Scuola universitaria professionale
della Svizzera italiana

SUPSI

Accessibilità
WAI-ARIA

Lorenzo Sommaruga

Dic 2018

SUPSI-DTI-ISIN

Corso Applicazioni Web 1 C02049 AA 2018-19

Obiettivi

- (BI-2) Comprendere le specifiche di WAI-ARIA
- (BI-1) Conoscere i principali elementi, proprietà e loro uso
- (BI-3) Saper applicare le linee guida per una valutazione preliminare di accessibilità di un sito

Linee guida per l'accessibilità WAI-ARIA

- Il W3C con la Web Accessibility Initiative (WAI) ha definito varie linee guida specifiche per l'accessibilità
  - WCAG (Web Content Accessibility Guidelines)
    riguardano pagine e applicazioni web
  - UAAG (User Agent Accessibility Guidelines)
    riguardano browsers web
  - ATAG (Authoring Tool Accessibility Guidelines)
    riguardano software usati per creare pagine Web
  - WAI-ARIA (Accessible Rich Internet Applications) (WAI-ARIA)
    - riguardano pagine dinamiche e web app (Ajax, …)
    http://www.w3.org/TR/wai-aria/
- WAI-ARIA
  è una specifica tecnica per rendere il contenuto Web dinamico e interattivo accessibile a persone con disabilità:
  Accessible Rich Internet Applications (WAI-ARIA) 1.0, W3C Rec. 20 March 2014
  Accessible Rich Internet Applications (WAI-ARIA) 1.1, W3C Rec. 14 December 2017

WAI-ARIA - problemi che risolve

- WAI-ARIA fornisce un framework per
  aggiungere attributi che identificano caratteristiche per l'interazione utente
  relazionarli tra loro
  conoscere il loro stato

- WAI-ARIA descrive nuove tecniche di navigazione per marcare regioni, strutture web tipiche (menus, contenuti principali e secondari, banner, etc.)

- Esempio: gli sviluppatori possono identificare regioni nelle pagine e abilitare una loro facile navigazione a utenti con tastiera piuttosto che con tab (uso tradizionale)

WAI-ARIA - Accessibilità

- WAI-ARIA rendono il contenuto più accessibile

- Attributi `role` e `aria-*` permettono di aggiungere info su elementi, come:
  - ruoli (`menuitem`, `slider`, `button`, `link`, etc)
  - stati (`aria-hidden`, `aria-disabled`, etc) o
  - proprietà (`aria-live`, `aria-label`, etc).

Ref. su uso: http://www.w3.org/TR/using-aria

Esempi

- Attributo `role` indica che il generico <*div*> è usato come <*button*>

```
<div tabindex="0" role="button">Invia</div>
```

- Attributo role indica un ruolo che non ha un corrrispondente nativo in HTML

```
<div tabindex="0" role="menuitem">Copia</div>
```

WAI-ARIA – ruoli e categorizzazione

- WAI-ARIA è un insieme di attributi specifici per accessibilità che possono essere aggiunti a dei markup, in particolare HTML

- Si identificano ruoli (roles) che definiscono dei widgets per la UI (sliders, tree controls, etc.), altri per la struttura della pagina (document, region, etc.) o come punti di riferimenti per la navigazione (main, navigation, etc.)

