ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Corso intensivo di accessibilità!

Vincenzo Rubano
Corso di laurea in Informatica
Alma Mater – Università di Bologna

Argomenti

- Cos’è (e perché ci interessa) l’accessibilità?
- WCAG 2.1
- Esempi in HTML
- La specifica WAI-ARIA
- ATAG 2.0

Che cos'è l'accessibilità?

- Per accessibilità si intende la possibilità che un sistema informatico (pagina web, applicazione, etc) possa essere utilizzato anche da persone con disabilità mediante tecnologie assistive. Si tratta di una pratica inclusiva di rimozione di tutte le “barriere” che impediscano l’utilizzo di un sistema mediante tali tecnologie.

Tipi di disabilità

- **Visive**: disabilità visive tra cui cecità, ipovisione e problemi nella distinzione dei colori.
- **Motorie**: difficoltà o impossibilità di usare le mani, inclusi tremori, lentezza muscolare, non completa padronanza della mobilità fine, ecc.
- **Uditive**: sordità e/o disturbi dell'udito.
- **Convulsioni**: attacchi foto-epilettici causati da effetti stroboscopici e/o lampeggianti.
- **Cognitive e intellettive**: disabilità dello sviluppo, difficoltà di apprendimento (dislessia, discalculia, ecc.) e disabilità cognitive (sindrome di Down, Alzheimer) di varie origini, che causano effetti negativi su memoria, attenzione, sviluppo, capacità logiche e di problem solving, etc.

Ma l'accessibilità è molto di più!

- Ma l'accessibilità non è vantaggiosa solo per le persone elencate nella diapositiva precedente, ma porta benefici a chiunque stia vivendo una disabilità permanente, temporanea o situazionale.
- Disabilità temporanea: un polso rotto rende impossibile il controllo del mouse.
- Per disabilità derivante dal contesto s'intende una situazione momentanea di disagio derivante da fattori esterni (ad esempio vista offuscata a causa dell'illuminazione solare, utilizzo di una sola mano quando si accompagna un bambino).

Perchè dovresti preoccupartene?

- motivi sociali ed etici;
- ragioni legali
- motivi economici
- passare questo esame 😁

Implicazioni sociali

- L'accessibilità è un diritto civile, riconosciuto dalla Convenzione delle Nazioni Unite sui diritti delle persone con disabilità (CRPD).
- Qualsiasi sito Web, applicazione o sistema non accessibile può essere considerato una discriminazione, poiché impedisce a gruppi di persone di utilizzarlo.
- I sistemi inaccessibili incidono negativamente sulla dignità, l'autonomia, la piena ed efficace partecipazione, le pari opportunità e molto altro su un alta percentuale di popolazione.

Ragioni legali

- Sono in vigore leggi e regolamenti sull'accessibilità per imporre la creazione di contenuti accessibili in tutto il mondo (Australia, Stati Uniti, Canada, Unione Europea, Italia ed altri).
- Le aziende con siti Web e / o applicazioni inaccessibili possono essere (e vengono) citate in giudizio per questo (Stati Uniti, Australia, Caada, UE coming soon, etc).
- In Italia, i siti Web e le applicazioni mobili sviluppati per conto delle pubbliche amministrazioni o delle società che forniscono servizi per loro conto devono essere accessibili. Gli enti pubblici (comprese scuole, musei, università, ecc.) non possono acquistare soluzioni IT inaccessibili.

Motivi economici

- È stato stimato che il reddito disponibile al netto delle imposte per le persone in età lavorativa con disabilità negli Stati Uniti è di circa 490 miliardi di dollari, dello stesso ordine di grandezza di altri importanti segmenti di mercato, come gli afroamericani ($ 501 miliardi) e gli ispanici ($ 582 miliardi di euro). In poche parole, proprio a causa della loro inaccessibilità, i sistemi inaccessibili perdono un'importante fetta di mercato.
- Le persone con disabilità non sono un mercato isolato; poiché sono circondati da familiari e amici che riconoscono anche il valore di prodotti e servizi che soddisfano i bisogni di tutti.
- Essere citati in giudizio per motivi legati all'inaccessibilità è oneroso e in ogni caso, oltre alle spese legali, bisognerà investire per rendere il sistema accessibile, oltre a considerare il danno d'immagine.

Passare questo esame

- Anche l'accessibilità (sia teoria che pratica) sarà un argomento d'esame.
- L'accessibilità potrebbe essere anche parte del progetto, con modalità e requisiti ancora in fase di definizione.
- Nel mese di giugno faremo un esperimento di progettazione di siti accessibili, e cerchiamo volontari.
- Inoltre sono disponibili tesi di laurea su questo argomento per gli studenti interessati ad approfondirli

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

L'accessibilità nel World Wide Web

Verso linee guida ufficiali

Al fine di garantire che un sistema possa essere utilizzato da chiunque, indipendentemente dalle tecnologie assistive di cui abbia bisogno, è necessario introdurre linee guida più complesse.

Data la varietà di tecnologie disponibili al giorno d'oggi per implementare nuovi sistemi, tali linee guida dovrebbero essere definite in modo sufficientemente astratto da essere valide per ciascuna di esse, ma comunque abbastanza concreto da rendere possibile l'implementazione effettiva di tali raccomandazioni.

WCAG 2.1

Web Content Accessibility Guidelines (WCAG) 2.1 è una raccomandazione del W3C che contiene una serie di linee guida che ogni sistema deve soddisfare per essere considerato accessibile.

Tali linee guida sono organizzate attorno a 4 principi fondamentali. Per ciascuna linea guida, vengono forniti i cosiddetti success criteria (dichiarazioni verificabili), che specificano cosa testare e i risultati attesi, in modo indipendente da una specifica tecnologia.

La conformità a WCAG 2.1 può essere classificata in tre diversi livelli (A, AA, AAA) a seconda dei criteri di successo che il sistema soddisfa.

Si noti che vengono forniti documenti di supporto aggiuntivi ("techniques for WCAG 2.1") per offrire esempi pratici su come soddisfare i criteri di successo utilizzando specifiche tecnologie (implementazioni concrete).

WCAG 2.1 principles

The guidelines and Success Criteria are organized around the following four principles, which lay the foundation necessary for anyone to access and use Web content. Anyone who wants to use the Web must have content that is:

- ***Perceivable***. Information and user interface components must be presentable to users in ways they can perceive. This means that users must be able to perceive the information being presented (it can't be invisible to all of their senses)
- ***Operable***. User interface components and navigation must be operable. This means that users must be able to operate the interface (the interface cannot require interaction that a user cannot perform).

