it).

Assicurati che la messa a fuoco sia sempre visibile, utilizzando lo stile dell'anello di messa a fuoco predefinito o applicando uno stile di messa a fuoco personalizzato distinguibile. Ricorda di non bloccare gli utenti che usano la tastiera: devono essere in grado di spostare lo stato attivo da un elemento utilizzando solo la tastiera.

Nota: potresti anche essere interessato agli approcci `tabindex` o `aria-activedescendant` itineranti, come descritto su MDN (https://developer.mozilla.org/docs/Web/Accessibility/Keyboard-navigable_JavaScript_widgets#Technique_1_Roving_tabindex)
.

Utilizzare la messa a fuoco automatica

L'attributo autofocus HTML consente all'autore di specificare che un determinato elemento deve acquisire automaticamente il focus al caricamento della pagina. `autofocus` è già supportato su tutti i controlli dei moduli web (https://html.spec.whatwg.org/multipage/forms.html#association-of-controls-and-forms), inclusi i pulsanti. Per impostare il fuoco automatico sugli elementi nei tuoi componenti dell'interfaccia utente personalizzati, chiama il metodo `focus()` (https://developer.mozilla.org/docs/Web/API/HTMLElement.focus), supportato su tutti gli elementi HTML

su cui è possibile impostare il fuoco (ad esempio `document.querySelector('myButton').focus()`).

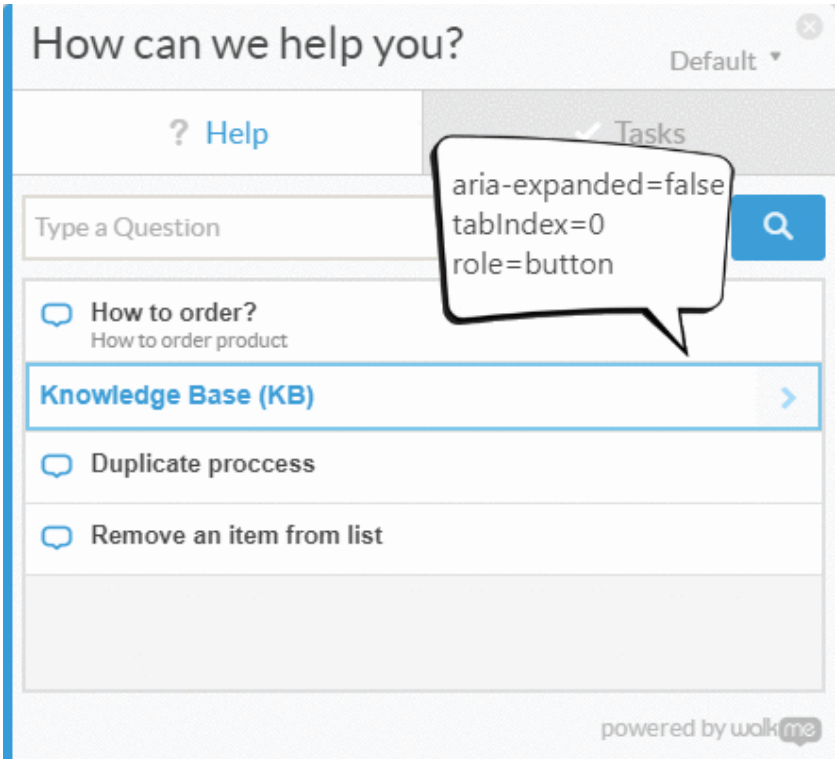
Aggiungere l'interazione con la tastiera

Una volta che il componente dell'interfaccia utente è attivabile, fornisci una buona storia di interazione con la tastiera quando un componente è attivo gestendo gli eventi appropriati della tastiera. Ad esempio, consenti all'utente di utilizzare i tasti Freccia per selezionare le opzioni di menu e `Space` o `Enter` per attivare i pulsanti. La guida ai pattern di progettazione (http://www.w3.org/TR/wai-aria-practices/#aria_ex) ARIA fornisce alcune indicazioni in merito.

Infine, assicurati che le scorciatoie da tastiera siano rilevabili. Una pratica comune è avere una legenda delle scorciatoie da tastiera (testo sullo schermo) per informare l'utente della presenza di scorciatoie. Ad esempio, "Premi ? per le scorciatoie da tastiera". In alternativa, un suggerimento come una descrizione comando può essere utilizzato per informare l'utente di una scorciatoia.

