Istituto di Informatica e Telematica del CNR

ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

Le RIA accessibili

Maria Claudia Buzzi — **IIT – CNR**

Barbara Leporini — **ISTI – CNR**

Corso Aquarius – Modulo G

Riferimenti (W3C)

- WAI-ARIA Overview
- WAI-ARIA
- WAI-ARIA 1.0 Authoring Practices
- WAI-ARIA Primer

Sommario

- Rich Internet Application
- Problematiche RIA quando si utilizzano AT
- WAI-ARIA suite
- Screen Reader
- Esempi di interazione via screen reader
- Esempi di soluzioni tecniche

Rich Internet Application

Applicazioni Web che rendono disponibili funzionalità che vanno oltre le capacità di un'interfaccia solo HTML

HTML non permette di creare contenuti dinamici o controlli avanzati per l'interfaccia utente, ma consente **l'inclusione di applet (Flash, Java)** e **script lato client (tipicamente JavaScript)** per farlo

- Per es. script lato client per aggiornare sezioni di una pagina senza richiederne una nuova al server

Rich Internet Application

Contenuti dinamici e controlli avanzati dell'interfaccia utente (UI) sviluppati con Ajax (Asynchronous JavaScript and XML), HTML, Javascript e tecnologie correlate

Applicazioni interattive, multimediali e di veloce esecuzione (uso asincrono di JavaScript)**:**

- parte **applicazione che elabora i dati** è trasferita a livello **client** (risposta veloce all'interfaccia utente - UI)
- **maggior parte dati e applicazione** è sul **server** remoto, con alleggerimento per il computer utente

Le RIA si fondano su un'architettura di tipo **distribuito**.
L'interazione con una RIA avviene in remoto, tramite un web browser.

Esempi RIA

Wikipedia

Facebook

YouTube

GoogleDocs

GoogleTalk

GoogleMaps

Delicious

Flickr

….

Tecnologie Assistive

Le persone con disabilità utilizzano Tecnologie Assistive (**AT**) **HW o SW** per interagire con computer e contenuti Web:

* **screen magnifier**
* **screen reader**
  - synthesized speech o un refreshable Braille display
* **text-to-speech software**
* **speech recognition software**
* **alternate input technologies**
  - head pointers, on-screen keyboards, single switches, …
* **alternate pointing devices**, usati per simulare puntamento e click del mouse

Tecnologie Assistive (AT)

Le AT possono **trasformare la presentazione dei contenuti** in un **formato più adatto per l'utente**, e possono consentire all'utente **di interagire in modi diversi**

Semantica dei contenuti

Un utente di AT **deve**:

poter fruire delle informazioni presenti nelle applicazioni ed essere informato dei cambiamenti dinamici che avvengono nell'interfaccia dell'applicazione caricata nel browser

Per poter fare questo nel modo migliore, l'AT deve capire la **semantica** dei **contenuti**: **ruoli, stati e proprietà** della UI e degli elementi in essa contenuti, e comunicarla all'utente

RIA unaccessibility

Applicazioni Web complesse diventano **non accessibili** quando:

- **AT non riescono a determinare la semantica di porzioni di un documento**
- o quando **l'utente non può navigare in maniera efficace** e **usabile** in ogni parte del documento

Problemi

Il **caricamento asincrono** di parte della pagina può essere un problema per gli utenti che utilizzano Tecnologie Assistive (AT) e che non possono utilizzare il mouse.

Lo **screen reader** ad esempio **non è pensato per intercettare i caricamenti asincroni** e quindi **non riesce a trasmettere correttamente agli utenti non vedenti il variare di queste informazioni**

Informare dei cambiamenti: come?

Comportamento intelligente delle AT, che, per trattare dati provenienti da un'applicazione dinamica, devono fare scelte complesse **basate sulla semantica, su metadati e sul livello di priorità** dei contenuti aggiornati.

Informare dei cambiamenti: come?

Importante è anche il modo in cui si viene informati

Es: e-mail in un'applicazione Ajax

- mentre l'utente è impegnato a digitare, lo screen reader non deve dirottare il focus verso il pannello dei messaggi in arrivo per la notifica di una nuova e-mail

Per l'utente è importante che le comunicazioni non rallentino troppo o rendano difficoltosa l'interazione

Informare dei cambiamenti

Attualmente, il comportamento intelligente è realizzabile in misura minima, perché gli **strumenti per comunicare a una tecnologia assistiva il valore semantico di un oggetto** modificato dinamicamente o **non esistono o sono molto rudimentali**

Problemi

Controlli ad albero per la navigazione di un sito Web

La tecnologia assistiva deve essere capace di interagire con questi controlli

Funzionalità drag-and-drop

non disponibile per utenti che usano la tastiera e non possono usare il mouse

Problemi

Contenuti che cambiano in base ad azioni dell'utente o del tempo o di aggiornamenti per eventi

non veicolati ad utenti che utilizzano screen reader o con disabilità cognitive

Anche **semplici siti Web** possono risultare difficili da utilizzare se nella navigazione via tastiera richiedono di premere contemporaneamente più tasti ('keystrokes')

Problemi

Gli **autori** di contenuti generati mediante **JavaScript** non usano elementi tag standard ma **fanno ampio uso di elementi** come i DIV **a cui applicano dinamicamente proprietà di presentazione mediante fogli di stile** e gestiscono cambiamenti dinamici del contenuto

- **I tag DIV non forniscono informazioni semantiche**

Es**.:** Menù di navigazione in un documento HTML, inserito all'interno di un DIV e reso visibile e invisibile tramite JavaScript

Il fatto che il blocco nel DIV sia un menu di navigazione e che sia in un certo momento visibile o invisibile è **informazione semanticamente rilevanti, che deve essere comunicata all'utente**