WCAG 2.1 principles II

Anyone who wants to use the Web must have content that is:

- ***Understandable***. Information and the operation of user interface must be understandable. This means that users must be able to understand the information as well as the operation of the user interface (the content or operation cannot be beyond their understanding).
- ***Robust***. Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies. This means that users must be able to access the content as technologies advance (as technologies and user agents evolve, the content should remain accessible).

- [Source: Introduction to understanding WCAG 2.1]

Esempi pratici (1/2)

- La codifica di un sito utilizzando HTML semanticamente valido, fornendo rappresentazioni testuali equivalenti per immagini e di testi dei link significativi aiutano gli utenti non vedenti che utilizzano screen reader e display Braille.
- Testo e immagini di grandi dimensioni e / o ingrandibili semplificano la lettura e la comprensione del contenuto da parte degli utenti con problemi di vista.
- Utilizzare link colorati e sottolineati o altrimenti differenziati assicura che gli utenti daltonici possano notarli.
- Link e aree cliccabili di dimensioni più ampie aiutano gli utenti che non possono controllare con precisione un mouse (o utilizzano il sito Web con un dispositivo touchscreen).

Esempi pratici (2/2)

- Scrivere codice che non ostacoli la navigazione mediante la sola tastiera o il controllo interruttori consente agli utenti che non possono utilizzare un mouse di poter usufruire comunque della pagina Web.
- Fornendo i sottotitoli, una trascrizione e / o la versione in lingua dei segni del sonoro di un video, gli utenti non udenti e ipoudenti potranno comprenderne i contenuti.
- Quando gli effetti stroboscopici e lampeggianti vengono evitati o resi opzionali, si diminuiscono i rischi per gli utenti che soffrono di convulsioni e/o attacchi foto-epilettici causate da questi effetti.
- Scrivendo contenuti didattici in un linguaggio semplice e illustrandoli con diagrammi e animazioni, essi risulteranno maggiormente comprensibili per gli utenti con difficoltà di apprendimento.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Un po' di esempi in HTML

Lingua del documento

- In HTML, l'attributo «lang» specifica la lingua in cui è scritto il documento. Specificare un valore corretto per tale attributo è importante per l'accessibilità: gli screen reader, per esempio. Possono usare l'informazione per attivare dizionari di pronunce specifiche per quella lingua.

- L'attributo lang ammette come valori un «language tag» secondo lo standard BCP47 («en» per inglese, «it» per italiano, ad esempio)), o la stringa vuota; questo secondo caso è da evitare! Può essere usato su ogni elemento HTML; se un elemento non lo specifica, la lingua usata è quella dell'elemento padre (se esiste).

Esempio

L'esempio mostra una pagina web in italiano che contiene al suo interno una citazione in inglese.

```
<!doctype html>
<html lang="it">
  [...]
  <p>William Faulkner ha detto:</p>
  <blockquote lang="en">
    The past is never dead. It's not even past.
  </blockquote>
  [...]
</html>
```

Descrivere le immagini

- Ad ogni immagine «non decorativa» (ovvero ogni immagine che non svolga esclusivamente funzioni estetiche nella pagina), deve essere associata una breve descrizione che consenta di comprenderne a pieno il contenuto anche senza vederla. Oltre che dai motori di ricerca, questa informazione è indispensabile per rendere la pagina accessibile: gli screen reader, per esempio, la utilizzano per permettere alle persone non vedenti di comprendere cosa è mostrato nell'immagine.

Descrivere le immagini in HTML

- Esistono vari modi, ma si consiglia l'utilizzo dell'attributo «alt» al cui interno si può inserire una stringa di testo.

```
<img src="kitten.jpg" alt="un gattino che insegue un gomitolo">
```

- Scrivere una «descrizione alternativa» per un'immagine non è semplice!

Immagini decorative

- Se l'immagine è decorativa e viene caricata mediante il tag «img», allora occorre specificare un attributo «alt» il cui valore sia la stringa vuota;
- Questo indica alle tecnologie assistive che quell'immagine va trattata, appunto, come immagine decorativa.
- Non specificare alcun valore per l'attributo alt è sempre un errore!

Sì: `<img src="decorazione.jpg" alt="">`

No: `<img src="decorazione.jpg">`

Intestazioni

- Per rendere una pagina Web accessibile le intestazioni sono fondamentali! In HTML ne esistono 6 livelli (ordinati gerarchicamente dal più importante al meno importante), che sono rappresentati dai tag «h1», «h2», «h3», «h4», «h5», e «h6».

  Le intestazioni vanno annidate gerarchicamente in modo corretto: è un errore, per esempio, utilizzare «h3» se in precedenza non è stato utilizzato il tag «h2» per l'intestazione di livello superiore.

Esempio corretto di annidamento delle intestazioni

```
[…]
<h1>1. Lezione di tecnologie Web sull'accessibilità</h1>
<p>Testo, anche più di un paragrafo.</p>
<h2>1.1. Introduzione</h2>
<p>Testo, anche più paragrafi</p>
<h2>1.2. Argomenti</h2>
<p>Testo opzionale, anche più paragrafi.</p>
<h3>1.2.1. Lingua del documento</h3>
<p>Testo, anche più paragrafi</p>
<h3>1.2.2. Descrivere le immagini</h3>
<p>Testo, anche più paragrafi</p>
<h2>1.3. I form</h3>
<p>Testo, anche più paragrafi</p>
[…]
```

Regole empiriche per le intestazioni

- Immaginare di dover attribuire delle numerazioni gerarchiche rende intuitivo capire quali livelli di intestazione usare (ed evitare di saltarne qualcuno!).
- Se l'aspetto grafico del livello di intestazione corretto non è adatto possiamo modificarlo tramite CSS.
- Ogni pagina deve sempre contenere al massimo un livello di intestazione 1 (il titolo della pagina!).
- Se un frammento di testo viene rappresentato come un sottotitolo (centrato, con colore diverso, font più grande, etc.) per metterlo in evidenza dal resto del testo, probabilmente usare un livello di intestazione può essere una buona idea.

Tabelle

- Non tutte le celle di una tabella sono uguali. Alcune di esse, infatti, fungono da intestazioni per righe e colonne; in un certo senso, queste celle «descrivono» la funzione dei dati a cui si riferiscono.