L'importanza della gestione dell'attenzione non può essere sottovalutata. Un esempio importante è un riquadro di navigazione. Se aggiungi un componente dell'interfaccia utente alla pagina, devi indirizzare lo stato attivo a un elemento al suo interno; in caso contrario, gli utenti potrebbero dover scorrere l'intera pagina per accedervi. Questa può essere un'esperienza frustrante, quindi assicurati di testare lo stato attivo per tutti i componenti della pagina navigabili con la tastiera.

Suggerimento: puoi utilizzare Puppeteer (https://github.com/puppeteer/puppeteer) per automatizzare l'esecuzione di test di accessibilità da tastiera per l'attivazione/disattivazione degli stati dell'interfaccia utente. WalkMe Engineering (https://medium.com/walkme-engineering/web-accessibility-testing-d499a7f7a032) ha una guida eccezionale su questo argomento che ti consiglio di leggere.




```
// Example for expanding and collapsing a category with the Space key
const category = await page.$(`.category`);

// verify tabIndex, role and focus
expect(await page.evaluate(elem => elem.getAttribute(`role`), category)).toEqual(`
expect(await page.evaluate(elem => elem.getAttribute(`tabindex`), category)).toEqu
expect(await page.evaluate(elem => window.document.activeElement === elem, categor

// verify aria-expanded = false
expect(await page.evaluate(elem => elem.getAttribute(`aria-expanded`), category)).

// toggle category by pressing Space
await page.keyboard.press('Space');

// verify aria-expanded = true
expect(await page.evaluate(elem => elem.getAttribute(`aria-expanded`), category)).
```

Garantire l'utilizzo corretto dello screen reader

Circa l'1-2% delle persone utilizza uno screen reader. Riesci a comprendere tutte le informazioni importanti e a interagire con il componente utilizzando solo lo screen reader e la tastiera?

Le seguenti domande dovrebbero aiutarti a risolvere i problemi di accessibilità per gli screen reader.

Tutti i componenti e le immagini hanno alternative di testo significative?

Ovunque le informazioni sul *nome* o sullo *scopo* di un componente interattivo vengano trasmesse visivamente, fornisci un'alternativa di testo accessibile.

Ad esempio, se il componente dell'interfaccia utente **`<fancy-menu>`** mostra solo un'icona a forma di ingranaggio per indicare che si tratta di un menu delle impostazioni, è necessario un'alternativa di testo accessibile, ad esempio "Impostazioni", che trasmetta le stesse informazioni. A seconda del contesto, puoi fornire un'alternativa di testo utilizzando un attributo **`alt`**, un attributo **`aria-label`**, un attributo **`aria-labelledby`** o del testo normale nel DOM ombra. Puoi trovare suggerimenti tecnici generali nella guida di riferimento rapido di WebAIM (http://webaim.org/resources/quickref/).

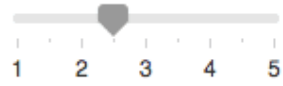
Qualsiasi componente dell'interfaccia utente che mostra un'immagine deve fornire un meccanismo per fornire un testo alternativo per l'immagine, analogo all'attributo **`alt`**.

I tuoi componenti forniscono informazioni semantiche?

Le tecnologie per la disabilità trasmettono informazioni semantiche che altrimenti vengono espresse agli utenti vedenti con indicatori visivi, come formattazione, stile del cursore o posizione. Gli elementi standardizzati hanno queste informazioni semantiche integrate nel browser, ma per i componenti personalizzati devi utilizzare ARIA (http://www.w3.org/WAI/PF/aria/) per aggiungerle.

In generale, qualsiasi componente che ascolta un evento di clic o passaggio del mouse del mouse deve avere un qualche tipo di listener di eventi della tastiera *e* un ruolo ARIA, eventualmente anche stati e attributi ARIA.

Ad esempio, un componente dell'interfaccia utente **`<fancy-slider>`** personalizzato potrebbe avere un ruolo ARIA di cursore, con alcuni attributi ARIA correlati: **`aria-valuenow`**, **`aria-valuemin`** e **`aria-valuemax`**. Se le associ alle proprietà pertinenti del componente personalizzato, puoi consentire agli utenti di tecnologie per la disabilità di interagire con l'elemento, modificarne il valore e persino modificare di conseguenza la presentazione visiva dell'elemento.