Problemi semantici

Il tag HTML DIV non fornisce informazione semantica:

- Non identifica il ruolo del DIV (come menu′ di navigazione, pop-up menù, ecc.)
- Non avverte la AT quando l′elemento ha il focus
- Non veicola informazioni utili per l′accessibilità, come se un pop-up menù è collassato o espanso
- Non definisce quali azioni possono essere eseguite sull′elemento

- JavaScript necessita di una **architettura di accessibilità** che possa essere **mappata sul framework di accessibilità della piattaforma** nativa dello user agent

Document Object Model (DOM)

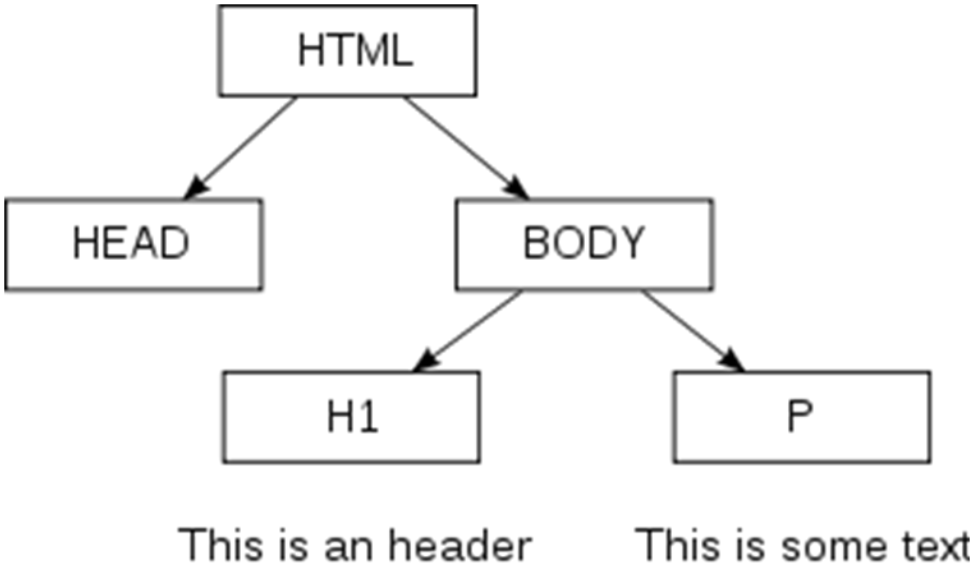
DOM: forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti

L'interfaccia pubblica del DOM è specificata nella sua Application Programming Interface (API)

Document Object Model (DOM)

DOM è lo standard ufficiale del W3C per la **rappresentazione di documenti strutturati in maniera da essere neutrali sia per la lingua che per la piattaforma** [Wikipedia]

Es: Rappresentazione ad albero del DOM di un semplice documento HTML

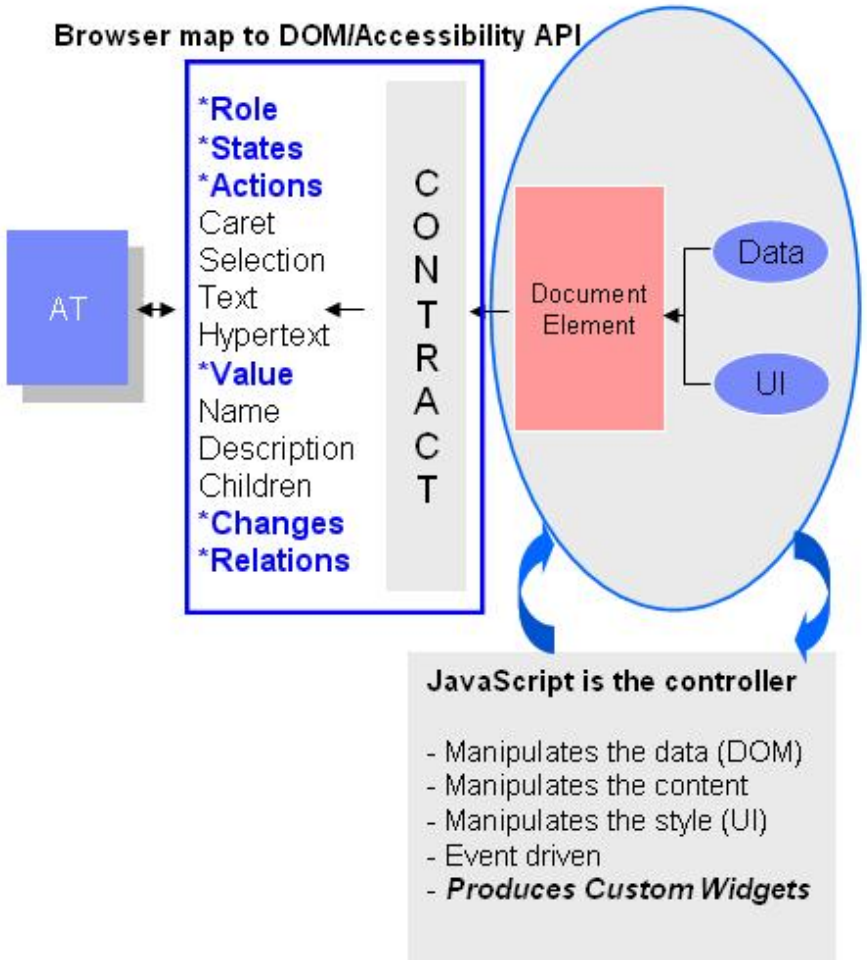


Javascript

JavaScript sovrascrive il comportamento di default dello user agent nel nodo DOM

Per produrre widget personalizzati si **manipolano i dati, il contenuto, e lo stile in risposta a eventi causati dall'interazione dell'utente**