- Non è sufficiente modificarne l'aspetto grafico in modo da metterle in risalto: affinché le tabelle siano accessibili queste relazioni devono essere «esposte» semanticamente usando i tag appropriati.

Esempio di tabella

```
[…]
<table>
  <caption>Orari di apertura dei locali</caption>
  <thead>
    <tr>
      <th>Locale</th><th>Lunedì</th><th>Martedì</th>
<th>Mercoledì</th><th>Giovedì</th><th>Venerdì</th><th>Sabato</th><th>Domen
ica</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Pizza pazza</th><td>Chiuso</td><td>12:30 - 23:30</td><td>12:30 -
23:30</td><td>12:30 - 23:30</td><td>12:30 -
23:30</td><td>Chiuso</td><td>Chiuso</td>    </tr>
    <tr>
        <td>La bistecca</td><td>Chiuso</td><td>11:30 - 15:30, 19:30 -
22:30</td><td>11:30 - 15:30, 19:30 - 22:30</td><td>11:30 - 15:30, 19:30 -
22:30</td><td>11:30 - 15:30, 19:30 - 22:30</td><td>11:30 - 15:30, 19:30 -
22:30</td><td>12:00 - 15:00, 19:00 - 23:00</td>
    </tr>
  </tbody>
</table>
[…]
```

Etichettare i campi di un form

- Una stringa di testo situata immediatamente a sinistra (o subito sopra) di un campo di un form è la sua etichetta.

- Questa relazione è ovvia, ma affinché la pagina sia accessibile deve essere «esposta» anche semanticamente con appositi tag e attributi HTML.

Esempio

```
[…]
<form>
    <label>Nome:
      <input type="text">
    </label>
    <label for="lastname">Cognome:</label>
    <input id="lastname" type="text" required>
    <label for="genere">Genere:</label>
    <select name="genere" id="genere">
        <option value="NA">-- Scegliere un opzione --</option>
        <option value="m">Maschile</option>
        <option value="f">Femminile</option>
        <option value="other">Altro, o preferisco non dirlo</option>
    </select>
    <input type="text" name="subject" title="Oggetto del messaggio"
           placeholder="Inserisci l'oggetto del messaggio">
    <label for="txt">Testo del messaggio:</label>
    <textarea id="text" cols="30" rows="10" required></textarea>
    <input type="checkbox" id="check">
    <label for="check">Accetto i termini di servizio</label>
    <button type="submit">Invia modulo!</button>
    <input type="reset" value="Pulisci i campi">
</form>
[…]
```

Osservazioni sui campi dei form

- Esistono più modi per associare un etichetta ad un campo;
- Ogni etichetta deve essere associata ad un campo; ogni campo deve essere associato ad un'etichetta;
- Per indicare che la compilazione di un campo sia obbligatoria possiamo usare l'attributo «required»;
- I pulsanti (tag button) non richiedono etichetta; le tecnologie assistive usano il testo al loro interno (o l'attributo alt di una eventuale immagine);
- Alcuni tag input (di tipo «submit» o «reset», per esempio) usano come accessibility name il valore dell'attributo «value».

Fieldset

- E’ possibile esprimere semanticamente un «raggruppamento» tra campi di un form, per esempio se essi sono riconducibili allo stesso macroargomento (il gruppo «dati personali» potrebbe comprendere i campi per nome, cognome e data di nascita, per esempio).

- Il loro utilizzo è opzionale, almeno che il form non sia molto complesso (perché ne facilitano la navigazione), o si debbano usare i radio button!

Esempio di radiogroup

```
[…]
    <fieldset>
      <legend>Cottura della bistecca:</legend>
      <input id="radio1" type="radio">
      <label for="radio1">Al sangue</label>
      <input id="radio2" type="radio">
      <label for="radio2">Cottura media</label>
      <input id="radio1" type="radio">
      <label for="radio3">Ben cotta</label>
    </fieldset>
[…]
```

Struttura della pagina

- In HTML 5 sono stati introdotti diversi tag che permettono di attribuire un valore semantico a diverse «aree» della pagina Web. Se usate correttamente, queste «annotazioni» possono contribuire a rendere la pagina più accessibile.

- In gergo a volte si usa il termine «landmark», giacché essi possono costituire dei punti di riferimento quando si naviga la pagina mediante tecnologie assistive.

Esempio di struttura

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="utf-8">
 <title>native landmarks examples</title>
</head>

<body>
 <header class="header">
  <img src="../common/desert-256.png" alt="Saharian logo">
  <p>Welcome to Saharian's official website, a place where you can find a lot of loremipsum to test accessibility landmarks!
  </p>
 </header>
 <nav class="nav">
  <h2>Fictional menu</h2>
  <ul>
   <li>
    <a href="./index.html" class="active" aria-current="page">Home</a>
   </li>
   <li>
    <a href="about-not-exist.html">About Saharian (page does not exist)</a>
   </li>
   <li>
    <a href="/about-us/">About us (page does not exist)</a>
   </li>
  </ul>
 </nav>
 <div class="search" role="search" aria-labelledby="searchblocktitle">
  <h2 id="searchblocktitle">Search this website</h2>
  <form action="" method="get">
   <label for="query">Search for:</label>
   <input type="search" id="query" name="q" aria-describedby="searchblockhelp">
   <p id="searchblockhelp">Enter the terms you wish to search for... Just be ware that this is a fictional web page
    to test landmark roles, therefore searchig does not work at all.</p>
   <input type="submit" value="Search">
  </form>
 </div>

 <main class="main">
  <h1>HomePage</h1>
  <p>Hello! Welcome to the HomePage of Saharian, a pretty useless website if you exclude the fact that this page can
   be used to test different aria landmarks.</p>
  <p>In deed, this has been created for the sole purpose of containing as many landmarks as the specification can
   provide, really. Not for any other purpose.</p>