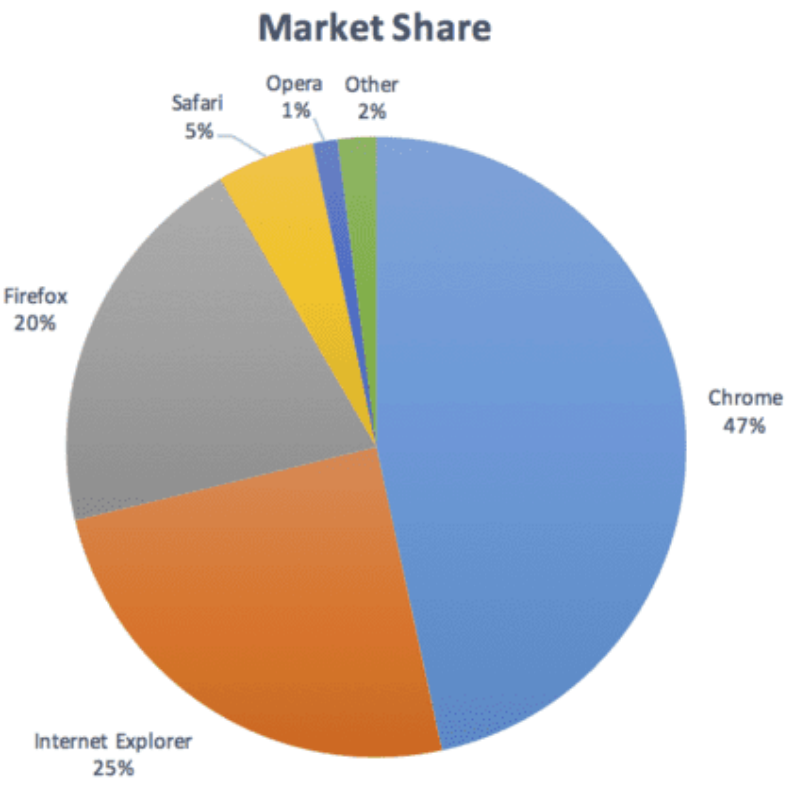


Un componente dispositivo di scorrimento con intervallo.

```
<fancy-slider role="slider" aria-valuemin="1" aria-valuemax="5" aria-valuenow="2.5
</fancy-slider>
```

Gli utenti riescono a capire tutto senza fare affidamento sul colore?

Il colore non deve essere utilizzato come unico mezzo per trasmettere informazioni, ad esempio per indicare uno stato, richiedere una risposta all'utente o visualizzare i dati. Ad esempio, se hai un grafico a torta, fornisci etichette e valori per ogni fetta in modo che gli utenti con disabilità visive possano comprendere le informazioni, anche se non riescono a vedere dove iniziano e finiscono le fette:



Un grafico a torta accessibile. (Dalla W3C Web Accessibility Initiative (https://www.w3.org/WAI/GL/low-vision-a11y-tf/wiki/Informational_Graphic_Contrast_(Minimum)).)

Il contrasto è sufficiente?

Qualsiasi contenuto di testo visualizzato nel componente deve soddisfare la soglia di contrasto minima del livello AA WCAG (http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast-contrast). Valuta la possibilità di fornire un tema ad alto contrasto che soddisfi la soglia AAA più elevata (http://www.w3.org/TR/2008/REC-WCAG20-20081211/#visual-audio-contrast7), e assicurati che sia possibile applicare gli stili dello user agent se gli utenti richiedono un contrasto personalizzato o colori diversi. Puoi utilizzare questo strumento di controllo del contrasto dei colori (http://webaim.org/resources/contrastchecker/) come aiuto per progettare il componente.

I contenuti in movimento o lampeggianti sono bloccabili e sicuri?

Gli utenti devono essere in grado di mettere in pausa, interrompere o nascondere i contenuti che si muovono, scorrono o brillano per più di cinque secondi. In generale, evita contenuti lampeggianti.