L'informazione di default dell'accessibilità non è più valida



Problemi

Ajax è basato su Javascript

I **browser che non supportano JavaScript**, **precludono l'accesso** alle informazioni veicolate con **Ajax**

utente dipende dal dispositivo/browser usato

Necessità

Fornire informazioni semantiche su widgets, strutture e comportamenti per permettere alle tecnologie assistive di veicolare le informazioni appropriate alle persone con disabilità

Suite WAI - ARIA

E' un insieme di documenti pubblicati dal W3C (WAI, Web Accessibility Initiative) che specificano come **aumentare l'accessibilità dei contenuti Web dinamici e dei componenti per l'UI** sviluppati con Ajax, HTML, JavaScript e altre tecnologie collegate

Specifica per **utenti che utilizzano screen reader e non usano il mouse** o altri dispositivi di puntamento

Working Draft

Suite WAI - ARIA

- **WAI-ARIA Technical specification**: combina i precedenti draft WAI-ARIA Roles e WAI-ARIA States and Properties

- **WAI-ARIA Primer**: introduzione per gli sviluppatori ai problemi di accessibilità, ai concetti fondamentali e all'approccio tecnico di ARIA

- **WAI-ARIA Authoring Practices**: descrive come gli sviluppatori di contenuti Web possono sviluppare RIA accessibili usando ARIA

- **WAI-ARIA User Agent Implementation Guide**: descrive come i browsers e gli altri User Agents devono supportare WAI-ARIA

Suite WAI - ARIA

Fornisce un framework per aggiungere a elementi della pagina **attributi** per identificare le possibili interazioni, eventuali **correlazioni**, e il loro **stato** corrente

Le tecniche WAI-ARIA si applicano a widgets come **bottoni, toolbar, liste drop-down, funzioni calendario, controlli ad albero** (come menù espandibili), ecc.

Suite WAI - ARIA

Descrive **tecniche di navigazione per marcare regioni e strutture** come menù, contenuti primari e secondari, banner, ecc.

Include tecnologie per **mappare controlli, live regions Ajax ed eventi alle APIs per l'accessibilità**, includendo controlli personalizzati usati dalle RIA

Suite WAI - ARIA

WAI-ARIA fornisce **semantica** ai widgets rendendoli **accessibili, usabili ed interoperabili con le tecnologie assistive**.

Vengono **identificati** i tipi di **widgets e strutture** fornendo loro un **ruolo**, in modo da permettere all'AT di sapere come elaborarli, e vengono assegnati **stati e proprietà** per definire attributi importanti per comprenderne il funzionamento.

Suite WAI - ARIA

WAI-ARIA x aggiungere **informazioni semantiche a pagine Web/Rich Internet widgets:**

* **Ruoli** (roles) x descrivere il **tipo di widget:**

  menu, treeitem, slider, progressmeter,…

* **Ruoli** x descrivere la **struttura della pagina Web**

  headings, regions, tables (grids)

Suite WAI - ARIA

* **Proprietà** x descrivere lo **stato del widget**

  checked - check box

  haspopup - menu che presenta un sotto-menu o altri popup

  expanded/collapsed - tree node

Suite WAI - ARIA

* **Proprietà** per definire **live region** (che possono essere aggiornate) e **interruption policy** per segnalare/comunicare gli aggiornamenti

Le AT (tecnologie assistive) possono comunicare **aggiornamenti critici appena si presentano**

Aggiornamenti incidentali possono essere presentati dopo aver completato il task corrente

Attributo aria-live

Suite WAI - ARIA

* **Proprietà** per **drag-and-drop** che descrivono **sorgenti** (del trascinamento) **e destinazioni** (del rilascio)

* **Metodo** per fornire **navigazione via keyboard per rich internet widgets**

Suite WAI - ARIA

Possibilità fornite da WAI-ARIA + informazione strutturale veicolata dal DOM permettono a chi crea pagine Web di **produrre soluzioni interoperabili con le tecnologie assistive**

WAI-ARIA Roles

Associare ad un elemento (widget/struttura) un **WAI-ARIA Role** usando l'attributo ***role*** per fornire alle AT informazioni su come trattare l'elemento stesso ed elaborarlo

** **

WAI-ARIA role taxonomy

Ruoli specificano dettagliatamente elementi d'interfaccia per i quali HTML e XHTML non possiedono specifici marcatori strutturali

http://www.w3.org/TR/2010/WD-wai-aria-20100916/roles#role_definitions

WAI-ARIA Roles (esempi)

slider:

elemento d'interfaccia che svolge il ruolo di cursore, per esempio una barra con un indicatore, che definisce il valore attuale in una gamma di valori possibili

tree:

elementi che formano una struttura ad albero, per es. una lista contenente sottoliste, le quali possono essere espanse o collassate (cioè rese visibili o nascoste)

region:

gruppo di elementi che insieme formano un'ampia sezione percepibile che l'autore ritiene dovrebbe essere inclusa in un sommario della pagina

toolbar:

collezione di funzioni usate comunemente, rappresentate in una forma visuale compatta

WAI-ARIA Roles (esempi)

alert:

messaggio con informazioni importanti generalmente time-sensitive

application:

regione dichiarata come Web application (vs Web document)

button:

Input che consente azioni user-triggered se premuto

composite

Widget che può contenere figli o discendenti che possono essere navigati

WAI-ARIA Roles

I ruoli si possono dividere in due tipologie: Widget e strutturali.

Widget: progressbar, slider, button, tree, textfield, checkbox, alert, dialog, ….