  <h2>Some facts</h2>
  <p>Here we could put anything. For example:</p>
  <ul>
   <li>a list</li>
   <li>of non sense elements.</li>
  </ul>
```

Sugli esempi visti finora

- Gli esempi visti finora sono particolarmente semplici, ma rendono l'idea del fatto che le raccomandazioni teoriche delle linee guida sull'accessibilità hanno riscontri estremamente pratici;
- In rete esistono svariate risorse estremamente utili per comprendere come rendere accessibile qualsiasi elemento di una pagina Web, la più importante delle quali è sicuramente «Techniques for WCAG 2.1».

Techniques for WCAG 2.1

E’ una raccolta di tutorial, esempi e guide che illustrano come soddisfare le WCAG 2.1 creando interfacce e contenuti accessibili con specifici linguaggi (HTML, CSS, JavaScript, ETC.). Si dividono in vari gruppi, di cui i più importanti sono:

- *client-side script techniques*, che spiegano come rispettare le WCAG 2.1 ;
- *CSS Techniques*, che illustrano come rispettare le WCAG 2.1 quando si usano I fogli di stile CSS;
- *common failures*, che illustrano errori comuni e forniscono le alternative “corrette”;
- *general techniques*, che illustrano come rendere accessibile un’applicazione in fase di design;
- *HTML techniques*, che illustrano aspetti specifici relativi a HTML/XHTML;
- *PDF techniques*, che spiegano come “rendere accessibili” documenti PDF (complicato!);
- *Server-Side script techniques*, che spiegano come migliorare l’accessibilità mediante I linguaggi server-side.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Creare un sistema accessibile

Un processo in più passaggi

Per definizione, è chiaro che nello sviluppo di un sistema accessibile sono coinvolte più fasi del suo ciclo di vita:

- **Design**: decisioni importanti per il progetto devono essere prese ancor prima di scrivere la prima riga di codice; è molto più semplice rendere accessibile un'interfaccia facile da usare piuttosto che una complessa.
- **Sviluppo**: se non effettuata correttamente,l ’implementazione del design può introdurre problemi di accessibilità.
- **Creazione dei contenuti**: anche le informazioni inserite nel sistema devono essere accessibili, altrimenti tutti gli sforzi fatti nelle fasi precedenti sono vani!

Design

Ogni decisione di progettazione ha il potenziale per includere o escludere dei clienti. Il design inclusivo è una filosofia di design che massimizza il contributo fornito dalla comprensione della diversità degli utenti per prendere tali decisioni, consentendo così di eliminare gli ostacoli che ne impediscono l'utilizzo per quante più persone possibili.

La diversità degli utenti riguarda le differenti capacità, esigenze, aspirazioni, etc.

Esempio

- Un’interfaccia viene concepita per supportare solo ed esclusivamente l’interazione mediante il drag-and-drop degli elementi con il mouse.

- SBAGLIATO: occorre progettare anche «esperienze utente» alternative che prevedano l’utilizzo di comandi da tastiera (o meccanismi sostitutivi per l’interazione).

Sviluppo

- Anche la fase di implementazione può introdurre ostacoli. Il progetto deve essere implementato sfruttando le tecnologie esistenti note per essere accessibili o adottando tutti i meccanismi necessari per renderle tali.

- Sono disponibili documenti di supporto e standard tecnici che spiegano come rispettare WCAG 2.1 in scenari specifici, ad es. Tecniche per WCAG 2.1

Contenuti

- Anche gli sforzi fatti per creare il sistema più accessibile svaniscono quando i suoi contenuti non vengono creati per essere accessibili. Gli esempi includono:
  - allegare documenti inaccessibili (file PDF scansionati senza OCR),
  - screenshot privi di descrizioni alternative che consentano di comprenderne il contenuto,
  - scrivere testi che non possono essere compresi da tutti (ad es. Usare informazioni che possono essere correlate solo a un senso),
  - non fornire abbastanza contesto per la comprensione del contenuto in caso di disabilità.

Livelli di conformità WCAG 2.1

A:

- livello più basso di conformità;
- rimuove le principali barriere per cecità, sordità e disabilità motorie.

AA:

- livello successivo di conformità (include A);
- rimuove le principali barriere per gli utenti ipovedenti;
- offre un piccolo aiuto per le disabilità cognitive.

AAA:

- massimo livello di conformità (include A e AA);
- non è consigliabile che sia richiesto come politica generale per interi siti perché non è possibile soddisfare tutti i requisiti di livello AAA per alcuni contenuti.

Come testiamo la conformità?

- Le verifiche di conformità alle linee guida sull'accessibilità possono essere automatizzate solo in parte, perciò è necessario comunque eseguire test manuali per valutarla in modo esaustivo ed attendibile.

- Consideriamo un semplice criterio di successo: tutte le immagini non decorative dovrebbero avere un testo alternativo descrittivo. Controllare che a un'immagine sia associato un testo alternativo è banale, ma assicurarsi che sia descrittivo per quell'immagine non lo è (ancora). Anche distinguere quali immagini sono decorative e quali no può essere complicato, (lo è per l'uomo, figurarsi per un sistema automatizzato).

Una nota sugli strumenti di test automatizzati dell'accessibilità

Si noti che i primi strumenti di test per l’accessibilità automatizzati di prima generazione considerano solo il markup originale della pagina Web, quindi non possono rilevare tonnellate di errori e non sono adatti per le applicazioni AJAX.

D'altro canto, gli strumenti più recenti testano il DOM effettivo di una pagina Web, offrendo così risultati più accurati.

Miti e leggende I

Mito: un sito Web accessibile è brutto e noioso.

Fatto: è possibile implementare siti Web sofisticati e ben realizzati, ma accessibili! Un design accessibile è più “usabile”, ma questa è un’altra storia!

Mito: l'accessibilità è costosa!

Fatto: sì, ma solo se viene considerata a posteriori. La riparazione di progetti inaccessibili richiede molti più sforzi, tempo e conoscenza (quindi denaro) rispetto alla creazione del suo equivalente accessibile. In ogni caso, il risultato finale potrebbe non essere così buono come sarebbe potuto essere considerando l’accessibilità fin dal principio.

Miti e leggende II

Mito: l'accessibilità avvantaggia troppo poche persone.

Fatto: si stima che circa il 10% della popolazione mondiale abbia una disabilità che influisce sull'uso di Internet. Circa 700 milioni di persone sono troppo poche? A queste occorre aggiungere le persone colpite da disabilità temporanee e derivanti dal contesto! E piaccia o no, con l'età diminuiscono l'udito, la vista e la destrezza, cambiando la nostra capacità di usare Internet.

Mito: i siti Web accessibili sono statici.

Fatto: siti Web altamente dinamici e sofisticati (anche applicazioni desktop-like) possono essere resi accessibili, è necessaria solo maggiore attenzione. Benvenuti in WAI-ARIA!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

WAI-ARIA

La specifica WAI-ARIA

- Accessible Rich Internet Applications (WAI-ARIA) 1.1 è una raccomandazione del W3C che fornisce un'ontologia di ruoli, stati e proprietà che definiscono gli elementi dell'interfaccia utente e possono essere utilizzati per migliorare l'accessibilità e l'interoperabilità dei contenuti e delle applicazioni Web. Progettato per consentire a un autore di comunicare in modo appropriato comportamenti dell'interfaccia utente e informazioni strutturali alle tecnologie assistive attraverso il markup.

- È uno strumento fondamentale per rendere accessibili le applicazioni Web desktop-like, in quanto vi sono (molti) widget avanzati (barre dei menu, visualizzazioni ad albero, barre degli strumenti, ecc.) ancora non disponibili in HTML.

Ruoli, proprietà e stati

È possibile usare WAI-ARIA sfruttando attributi specifici da applicare su qualsiasi elemento HTML:

- l'attributo **role**, che specifica il ruolo (semantica) dell'elemento (button, checkbox, tree, tablist, tab, ecc);
- proprietà (**aria-label, aria-labelledby, aria-valuenow**, ecc.), attributi essenziali per la natura di un dato oggetto o che rappresentano un valore di dati associati ad esso. Una modifica di una proprietà può avere un impatto rilevante sul significato o sulla presentazione di un oggetto;
- stati (**aria-checked, aria-selected**, ecc.), proprietà dinamiche che esprimono le caratteristiche di un oggetto che possono cambiare in risposta ad azioni intraprese dall'utente o a processi automatizzati. Gli stati non influenzano la natura essenziale dell'oggetto, ma rappresentano i dati associati all'oggetto o alle possibilità di interazione dell'utente con esso.

Tipi di ruoli

I ruoli facenti parte della specifica WAI-ARIA possono essere raggruppati in:

- **landmark**, che consentono di «annotare» punti di riferimento nella pagina. Spesso esistono tag HTML equivalenti;
- **widget** per cui esistono analoghi elementi in HTML: è questo il caso di «checbox», «radio», radiogroup» o «table», per esempio;
- **semantica di widget non rappresentati da alcun elemento in HTML**; è questo per esempio il caso di «tab», «tablist», «tabpanel», «tree», treeitem», etc. In questo caso la specifica WAI-ARIA diventa l'unico modo disponibile per creare una rappresentazione accessibile di quel widget.

Regole di ARIA

- Non usare ARIA a meno che non sia necessario; se esiste un elemento HTML o un attributo nativo con la semantica ed il comportamento necessari, usa quello. Eccezioni:
  - se la funzionalità è disponibile in HTML ma non è implementata o la sua implementazione non fornisce supporto di accessibilità;
  - Se i vincoli di progettazione visiva escludono l'uso di un particolare elemento nativo, in quanto non può essere “stilizzato” come necessario.
- Non modificare la semantica nativa, a meno che non sia necessario. Si noti che se un elemento non interattivo (ad es. span) viene utilizzato come interattivo (ad es. pulsante), è compito dello sviluppatore implementare il comportamento appropriato utilizzando JavaScript.
- tutti i controlli interattivi ARIA devono essere utilizzabili con la tastiera. Il supporto deve essere implementato dallo sviluppatore.
- Non utilizzare role = "presentation" o aria-hidden = "true" su un elemento focalizzabile, altrimenti il focus potrebbe essere posizionato in punti su cui non potrebbe esserlo, causando effetti non determinati.
- Tutti gli elementi interattivi devono avere un accessible name.

Un ruolo è una promessa

Quando si usa un ruolo della specifica ARIA si sta «facendo una promessa» agli utenti di tecnologie assistive, dichiarando che quel'elemento si comporta secondo la semantica di ciò che rappresenta il ruolo che gli è stato attribuito. Tali comportamenti, però, devono essere implementati dal programmatore (spesso tramite JavaScript), seguendo la documentazione disponibile (e.g. WAI-ARIA Authoring Practices).

Miglioriamo un form con ARIA

- Gli attributi della specifica WAI-ARIA possono essere utilizzati insieme ad elementi HTML nativi per migliorarne l'accessibilità. Riprendiamo il form visto in precedenza, ed associamo ad alcuni campi anche un testo di aiuto che spieghi all'utente come compilarli.

Campi con descrizioni