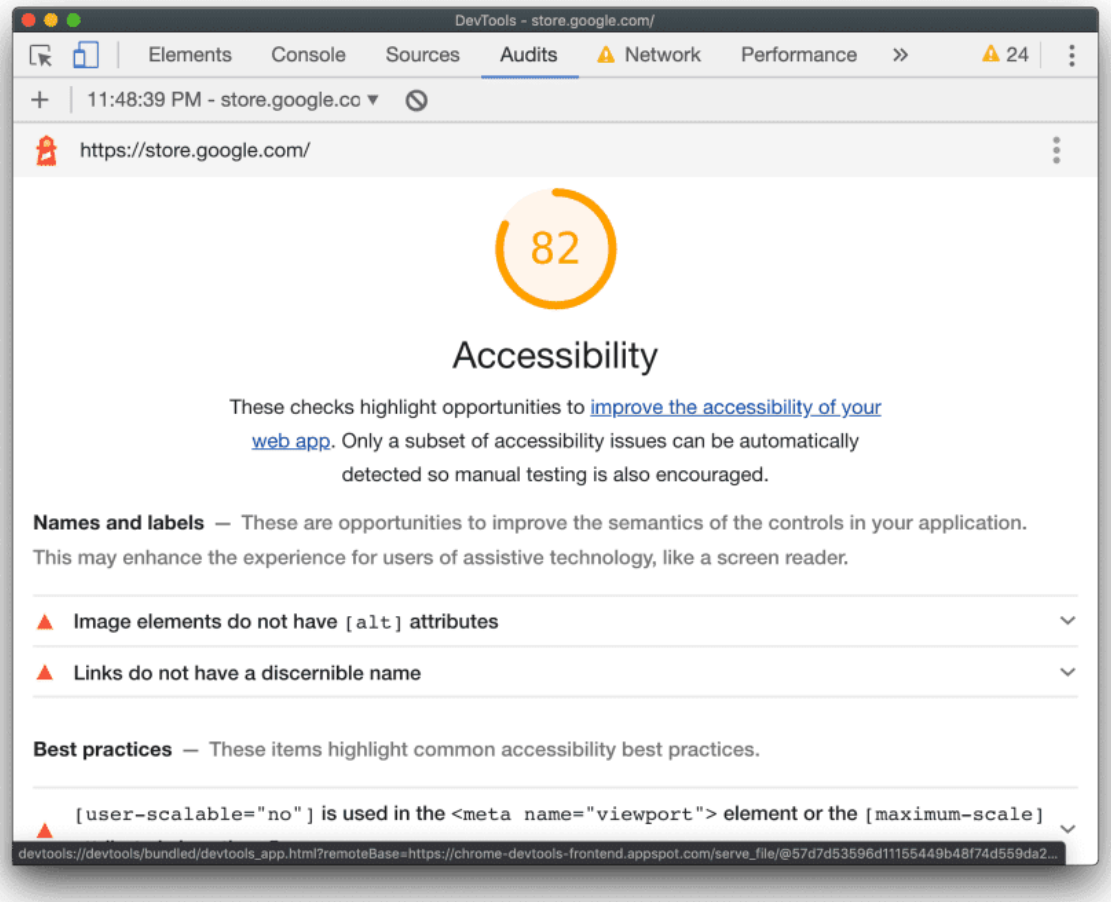
Se qualcosa deve lampeggiare, assicurati che non lo faccia più di tre volte al secondo.

Strumenti e test di accessibilità

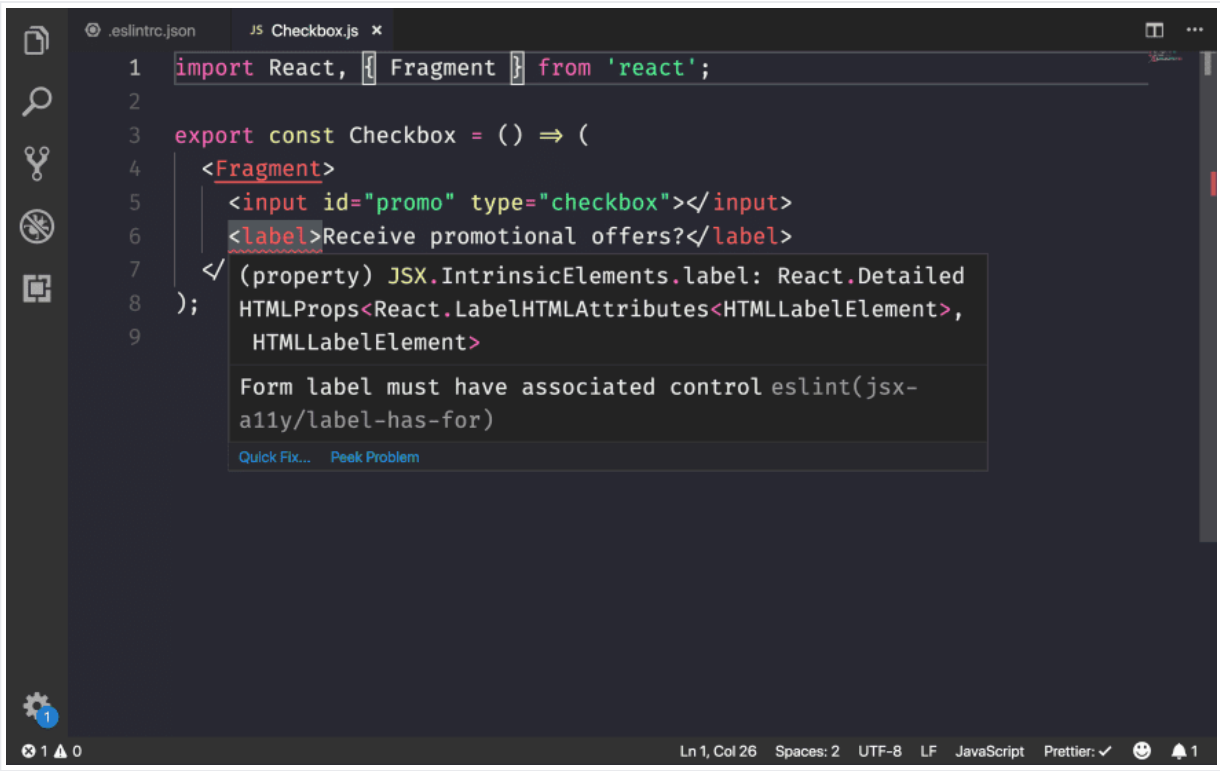
Esistono oltre 100 strumenti disponibili per valutare l'accessibilità del tuo sito (https://www.w3.org/WAI/test-evaluate/tools/list/) e dei suoi componenti. Alcuni strumenti sono automatici, mentre altri richiedono test manuali.