Strutturale: main, secondary, group, section, liveregion (il cui contenuto viene aggiornato da AJAX o altre tecniche), navigation, search, o banner

WAI-ARIA States and Properties

Ogni elemento, a seconda del suo ruolo, può trovarsi in una serie di stati differenti e avere differenti proprietà

Stati e proprietà possono essere modificati dinamicamente da un linguaggio di scripting

AT deve sapere in quale stato si trova un dato elemento d'interfaccia, così da poter comunicare l'informazione all'utente

WAI-ARIA States and Properties

Dopo aver identificato i tipi di widgets e le strutture fornendo loro un **ruolo**, vengono assegnati **stati e proprietà** per definire **attributi importanti per comprenderne il funzionamento**

Forniscono **informazioni specifiche** sull'oggetto e **fanno parte della definizione della natura dei ruoli**

Entrambi sono attributi markup con prefisso **aria-**

WAI-ARIA States

Uno **stato** (come aria-checked) è un proprietà dinamica che esprime **caratteristiche di un oggetto che può cambiare in risposta all'azione dell'utente o a processi automatici**

Gli stati non modificano la natura essenziale dell'oggetto ma rappresentano **dati associati all'oggetto o alle possibili interazioni dell'utente**

WAI-ARIA Properties

Attributi che sono **essenziali per la natura dell'oggetto** o che rappresentano **un valore dei dati associati con l'oggetto stesso**

Un **cambio di una proprietà** può impattare significativamente sul **significato** o sulla **presentazione** di un oggetto

WAI-ARIA States and Properties

Relativi a caratteristiche simili, forniscono entrambi **informazioni su un oggetto** e contribuiscono a specificare la **natura del Role**

Trattati in modo distinto x la lieve differenza del loro significato

Il valore di una proprietà (come aria-labelledby) **è meno soggetto a cambiamenti durante il ciclo di vita dell'applicazione del valore di uno stato** (come aria-checked) che può cambiare frequentemente per l'interazione dell'utente

WAI-ARIA States and Properties

W3C Working Draft

Characteristics of button

Characteristic	Value
Superclass Role:	command
Base Concept:	HTML button
Related Concepts:	link XForms trigger
Supported States and Properties:	aria-expanded (state) aria-pressed (state)
Inherited States and Properties:	aria-atomic aria-busy (state) aria-controls aria-describedby aria-disabled (state) aria-dropeffect aria-flowto aria-grabbed (state) aria-haspopup aria-hidden (state) aria-invalid (state) aria-label aria-labelledby aria-live aria-owns aria-relevant
Name From:	contents author
Accessible Name Required:	True
Children Presentational:	True

WAI-ARIA States and Properties

W3C Working Draft

aria-expanded (state)

Indicates whether the element, or another grouping element it controls, is currently expanded or collapsed.

For example, this indicates whether a portion of a tree is expanded or collapsed. In other instances, this may be applied to page sections to mark expandable and collapsible regions that are flexible for managing content density. Simplifying a user interface by collapsing sections may improve usability for all, including those with cognitive or developmental disabilities.

If the element with the `aria-expanded` attribute controls the expansion of another grouping container that is not 'owned by' the element, the author **SHOULD** reference the container by using the `aria-controls` attribute.

Characteristics of aria-expanded

Characteristic	Value
Related Concepts:	Tapered prompts in voice browsing. Switch in *SMIL* [*SMIL*].

Used in Roles:	`button`
	`document`
	`link`
	`section`
	`sectionhead`
	`separator`
	`window`

Value:	true/false/undefined

Values of aria-expanded

Value	Description
true:	The element, or another grouping element it controls, is expanded.
false:	The element, or another grouping element it controls, is collapsed.
undefined (default):	The element, or another grouping element it controls, is neither expandable nor collapsible; all its child elements are shown or there are no child elements.

Landmark roles

Ci sono Ruoli che indicano **regioni** della pagina intese come Landmarks di navigazione semantica

Aree percepibili di una pagina/documento che contengono **informazioni associabili semanticamente**. Possibile suddividerle in sottoregioni, a seconda dalla complessità dell'applicazione Web

Ad ogni regione va associata una landmark di navigazione e un heading referenziato da aria-labelledby

Landmark roles

Le landmark sono un miglioramento rispetto alla tecnica rudimentale "skip to main content“ che:

- può impattare sul **posizionamento** degli elementi dell’applicazione web => possono essere necessari aggiustamenti del layout
- è **limitato**, indirizzando solo una delle aree pricipali.

WAI-ARIA fornisce un insieme di landmark che sono applicabili ad ogni regione della pagina/interfaccia

Landmark roles

Le landmarks permettono alla AT di fornire un’esperienza di navigazione consistente

Semplificano la navigazione e la quantità di informazione da processare (utenti con disabilità cognitive o dell’apprendimento)

La navigation con le landmarks è indipendente dal device

```
<div role="log" title="chat log">
<div role="region" title="Game Statistics">
```

Landmark roles

- application: una regione dichiarata come applicazione Web (in opposizione a documenti Web)
- banner una regione che contiene il primo heading o il titolo di una pagina
- complementary qualsiasi sezione del documento che supporta ma è separabile dal contenuto principale, ed è significativa anche da sola
- contentinfo Meta informazione che si applica al primo immediato antenato il cui ruolo non sia presentation
- form una regione del documento che representa una insieme di elementi di un form
- main contenuto principale del documento
- navigation un insieme di link per la navigazione del documento o di documenti correlati
- search lo strumento di ricerca

Application vs Web Document

- Deve prevalere la predominanza
- Se si imposta un ruolo applicazione sul tag body, si dovrebbe impostare il focus su un widget al caricamento della pagina
- Se la pagina ha solo pochi widgets isolati, come pop-up calendars, non importa settare il ruolo ad application. Gli Screen readers devono fornire l'accesso a questi widgets anche senza riconoscere l'intera pagina come applicazione

Esempio landmarks

```
<div role="banner">

...

</div>


<div role="navigation">

...

</div>


<div role="main">

...

</div>
```