```
[…]
<form>
    <label>Nome:
      <input type="text">
    </label>
    <label for="lastname">Cognome:</label>
    <input id="lastname" type="text" required
           aria-describedby="lastname-help">
    <p id="lastname-help">Questa è la descrizione aggiuntiva di un campo
       obbligatorio, da utilizzare per un testo d'aiuto all'utente.</p>
    <label for="genere">Genere:</label>
    <select name="genere" id="genere">
        <option value="NA">-- Scegliere un opzione --</option>
        <option value="m">Maschile</option>
        <option value="f">Femminile</option>
        <option value="other">Altro, o preferisco non dirlo</option>
    </select>
    <input type="text" name="subject" title="Oggetto del messaggio"
           placeholder="Inserisci l'oggetto del messaggio">
    <label for="txt">Testo del messaggio:</label>
    <textarea id="text" cols="30" rows="10" required></textarea>
    <input type="checkbox" id="check">
    <label for="check">Accetto i termini di servizio</label>
    <button type="submit">Invia modulo!</button>
    <input type="reset" value="Pulisci i campi">
</form>
[…]
```

Le live region

- Sono un aspetto particolarmente importante della specifica WAI-ARIA. Si tratta di un meccanismo che consente di «informare» le tecnologie assistive (specialmente screen readers) del fatto che alcune porzioni della pagina cambiano dinamicamente, e che gli aggiornamenti devono essere «comunicati» all'utente prima possibile.

- Si pensi per esempio ad un timer: il testo che indica il tempo residuo deve essere comunicato costantemente affinché il timer sia accessibile!

Attributi per le live region

- Si usano gli attributi aria-live (che può assumere valori «assertive», «polite» oppure «off») e «aria-atomic» (che può assumere valore «true» o «false») su un qualsiasi elemento HTML con role «region».

- In alternativa esistono diversi ruoli appositamente predisposti per scopi specifici: «status», «timer», «log».

- Se mediante JavaScript viene aggiornato il DOM di una live region, le tecnologie assistive comunicheranno l'aggiornamento in modo appropriato secondo gli attributi usati.

Landmark con WAI-ARIA I

- Come detto, può capitare che alcuni attributi della specifica WAI-ARIA siano del tutto equivalenti ad altri elementi e attributi già presenti in HTML. Vediamo la stessa struttura per una pagina Web analizzata in precedenza, ma stavolta implementata solo con attributi della specifica WAI-ARIA.

Landmark con WAI-ARIA II

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>native landmarks examples</title>
</head>

<body>
  <div role="banner" class="header">
    <img src="../common/desert-256.png" alt="Saharian logo">
    <p>Welcome to Saharian's official website, a place where you can find a loto fo loremipusm to test aria landmarks!
    </p>
  </div>
  <div role="navigation" class="nav">
    <h2>Fictional menu</h2>
    <ul>
      <li>
        <a href="./index.html" class="active" aria-current="page">Home</a>
      </li>
      <li>
        <a href="about-not-exist.html">About Saharian (page does not exist)</a>
      </li>
      <li>
        <a href="/about-us/">About us (page does not exist)</a>
      </li>
    </ul>
    </nav>
    <div class="search" role="search" aria-labelledby="searchblocktitle">
      <h2 id="searchblocktitle">Search this website</h2>
      <form action="" method="get">
        <label for="query">Search for:</label>
        <input type="search" id="query" name="q" aria-describedby="searchblockhelp">
        <p id="searchblockhelp">Enter the terms you wish to search for... Just be ware that this is a fictional web page
          to test landmark roles, therefore searchig does not work at all.</p>
        <input type="submit" value="Search">
      </form>
    </div>

    <div role="main" class="main">
      <h1>HomePage</h1>
      <p>Hello! Welcome to the HomePage of Saharian, a pretty useless website if you exclude the fact that this page can
        be used to test different aria landmarks.</p>
      <p>In deed, this has been created for the sole purpose of containing as many landmarks as the specification can
        provide, really. Not for any other purpose.</p>

      <h2>Some facts</h2>
      <p>Here we could put anything. For example:</p>
      <ul>
        <li>a list</li>
        <li>of non sense elements.</li>
      </ul>

      <p>And why not wrapping <code>a simple sentence</code> as source code? <em>Don't do that!</em>, let me emphasize
        that. <i>Don't do that!</i>, maybe italics works better for you.</p>
```