Eccone alcuni da prendere in considerazione:

- Axe (http://www.deque.com/products/axe/) fornisce test di accessibilità automatici per il tuo framework o browser preferito. Axe Puppeteer (https://www.deque.com/blog/axe-and-attest-integration-puppeteer/) può essere utilizzato per scrivere test di accessibilità automatici.
- Un audit di accessibilità di Lighthouse (https://developer.chrome.com/docs/lighthouse?hl=it) fornisce informazioni utili per scoprire i problemi di accessibilità più comuni. Il punteggio di accessibilità è una media ponderata di tutti i controlli di accessibilità basati sulle valutazioni dell'impatto sugli utenti di Axe (https://github.com/dequelabs/axe-core/blob/develop/doc/rule-descriptions.md). Per monitorare l'accessibilità con l'integrazione continua, consulta Lighthouse CI (https://github.com/GoogleChrome/lighthouse-ci).



- Tenon.io (https://tenon.io/) è utile per testare i problemi di accessibilità più comuni. Tenon offre un'integrazione solida tra strumenti di compilazione, browser (tramite estensioni) e persino editor di testo.
- Esistono molti strumenti specifici per librerie e framework per evidenziare i problemi di accessibilità dei componenti. Ad esempio, utilizza eslint-plugin-jsx-a11y (https://www.npmjs.com/package/eslint-plugin-jsx-a11y) per evidenziare i problemi di accessibilità per i componenti React nell'editor.