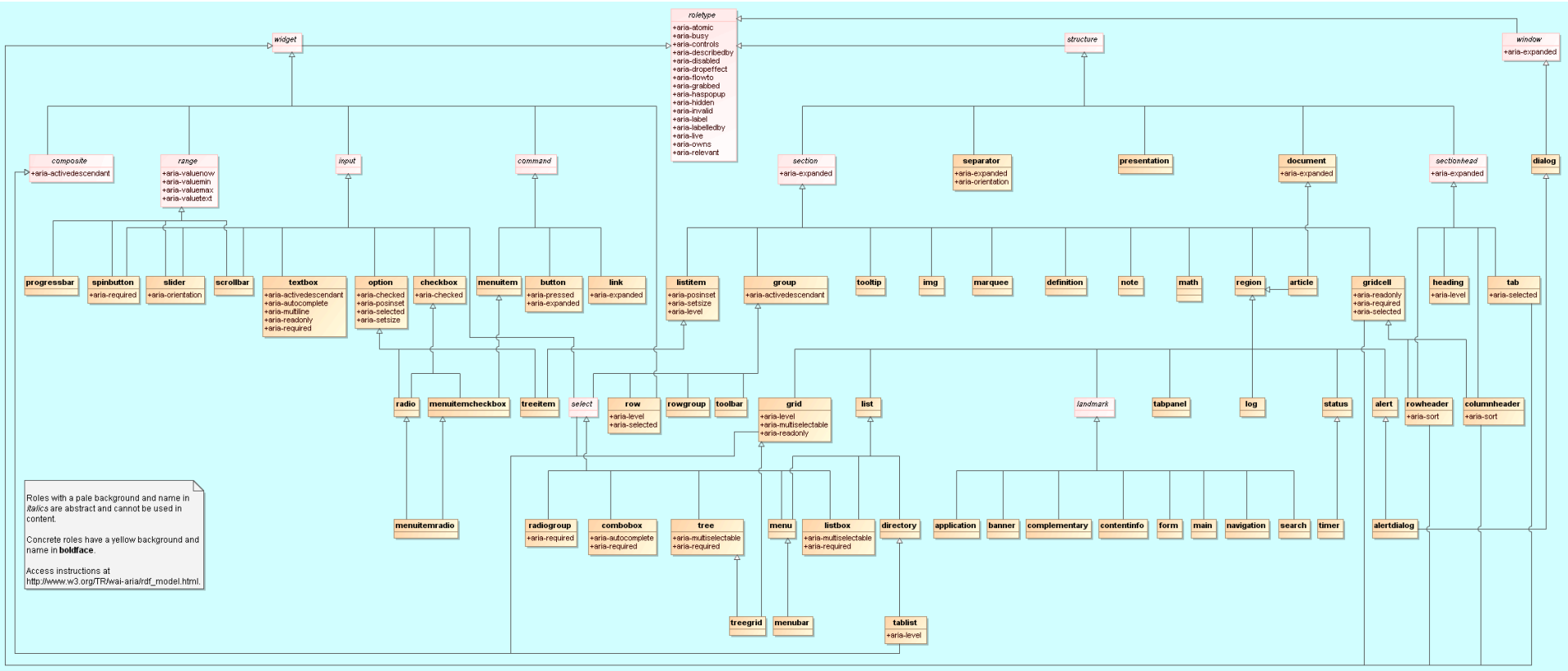
- Divisi in 4 categorie:

  1. Abstract Roles
  2. Widget Roles
  3. Document Structure Roles
  4. Landmark Roles

- ref. http://www.w3.org/TR/wai-aria/#role_definitions

WAI-ARIA – classdiagram dei ruoli

ref. http://www.w3.org/TR/wai-aria/#role_definitions
SVG Class diagram del data model di role



WAI-ARIA – Stati e Proprietà

Stati

- Aggiungono informazioni dinamiche sullo stato attuale di un elemento (aggiornate via JS)
- E.g.

```
<div tabindex="0" role="checkbox" aria-checked="true"/>
```

Live regions

- Usate quando parti di una pagina cambiano
- Proprietà `aria-live` identifica contenuti dinamici
- Descrive il tipo di aggionramento che riceve l'elemento

- E.g.

```
<div id="status" role="alert" aria-live="assertive"
    class="online"> <p>You are online.</p> </div>
```

WAI-ARIA – Landmarks

Landmarks

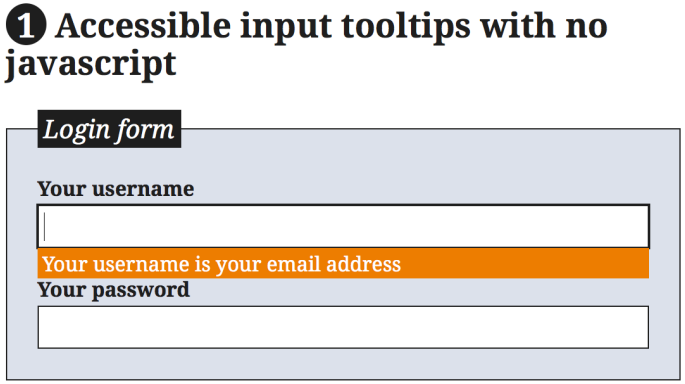
- Oltre alle convenzioni sulla strutturazione delle pagine con HTML5 (sections, list, etc.)
- Usate per identificare aree distinte nella pagina
- E.g.

```
<div tabindex="0" role="search">Cerca nel sito …</div>
```

- Generalmente tre livelli nell'uso di landmarks:
  1. A livello di navigation della pagina
  2. A livello di sidebar
  3. Nel main content

WAI-ARIA – Un esempio HTML

- Alcuni attributi ARIA in un form




```
<form action="">
  <fieldset>
    <legend>Login form</legend>
    <div>
      <label for="username">Your username</label>
      <input type="text" id="username" aria-describedby="username-tip" required />
      <div role="tooltip" id="username-tip »>Username is your email address</div>
    </div>
    <div>
      <label for="password">Your password</label>
      <input type="text" id="password" aria-describedby="password-tip" required />
      <div role="tooltip" id="password-tip">Emailed to you when signed up</div>
    </div>
  </fieldset>
</form>
```

```
CSS
input:focus + [role="tooltip”] {
    display: block;
    position: absolute;
    top: 100%; }
```