Esempio landmarks

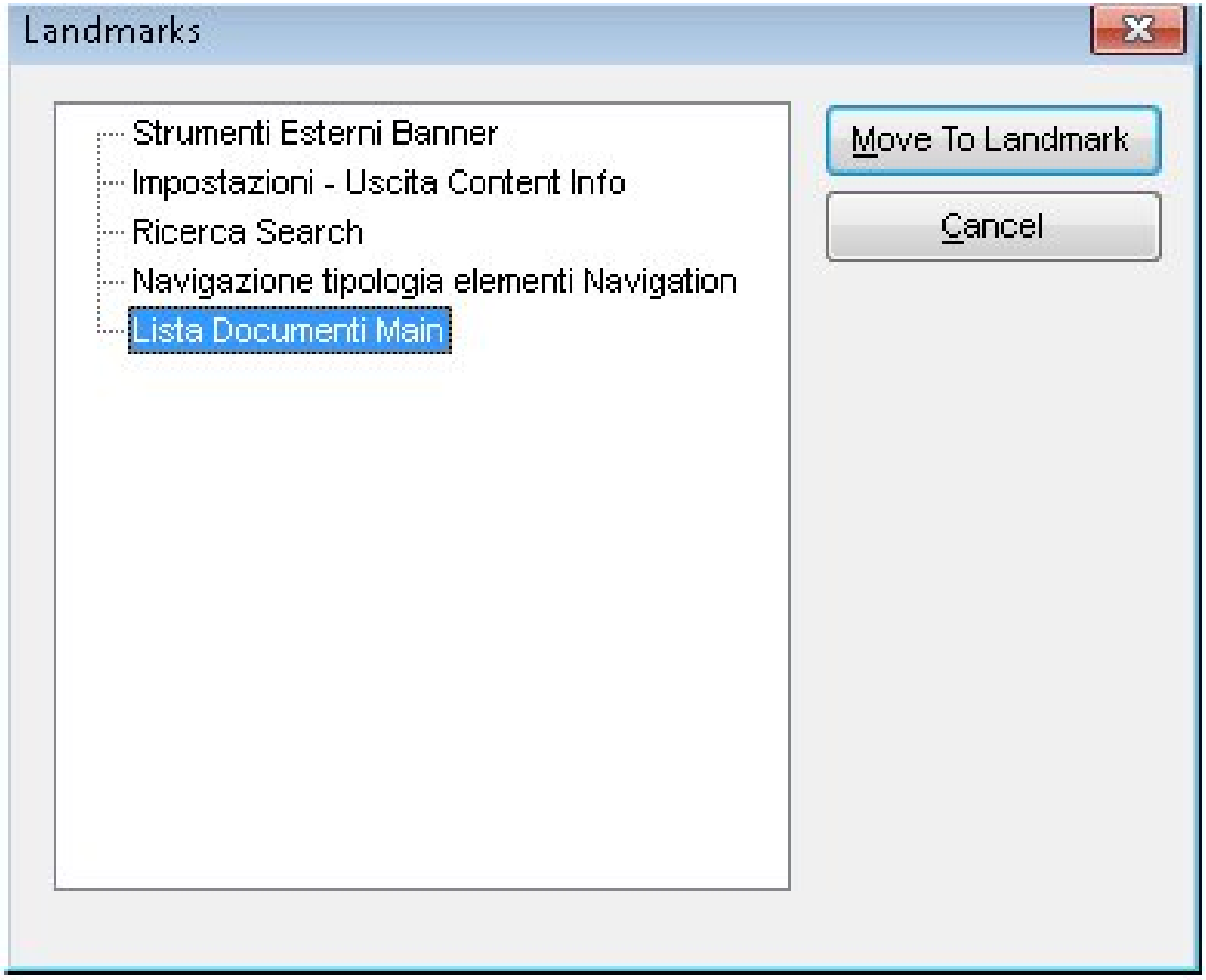
```
<div role="dialog" aria-labelledby="dialogheader">
<h2 id="dialogheader">Sei sicuro?</h2>
... Dialog contents
</div>
```

Landmark roles

Nell'identificare una regione di una pagina Web, gli autori usano generalmente **landmarks roles standard**

Se queste definizioni sono inadeguate, **è possibile usare landmarks personalizzate, mediante role 'region' + un nome appropriato accessibile**

Landmark roles



Landmark roles

Tecniche sufficienti

- **Raggruppare blocchi di materiale ripetuto** in modo che possano **essere saltati**
  - H69: Fornire elementi **heading** all'inizio di ogni sezione del contenuto (HTML)
- **Intestazioni ed etichette**
  - G130: Fornire **heading** descrittivi
- **Informazione e Relazioni**
  - H42: Usare **h1-h6** per identificare le **intestazioni** (HTML)

Live Region

Le live regions sono parti di una pagina Web che possono cambiare:

- **tabelle con contenuti che si aggiornano dinamicamente (risultati sportivi, valori/informazioni su azioni, ecc.)**
- **logs dove nuove informazioni sono aggiunte**
- **aree di notifica (alert, stati, ecc.)**
- **…**

Live Region

Attributi specifici per definire **live region** nelle rich internet applications

Scopo: **indicare i cambiamenti del contenuto** che possono esserci senza che vi sia il focus (e che l'utente se ne possa accorgere) e fornire alle AT **informazioni su come processare tali aggiornamenti**

* aria-atomic
* aria-busy (state)
* aria-live
* aria-relevant

Alcuni **ruoli** specificano il **valore di default dell'attributo aria-live** specifico

Live Region

Attributi per specificare Interruption policy

Politeness: indica che **grado di priorità** che deve essere considerato per le notifiche in una live region

Valori di politeness disponibili per **attributo**

aria-live: **off | polite | assertive | rude**

Live Region

aria-live="off"