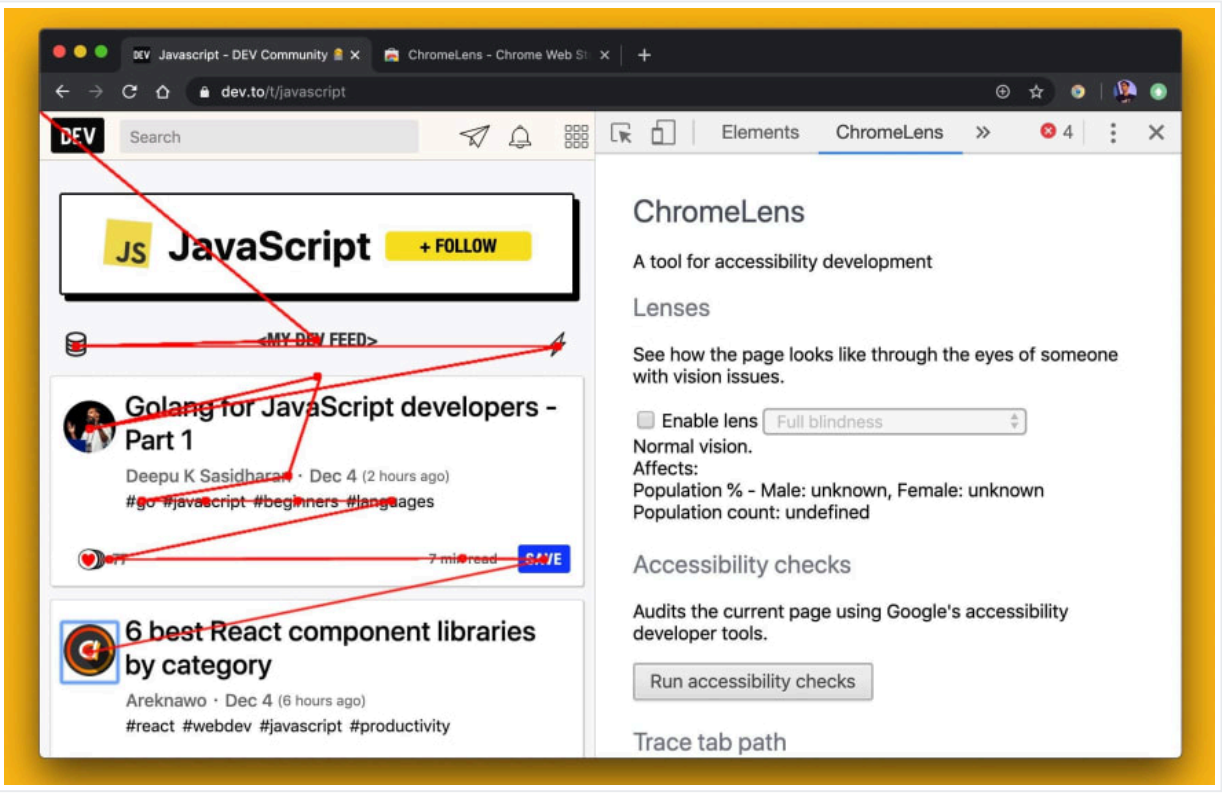


Se utilizzi Angular, codelyzer (https://web.dev/articles/accessible-angular-with-codelyzer?hl=it) fornisce anche controlli di accessibilità in-editor:

Lavorare con le tecnologie per la disabilità

Suggerimento: scopri come eseguire i test delle tecnologie per la disabilità (https://web.dev/learn/accessibility/test-assistive-technology?hl=it).

- Puoi esaminare il modo in cui le tecnologie per la disabilità vedono i contenuti web utilizzando Accessibility Inspector (https://developer.apple.com/documentation/accessibility/accessibility-inspector) (Mac), Windows Automation API Testing Tools (http://msdn.microsoft.com/en-us/library/windows/desktop/dd373661(v=vs.85).aspx) e AccProbe (http://accessibility.linuxfoundation.org/a11yweb/util/accprobe/) (Windows). Puoi anche vedere l'albero di accessibilità completo creato da Chrome andando a `about://accessibility`.
- Il modo migliore per verificare il supporto dello screen reader su un Mac è utilizzare l'utilità VoiceOver. Usa ⌘`F5` per attivarlo o disattivarlo, `Ctrl+Option` ↔ per spostarti nella pagina e `Ctrl+Shift+Option` + ↑↓ per spostarti verso l'alto e verso il basso nella struttura ad albero di accessibilità. Per istruzioni più dettagliate, consulta l'elenco completo dei comandi di VoiceOver (http://www.apple.com/voiceover/info/guide/_1131.html) e l'elenco dei comandi di VoiceOver Web (http://www.apple.com/voiceover/info/guide/_1131.html#vo27972).
- Su Windows, NVDA (http://www.nvaccess.org/) è uno screen reader senza costi e open source. Tuttavia, ha una curva di apprendimento ripida per gli utenti vedenti.



- ChromeOS ha un screen reader integrato (https://support.google.com/chromebook/answer/7031755?hl=it).

C'è ancora molto da fare per migliorare l'accessibilità sul web. Secondo l'almanacco web (https://almanac.httparchive.org/en/2019/accessibility):

- 4 siti su 5 hanno un testo che si fonde con lo sfondo, rendendoli illeggibili.

- Il 49,91% delle pagine continua a non fornire attributi `alt` per alcune delle proprie immagini.
- Solo il 24% delle pagine che utilizzano pulsanti o link include etichette.
- Solo il 22,33% delle pagine fornisce etichette per tutti gli input del modulo.

Possiamo fare molto per creare esperienze più accessibili per tutti.

Suggerimento: per scoprire di più sulle nozioni di base dell'accessibilità e contribuire a migliorare queste statistiche, consulta Accessibile a tutti (https://web.dev/explore/accessible?hl=it) e Scopri l'accessibilità (https://web.dev/learn/accessibility?hl=it).

Ultimo aggiornamento 2024-02-20 UTC.