Conclusioni

- Nello sviluppo di web application:

  1. Condurre test di accessibilità
  2. Chiedere feedback a utenti
  3. Garantire la migliore user experience

- Suggerimento di altro strumento utile: ChromeVox screen reader

Quando usare WAI-ARIA

Ref. https://developer.mozilla.org/it/docs/Learn/Accessibilit%C3%A0/WAI-ARIA_basics

Quando dovresti usare WAI-ARIA?

Abbiamo già discusso di alcuni dei problemi che hanno spinto alla creazione di WAI-ARIA, ma essenzialmente ci sono 4 grandi aree in cui WAI-ARIA è utile:

1. **Indicatori/riferimenti**: gli attributi role funzionano come descrizioni che possono replicare il valore semantico degli elementi HTML5 (per esempio <nav>), oppure andare oltre HTML5 e funzionare come indicatori che descrivono differenti aree funzionali, per esempio search, tabgroup, tab, listbox, etc.
2. **Aggiornamento dinamico del contenuto**: i lettori di schermo in generale hanno difficoltà a indicare quando il contenuto subisce cambiamenti; con ARIA possiamo usare aria-live per indicare agli utenti che usano lettori di schermo quando un' area del contenuto viene aggiornata, per esempio tramite XMLHttpRequest, o DOM APIs .
3. **Migliorare l'accessibilità da tastiera**: ci sono elementi HTML che hanno accessibilità da tastiera nativa; quando però invece di usare tali elementi se ne usano altri che li "simulano" in combinazione con JavaScript, l'accessibilità da tastiera e la qualità di lettura dei lettori di schermo ne risentono. In questi casi possiamo usare WAI-ARIA per dare focus a tali elementi (usando tabindex).
4. **Accessibilità dei controlli non semantici**: quando si usano una serie di annidati in combinazione con CSS/JavaScript per creare una funzionalità IU particolarmente complessa, oppure quando un controllo nativo viene notevolmente modificato tramite JavaScript, l'accessibilità può esserne danneggiata. Gli utenti che usano lettori di schermo troveranno difficile capire come funzionano gli elementi se non ci sono indicazioni semantiche che lo spieghino. In situazioni come queste la tecnologia ARIA può aiutare a fornire le indicazioni necessarie tramite una combinazione di ruoli come button, listbox, o tabgroup, e proprietà come aria-required o aria-posinset.

Ricorda: **dovresti ricorrere a WAI-ARIA solo quando è necessario!** Idealmente, dovresti usare *sempre* funzionalità HTML native per fornire le indicazioni semantiche necessarie ai lettori di schermo per interpretare correttamente il contesto. A volte però ciò non è possibile, forse perchè non hai pieno controllo sul codice, o perchè stai creando qualcosa di particolarmente complesso, che non puoi implementare con un elemento HTML standard. In tali casi, WAI-ARIA può essere un utile strumento di miglioramento dell'accessibilità.

Bibliografia

- WAI-ARIA
  Accessible Rich Internet Applications (WAI-ARIA) 1.0, W3C Rec. 20 March 2014
  http://www.w3.org/TR/wai-aria/
- Accessible Rich Internet Applications (WAI-ARIA) 1.1
  W3C Recommendation 14 December 2017
  http://www.w3.org/TR/wai-aria-1.1/
- WAI-ARIA Overview
  http://www.w3.org/WAI/intro/aria
- WAI-ARIA in HTML5.2 W3C's rec 14 dic 2017
  http://www.w3.org/TR/html5/dom.html#wai-aria
- Using WAI-ARIA in HTML, W3C Editor's Draft 20 October 2015
  http://rawgit.com/w3c/aria-in-html/master/index.html#recommendations-table
  Moved to https://w3c.github.io/using-aria/
- W3C Working Draft on "using ARIA" in HTML - 5 regole et al
  http://www.w3.org/TR/aria-in-html/

Esempi Pratici

- Esempi pratici: ARIA examples

  Some practical ARIA examples to enhance your application accessibility

  http://heydonworks.com/practical_aria_examples

- Bootstrap components

  Esempi di uso di ARIA in un framework

  http://v4-alpha.getbootstrap.com/components/buttons/

- WAI-ARIA Authoring Practices 1.1, W3C Working Group Note 14 December 2017

  Esempi pratici di uso di ARIA in html con design patterns

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