Default. **Qualsiasi aggiornamento non deve essere annunciato all'utente**. Per impostare cose che cambiano molto frequentemente come coordinate GPS per un veicolo in movimento

aria-live="polite"

Qualsiasi aggiornamento deve essere annunciato solo se l'utente non sta facendo altro. Deve essere usato nella maggior parte delle situazioni, in modo da presentare all'utente le nuove informazioni ma non interromperlo nello svolgimento dei task e disorientarlo

Live Region

aria-live="assertive"

Qualsiasi aggiornamento è importante e deve essere annunciato all'utente il prima possibile, anche se non è necessario interrompere immediatamente l'utente. Deve essere usato per esempio con messaggi di warning nella form che si sta riempiendo

Nella precedente versione del draft:

aria-live="rude": indica il più alto livello di priorità, tale da dover interrompere l'attività dell'utente, anche a costo di creare un disorientamento

Live Region

aria-busy (state)="true"

Per interrompere la presentazione dei cambiamenti fino a chè una regione non ha terminato l'aggiornamento oppure fino a che una sequenza di aggiornamenti veloci non è terminata

Finchè è settato a true, l'AT terrà conto delle modifiche ma le annuncerà solo quando la regione non sarà più busy

Live Region

Alcune volte a seguito di un aggiornamento, **tutta la live region deve essere letta per poter rendere all'utente il senso della modifica**. Ad esempio non ha senso dare solo il valore aggiornato di un azione, senza dire il nome dell'azione

aria-atomic="false"

Default. Il cambiamento nella regione può essere presentato da solo

aria-atomic="true"

Deve essere presentata tutta la live region

Live Region

Non tutti gli aggiornamenti sono sempre rilevanti

Es: se in una lista viene **rimossa un'intestazione e sostituita con una nuova**, la vecchia probabilmente **non** è **importante** da dover essere annunciata all'utente

Diversamente in un'applicazione chat, **quando un utente esce dalla chat** la sua rimozione dai presenti può essere **rilevante** abbastanza da essere annunciata

Cambiamenti **non rilevanti** saranno trattati come se la regione avesse **aria-live="off" e non saranno annunciati**

Live Region

aria-relevant="additions"

L'inserimento di nodi nella live region deve essere considerato **rilevante**

aria-relevant="removals"

La **rimozione** di nodi nella live region deve essere considerato **rilevante**

aria-relevant="text"

Cambiamenti al contenuto testuale di nodi già esistenti nella live region devono essere considerati **rilevanti**

Contenuti testuali includono ad esempio **text equivalents**, come l'attributo **alt delle immagini**

Live Region

Quando si applicano proprietà live region, è raccomandato effettuare uno user testing (AT, eventualmente anche user)

Operazioni di Drag-and-drop

- **Difficili per persone con disabilità**
  - **Problemi per l'utilizzo del mouse** (disabilità motorie, non vedenti)
  - **Screen readers e i sistemi di input alternativi permettono** all'utente di **simulare** un click, trascinamento, e rilascio
  - L'utente deve trovare un **target** per l'oggetto trascinato e potrebbe non essere a conoscenza se **l'operazione di rilascio desiderata è supportata** così come **quali oggetti possono essere selezionati per trascinamento**

Operazioni di Drag-and-drop

- **WAI-ARIA introduce due nuove proprietà specifiche per le operazioni Drag and Drop:**
  - **aria-grabbed**

    si utilizza per la sorgente/i da draggare
  - **aria-dropeffect**

    si utilizza per il/i target

Drag-and-drop

- **Identificare oggetti draggable (es.:** listitem and treeitem)

  Lo stato di **default** per tutti gli oggetti è indefinito, non draggable.
  Per oggetti che possono essere draggati, settare lo stato
  aria-grabbed a **"false"**

  Tutti gli oggetti draggable devono essere navigabili via tastiera

- **Consentire** all'utente di iniziare l'appropriata operazione di **trascinamento** (drag) **usando la tastiera**

  Raccomandato usare space bar x la selezione; shift + space x oggetti multipli contigui; control + space x multipli non contigui

  aria-grabbed a "**true**"

Drag-and-drop

- **Marcare i target di rilascio** (drop) settando **aria-droppeffect**:
  - **none**: non vengono supportate operazioni di drag. **Default**
  - **copy**: duplica la sorgente
  - **move**: sposta la sorgente
  - **link**: crea una reference o short cut
  - **execute**: viene eseguita la funzione supportata dal target, usando la sorgente come input
  - **popup**: viene fornito un popup menu/dialog per consentire all'utente di scegliere le operazioni di drag (copy, move) o altre operaz. come cancel

Drag-and-drop

- **Implementare funzionalità di tastiera** per assistere l'utente e la AT nell'esecuzione del drop

  es.: Permettere navigazione del target

- **Cancellare una operazione di trascinamento**

  premendo ESC (eccezione popup)

- **Ripulire dopo drag-and-drop (DOM)**

  aria-droppeffect: none; aria-grabbed: false

- **Documentare la navigazione via tastiera se viene utilizzata una modalità differente da quella raccomandata**

Costruire un widget accessibile

1. Selezionare il **tipo di widget** (role) dalla tassonomia WAI-ARIA (http://www.w3.org/TR/2010/WD-wai-aria-20100916/roles#role_definitions)
2. Partendo dal ruolo, prendere in esame la lista degli **stati e delle proprietà supportate** per utilizzare quelli necessari
3. Stabilire la **struttura del widget** nel markup (parent/child)

   tutte le funzioni che si vogliono inserire widget devono essere contenute in esso (es.: toolbar)
4. Ripetere i passi 1-3 per gli **elementi figli** (child)

Costruire un widget accessibile

5. Stabilire la **navigazione via keyboard del widget** e pianificare come sarà navigato all'interno del documento (deve esserci un equivalente via keyboard per ogni operazione via mouse)

   es. toolbar: focus ai children tramite aria-activedescendant property; tabindex per ordinamento via tab