Un ruolo è una promessa…

- …E ogni promessa è debito. Ma cosa vuol dire? Vediamolo con un esempio pratico: implementiamo un radiogroup usando la specifica WAI-ARIA anziché gli elementi nativi forniti da HTML. Questo dovrebbe convincervi del fatto che conviene usare la specifica ARIA solo quando (e se) è strettamente indispensabile!

Leggiamo la specifica di radiogroup

Ci viene in aiuto il documento «WAI-ARIA 1.1 authoring practices», che illustra nel dettaglio cosa un utente di tecnologie assistive si aspetti da ogni implementazione di questo design pattern. Per brevità concentriamoci sui comandi da tastiera:

- Pressing «tab» moves focus to the checked radio button in the radiogroup. If a radio button is not checked, focus moves to the first radio button in the group.
- If the radio button with focus is not checked, pressing “spacebar” changes the state to checked.
- Pressing “Down or Right arrow” moves focus to and checks the next radio button in the group. If focus is on the last radio button, moves focus to the first radio button. Always changes the sate of the previously checked radio button to “unchecked”.
- Pressing “Up or Left arrow” does the opposite.

Markup di Radiogroup con aria

```
[…]
<div id="rg1" class="radiogroup" role="radiogroup" aria-
labelledby="radiogroup1-title">
  <h3 id="radiogroup1-title">Steak doneness:</h3>
  <span class="radio" role="radio" aria-checked="true"
        tabindex="0">rare</span>
  <span class="radio" role="radio" aria-checked="false"
        tabindex="-1">medium-rare</span>
  <span class="radio" role="radio" aria-checked="false"
        tabindex="-1">medium</span>
  <span class="radio" role="radio" aria-checked="false"
        tabindex="-1">medium-well</span>
  <span class="radio" role="radio" aria-checked="false"
        tabindex="-1">well-done</span>
</div>
[…]
```

Ma cos’è tabindex?

- Quando si naviga una pagina Web mediante la tastiera esiste il cosiddetto «tabbing order», ovvero l’ordine in cui viene posizionato il focus sugli elementi quando si preme «tab» (o «shift+tab» per l’ordine inverso).

- Tale ordine viene determinato da vari fattori tra cui la natura degli elementi, la loro posizione nel dom, nonché il valore dell’eventuale attributo “tabindex”.

Valori di tabindex

- L'attributo tabindex può avere tre valori:
  - «-1», che indica che l'elemento può ricevere il focus esclusivamente mediante apposite istruzioni JavaScript; i comandi da tastiera ignorano l'elemento (in gergo, è «fuori dal tabbing order»);
  - Valore «0», il valore di default: l'elemento può ricevere il focus sia mediante i comandi da tastiera che attraverso istruzioni JavaScript. La sua posizione nel «tabbing order» deriva dalla sua posizione nel DOM della pagina;
  - Valore positivo: l'elemento può ricevere il focus sia mediante comandi da tastiera che tramite istruzioni JavaScript. La sua posizione nel «tabbing order» è influenzata dal valore di tabindex;
- Si noti che tabindex influenza la propagazione di eventi JavaScript relativi a comandi da tastiera (e.g. 'keydown') verso l'elemento padre;
- Di fatto, a causa di differenze implementative nei browser è sconsigliatissimo utilizzare valori di tabindex positivi maggiori di 0.

Perché usiamo tabindex?

La corretta implementazione di molti design pattern che sfruttano la specifica WAI-ARIA necessita la scrittura di codice JavaScript per gestire appositi comandi da tastiera (che sono specifici per ogni widget) in modo adeguato. Sfruttare le proprietà di «tabindex» è la chiave per implementarli attraverso due tecniche:

- «rowing tabindex», in cui il «padre» gestisce il tabindex di tutti i suoi figli; in ogni momento, solo uno di essi avrà tabindex uguale a 0.
- «aria-activedescendant»: sfruttando un particolare attributo della specifica WAI-ARIA, solo il «padre» avrà tabindex uguale a 0, mentre i figli avranno sempre «tabindex» negativo; l'attributo «aria-activedescendant» conterrà sempre un ID univoco che si riferisce al figlio «selezionato», qualunque cosa significhi per il widget da implementare (radiogroup, menubar, tablist, ETC. hanno definizioni diverse di «selezione» che derivano dalla loro semantica).