6. Applicare e gestire **stati e proprietà WAI-ARIA** necessari in risposta a eventi generati da user input

7. Sincronizzare la **UI visuale con gli stati e le proprietà di accessibilità** per supportare gli user agent

   Collegare modifiche della visual UI direttamente a modifiche di stati e proprietà WAI-ARIA

Costruire un widget accessibile

8. Mostrare e nascondere **Sections in un Widget**

   per menus, alerts, dialogs che appaiono o scompaiono: **aria-hidden** (true/false); **CSS**: **display** (block/none); **visibility** (visible/hidden)

9. Fornire l'**accessibilità di base**

   es.: testo alternativo per le immagini, soprattutto se rappresentano informazioni dentro un componente

10. Stabilire eventuali relazioni WAI-ARIA tra questo widget e altri

    aria-flowto (elemento successivo nel ordine raccomandato di lettura)

Costruire un widget accessibile

11. Ricontrollare il widget per essere sicuri di **non** aver le dimensioni **hard coded** (font e contenitori)
12. Fare **test con user agent, tecnologia assistiva e persone con disabilità**

Tecniche sufficienti

- **2.1.1 Tastiera**
- G202: Assicurare il **controllo da tastiera per tutte le funzionalità**
  - H91: Usare controlli di form e link HTML (HTML)
- G90: Fornire gestori di eventi **azionabili da tastiera** (keyboard-triggered) usando una delle seguenti tecniche:
  - SCR20: Usare sia **tastiera** sia altre **funzioni specifiche del dispositivo** (Scripting)

Focus

In html e xhtml gli oggetti che possono ricevere il focus sono **link** e gli **elementi di controllo dei form**.

Con ARIA è possibile **estendere gli oggetti che possono essere raggiunti mediante il tasto di Tabulazione (Tab)** e sui quali l'utente avendo il focus può andare ad agire (es. attivare, selezionare, ecc.)

Navigazione da tastiera tra widgets

L'attributo **tabindex** permette al **focus di essere mosso tra gli elementi HTML interattivi via tastiera**

Quando un widget ha il focus da tastiera, **frecce, spazio, enter**, ecc. possono essere usati per **navigare** le opzioni del widget, **cambiare il suo stato, o innescare una funzione applicativa associata al widget**

Navigazione da tastiera tra widgets

Tab and Shift+Tab key: muovere il focus tra widgets e controlli standard HTML

* **Widgets with tabindex=0**: saranno inseriti nella sequenza TAB in base all'ordine nel documento

* **Widgets with tabindex>0**: saranno inseriti nella sequenza TAB in base al valore TABINDEX (in ordine crescente)

* **Widgets with tabindex<0**: non saranno inseriti nella sequenza TAB (ma possono ricevere il focus da tastiera via script)

Tecniche sufficienti

- 3.2.1 **Al Focus:** Quando qualsiasi componente riceve il focus, non deve avviare automaticamente un cambiamento di contesto
  - G107: Per **avviare automaticamente un cambio di contesto** usare "**activate**" piuttosto che "focus"
- 3.2.2 **All'Input:** Cambiare l'impostazione di qualsiasi componente dell'interfaccia utente non provoca automaticamente un cambiamento di contesto a meno che l'utente sia stato informato del comportamento prima di utilizzare il componente.
  - G80: Fornire un **tasto submit** per iniziare un **cambiamento di contesto**

Navigazione da tastiera dentro i widgets

JavaScript puo usare:

1. focus() method per muovere il focus all'elemento appropriato del widget

2. WAI-ARIA property: aria-activedescendant nel widget contenitore per indicare quale elemento nel widget deve avere il focus

2. più semplice e pulito!

Navigazione da tastiera dentro i widgets

Per la gestione del focus ad elementi figli dentro un widget (tree, grid, toolbar, ecc.) il parent può usare la proprietà **aria-activedescendant** per indicare il figlio attivo

L'elemento radice del widget deve avere un valore **TABINDEX maggiore o uguale a "0"** per assicurare che sia presente nell'ordine di TABBING del documento

Es. Toolbar

```
<div role="toolbar" tabindex="0" aria-activedescendant="button1"
    id="tb1" onkeydown="return optionKeyEvent(event);"
    onkeypress="return optionKeyEvent(event);">
<img src="buttoncut.png" alt="cut" role="button" id="button1">
<img src="buttoncopy.png" alt="copy" role="button" id="button2">
<img src="buttonpaste.png" alt="paste" role="button" id="button3">
 </div>
```

Definire la struttura di navigazione logica

- **Identificare la struttura logica della pagina**

  Dividendo la pagina in aree di blocchi percepibili contenenti informazioni semanticamente correlate dette "regioni". Se necessario frammentare ogni regione in ulteriori regioni logiche

- **Implementare la struttura logica del blocco nel markup** evidenziando gli elementi contenuti in ogni "region" come un o <iframe> con bordi visibili in modo che ogni regione o blocco sia percepibile dall'utente

Definire una struttura di navigazione logica

- **Etichettare ogni regione**

  L'inizio di ogni regione deve avere un header percepibile che deve essere utilizzato per etichettare la regione.

  Molto importante perchè senza una etichetta l'utente **non conoscerà la funzione/scopo della regione**; la lista delle regioni permette di avere un **sommario della pagina e di orientarsi meglio**

  ** Top News Stories **

Definire una struttura di navigazione logica

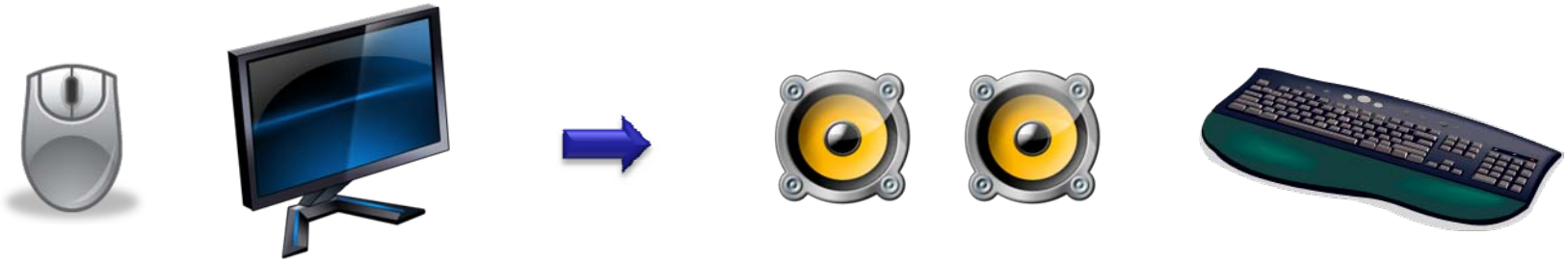
- **Assegnare un “landmark role” ad ogni regione** (application, banner, complementary, contentinfo, ecc. oppure definite dall’autore)

Screen Reader

Tecnologia assistiva che identifica e interpreta I contenuti a video

Output: voice synthesizer (generalmente)

Alternativa: Braille display (lento e costoso, usato raramente)



Le persone percepiscono i contenuti delle pagine ascoltandoli oralmente e navigando via tastiera => il processo di lettura può risultare lento, difficile e frustrante

Problemi introdotti dall'utilizzo di Screen reader per la navigazione Web

Content serialization
- time- and resource- consuming; losing layout/style information

Information overload
- thus user prefers to navigate via Tab key

Content and structure mixing
- difficulty processing page content => reading difficult

Lack of interface overview
- navigate for a long time without finding the most relevant content

Lack of context
- small portion of text => reiterate the reading process

Difficulty understanding UI elements
- links and labels should be context-independent and self-explanatory

Inability to fully access multimedia content

Cursore virtuale

JAWS for Windows: più diffuso screen reader in Italia

Presenta le pagine Web usando il **Cursore Virtuale**, che permette agli utenti di leggere e navigare una pagina Web come se fosse un documento testuale.

Arrow keys: leggere riga per riga, parola per parola, carattere per carattere.

Keys per la **navigazione veloce**, che permettono di muovere il cursore virtuale su links, headings ed elementi di controllo

TAB key: muoversi tra gli elementi della pagina che possono ricevere il focus

Cursore virtuale

Arrow keys o quick nav keys: **cambiano** la posizione del **Cursore Virtuale** ma **non cambiano** il punto attuale del **focus nell'applicazione**.

=> Anche se JAWS legge il testo di un link in una pagina Web non è detto che che vi sia il focus della tastiera.

Premendo TAB or SHIFT+TAB per navigare sposta il punto del focus e anche il Cursore Virtuale

Seconda parte

Esempi di interazione via screen reader

Esempi di problemi nell’utilizzo di RIA

Esempi di soluzioni tecniche

Qualche esempio

NOTA:

Alcuni degli esempi indicati sono prototipi e possono non essere sempre in linea: nel caso di problemi contattateci (vedere sezione riferimenti)

Esempio applicazione regions/landmarks personalizzate (solo con JAWS v. 11 o superiore):

http://whitemoon.iit.cnr.it/googleDocs

Esempio questionario senza uso di WAI-ARIA:

http://spreadsheets.google.com/formResponse?formkey=dFBWUGQ0LUVtOHZKaHdqelBRajNfaEE6MQ&theme=0AX42CRMsmRFbUy1mNGY3MzQyYi02MzhlLTQ5OTEtOTdiMS1jZjNmNGNiY2I0NGU&ifq

Esempio questionario con uso di WAI-ARIA:

http://whitemoon.iit.cnr.it:8080/user-test/Istruzioni-per-gli-acquisti-in-rete-strutturato/FineA.php

Qualche esempio

Wikipedia originale (come era nel 2009, in versione locale):

http://server2.iit.cnr.it/caterina/Progetto_Demo/wikipediaOR/index.php.htm

Wikipedia modificata applicando WAI-ARIA (stati, proprieta')

http://server2.iit.cnr.it/caterina/Progetto_Demo/WikipediaInterfacciaModificata_region5_ing.php

Esempio di definizione di live regions con WAI-ARIA

http://museodelmarmo.altervista.org/home.php

Materiale in rete

- **introduzione allo standard di Martin Kliehm** http://www.alistapart.com/articles/waiaria
- **esempi di widget javascript accessibili** http://wiki.codetalks.org/wiki/index.php/How_to_create_accessible_JavaScript_widgets
- **Università dell'Illinois**

  http://test.cita.uiuc.edu/aria/
- **documento di best practices per gli sviluppatori che si avviano alle prime realizzazioni**

  http://www.w3.org/TR/wai-aria-practices/

Materiale in rete

- **Possibilità di scaricare una versione di prova in italiano dello screen reader JAWS (valida per 40 minuti ad ogni avvio del computer):**

  http://www.subvisionmilano.com

- **Documentazione tecnica su JAWS:**

  http://www.freedomscientific.com/documentation/screen-readers.asp

Riferimenti

Maria Claudia Buzzi – IIT CNR

Claudia.Buzzi@iit.cnr.it

Barbara Leporini – ISTI CNR

Barbara.Leporini@isti.cnr.it

Commissione osservatorio siti Internet UIC (Unione Italiana Ciechi ed Ipovedenti) per la verifica dei contenuti web:

http://www.uiciechi.it/osi/

email: commissioneosi@uiciechi.it