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ATAG

ATAG 2.0

Le linee guida per l'accessibilità degli strumenti di authoring (ATAG) 2.0 sono una raccomandazione del W3C creata appositamente per garantire l'accessibilità degli strumenti di authoring come:

- strumenti per la creazione di pagine Web (ovvero editor HTML WYSIWYG);
- software per la generazione di siti Web (ovvero sistemi CMS);
- software che convertono i contenuti in tecnologie web;
- strumenti di authoring multimediale;
- siti Web i cui utenti possono aggiungere contenuti (ad es. social network).

ATAG 2.0 II

- La specifica è divisa in due parti principali:
  - parte a, che consiste nel rendere accessibili gli strumenti di creazione in modo che le persone con disabilità possano utilizzarli;
  - parte b, che consiste nell'aiutare gli autori a produrre contenuti accessibili, ovvero contenuti conformi a WCAG 2.1.

- Come in WCAG 2.1, in ATAG troviamo linee guida organizzate attorno a principi chiave, la cui soddisfazione può essere valutata sulla base della conformità ai criteri di successo sugli stessi 3 livelli (A, AA, AAA).

ATAG 2.0 principles

Part A principles:

A1. The authoring tool user interface follows applicable accessibility guidelines.

A2. Editing-views are perceivable.

A3. Editing-views are operable

A4. Editing-views are understandable

Part B principles:

B1. Fully automatic processes produce accessible content.

B2. Authors are supported in producing accessible content

B3. Authors are supported in improving the accessibility of existing content

B4. Authoring tools promote and integrate their accessibility features.

Contenuti multimediali

- Anche contenuti aggiuntivi come file audio e video possono presentare problemi di accessibilità, ed esistono accorgimenti per evitare tli problemi.

- Accompagnare video con opportune audiodescrizioni e sottotitoli può far sì che essi siano maggiormente fruibili sia per i non vedenti che per i non/ipo udenti.

- Associare al file audio di un podcast la sua trascrizione lo renderà fruibile anche da parte di utenti non udenti.

Post social e documenti PDF

- Col passare del tempo sta diventando possibile la creazione di contenuti accessibili anche sui social network; le principali piattaforme, infatti, oramai supportano funzioni quali l'inserimento di testo alternativo per le immagini o la sottotitolazione dei video.

- Per i documenti PDF, invece, la situazione resta complessa. Nonostante esistano opportuni standard di riferimento (PDF/UA), gli strumenti che consentono di produrre documenti PDF realmente accessibili sono pochissimi, sono molto complessi e richiedono quantità di lavoro aggiuntivo decisamente elevate.

Accessibility essentials

Tip: su cosa concentrarsi?

- Correttezza e conformità agli standard e alle linee guida per l'accessibilità del markup;
- navigazione e operabilità mediante tastiera;
- Gestione del focus;
- Accessibilità dei contenuti (descrizione per le immagini, testi di link ed etichette, etc.);

Risorse (molto) utili

- Web Content Accessibility Guidelines (WCAG) 2.1 | W3C
- understanding WCAG 2.1 | W3C
- Techniques for WCAG 2.1
- Accessible Rich Internet applications (WAI-ARIA 1.1)
- Design patterns | WAI-ARIA 1.1 authoring practices
- Articles | WebAIM
- Accessibility resources and code examples (by Deque Systems)

Domande?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Vincenzo Rubano

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

vincenzo.rubano@unibo.it

www.unibo.it

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Extra

Suggerimenti e trucchi

- Ecco alcuni suggerimenti che possono essere utili per l'implementazione del progetto e più in generale creare applicazioni web accessibili.

Suggerimento I: mistrust the authority

- Se si utilizza un framework di componenti per l'interfaccia utente (bootstrap, Angular-material, element-ui, ecc.), non dare per scontato che tali componenti siano accessibili. Verificare sempre la loro accessibilità ed eventualmente aggirare i problemi riscontrati. Scegliere un framework diverso, se necessario.

Suggerimento II: test con un lettore di schermo

- Esistono prove evidenti del fatto che la navigazione di un'applicazione Web e l'interazione con essa tramite uno screen reader offrono la valutazione più completa della sua accessibilità.
- Essa non copre tutti gli aspetti da verificare, ma è comunque un buon punto di partenza!

Suggerimento III: scegli uno screen reader facile da usare per i test

- Se decidi di testare la tua applicazione web con uno screen reader, assicurati di sapere come usarlo (funzionalità principali, scorciatoie da tastiera, ecc.).
- Sembra ovvio ma, assolutamente, assicurati di sapere come disabilitarlo: gli screen reader cambiano spesso le modalità di interazione con il computer, quindi possono essere considerati invasivi; assicurati di sapere come gestirli.
- Da questo punto di vista Chrome Vox è un'ottima opzione, in quanto offre un ottimo tutorial interattivo introduttivo ed è un'estensione del browser.

Suggerimento III: test automatizzati

- Esamina sempre i problemi segnalati da strumenti automatici, poiché in alcuni casi potrebbero non essere errori reali.
- Distinguere tra problemi segnalati come errori e problemi segnalati come potenziali errori (potrebbero esserlo o no, ma gli strumenti automatizzati non sono in grado di dedurre una risposta).
- In caso di incertezza meglio concentrarsi sugli errori, che vanno risolti.

Cheatsheet di design accessibile

- Come si dovrebbe progettare l'interfaccia del progetto, e in generale qualsiasi interfaccia web, per massimizzare le possibilità che essa sia accessibile? Scopriamolo insieme.

Distinguere widget e design pattern

- Identificare i design pattern richiesti per visualizzare i dati e i widget per rappresentarli, inserirli o interagire con essi in altro modo.
- Cercare di comporre l'interfaccia con il minor numero possibile di widget, sii coerente.

- Chiamando Element ogni widget o design pattern individuato, allora…

Elementi HTML

- Element è disponibile in HTML?
- Bene, usalo ... Non provare a emularlo, a meno che tu non abbia una ragione molto, molto buona per farlo (probabilmente non ce l'hai). Sia e_html tale elemento HTML.
- L'elemento e_html richiede l'utilizzo di attributi particolari per renderlo accessibile? Dovresti essere in grado di rispondere a questa domanda con una buona comprensione dei principi e delle linee guida WCAG 2.1, ma puoi anche scoprirlo guardando "Techniques for WCAG 2.1".
- Regola empirica: se si tratta di un'immagine o di un elemento interattivo (ad es. un campo di un modulo), è così. Riempire tali attributi con valori significativi.

E_html è un'immagine?

- Se e_html è un'immagine statica (ovvero l'interazione con essa non avvia alcuna azione che altera lo stato dell'applicazione), chiediti "e_html è un'immagine decorativa?"
- In tal caso, assicurati che abbia un attributo alt vuoto (alt = "") per trasmettere tale informazione alle tecnologie assistive o visualizzala tramite CSS.
- Altrimenti, assicurati che abbia una descrizione significativa usando l'attributo "alt". Test: disattiva la visualizzazione delle immagini nel browser. Riesci a dare un senso a ciò che viene mostrato in quelle immagini semplicemente leggendo la loro descrizione?

Widget e schemi ARIA

- Element è un widget o design pattern che richiede l’utilizzo della specifica WAI-ARIA per essere reso accessibile? È molto probabile che sia documentato in "Esempi di modelli di progettazione ARIA", parte delle pratiche di authoring WAI-ARIA. Assicurati che si comporti esattamente come documentato nella guida. Sentiti libero di essere ispirato dal codice di quelle pagine per implementarli se necessario (dopo tutto è stato scritto proprio per questo scopo ;)

Gestione del focus

- Più dinamica e sofisticata è la tua applicazione, maggiore è l'importanza della gestione del focus! Ogni volta che si verificano cambiamenti dello stato dell'interfaccia, chiediti:
  - Dov'è il focus?
  - Dove dovrebbe essere?
- La gestione del focus errato o mancante causa il disorientamento degli utenti di tecnologie assistive (una finestra di dialogo modale viene chiusa, ma il focus non viene riportato sull'elemento che ne ha causato l'apertura) e lo stato dell'interfaccia utente cambia inosservato (ad esempio viene visualizzata una finestra di dialogo modale, ma il focus non viene posizionato sul suo primo figlio focalizzabile).
- Spostare il focus su un elemento potrebbe far sì che gli utenti di tecnologie assistive ignorino gli elementi che lo precedono,quindi scegli saggiamente quando farlo.
- Regola empirica: nessun autofocus di una pagina / vista / schermo, a meno che non sia richiesto dal design pattern che la implementa.

Informazioni sui framework esterni

- Realisticamente, è probabile che Element sia un componente fornito da un framework esterno o un elemento HTML migliorato da questo. Occorre sempre assicurarsi che sia accessibile o risolvere i suoi problemi di accessibilità nel tuo codice. Scegli un framework noto per essere un buon punto di partenza in questo senso: Bootstrap (ultima versione 4.x), angular-material, elements-ui per citarne alcuni. In ogni caso, aspettati di fare un po 'di lavoro su questo fronte. Ricorda, sei sempre responsabile per qualsiasi cosa tu consegni.

A proposito di styling

- Sentiti libero di “stilizzare” i tuoi elementi come desideri, ma tieni a mente i principi di accessibilità.
- Presta particolare attenzione al contrasto cromatico, alle dimensioni dei caratteri e rendi il tuo layout il più responsive possibile per rispondere alle modifiche alle dimensioni dei caratteri, allo zoom, ecc.
- Scegli i caratteri che rendono il contenuto più leggibile (tenendo anche presente il contesto).

Test automatizzati

- Gli strumenti di test di accessibilità automatizzati possono aiutarti a identificare i problemi di accessibilità, usali! Fai attenzione a scegliere quelli affidabili.
- Ax per Google Chrome, Ax per Mozilla Firefox da Deque, Systems e / o il Strumento di valutazione Wave per Chrome e Firefox da WebAIM sono raccomandati.
- Assicurati di eseguire tali strumenti per ogni variazione della tua interfaccia (ad es. quando un trigger di modulo si guasta o no, viene presentato o meno un modale, un menu viene espanso o compresso, ecc.).

Test manuale I

- Il test di accessibilità manuale è essenziale.

- È possibile interagire con ogni parte dell'interfaccia esclusivamente utilizzando la tastiera? Verificalo!
- Prova a utilizzare uno screen reader per interagire con la tua applicazione: ha senso il "navigation flow"? La tua interfaccia è utilizzabile, comprensibile e percepibile? Lo stato cambia anche manualmente o automaticamente?

- L'uso di uno screen reader nativo è piuttosto complesso, ma il Chrome Vox è consigliata l'estensione per Google Chrome; ha un ottimo tutorial introduttivo che spiega come usarlo, seguilo!

Test manuale II

- Presta attenzione a identificare le informazioni trasmesse solo dai colori, anche se si spera a questo punto non dovresti averne. Se trovate, ripetere questo processo su quel caso particolare per trovare una rappresentazione accessibile.

- Prova ad ingrandire le dimensioni dei caratteri (almeno da 2x a 4x). La tua interfaccia si adatta bene a questo cambiamento?

Massime priorità

- Supporto tastiera
- Etichette per moduli
- Gestione del focus
- Alternative testuali
- Contrasto cromatico e dimensioni dei caratteri

Non dimenticare i contenuti

- Come abbiamo detto, la progettazione e lo sviluppo sono solo due dei tre principali componenti coinvolti nel rendere accessibile un sistema.
- Ricorda, una delle storie del tuo progetto deve essere accessibile, cioè giocabile anche da persone non vedenti. Ricorda che le persone con disabilità possono usare la tecnologia, ma con adattamenti (tecnologie assistive).
- Non richiedere azioni che una persona disabile non può eseguire (ad esempio, cercare qualcosa che si trova in una posizione elevata per una persona con sedia a rotelle, distinguere tra i colori per i non vedenti, ecc.).
- Ricorda i principi WCAG 2.1

Esempio: Un campo di un form I

```
<span>CVV: </span>
<input type = "text">
<p>Inserisci il codice di tre o quattro cifre della tua
carta di credito, generalmente situato sul retro. </p>
```

Questo controllo è accessibile?

Un campo di un form II

- NOO! Non esiste alcuna relazione tra il campo del modulo e la sua etichetta, così come tra il campo e la sua descrizione. Ecco una versione equivalente e accessibile di esso.

```
<label for="cvv"> CVV: </label>
<input type="text" id="cvv" aria-describedby = "cvvdescr">
<p id="cvvdescr"> Inserisci il codice di tre o quattro cifre
della tua carta di credito, generalmente situato sul retro.
</p>
```

Domande?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Vincenzo Rubano

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

vincenzo.rubano@unibo.it

www.unibo.it