How to create accessible SW

Zdeněk Míkovec
Dept. of computer graphics and interaction
Czech Technical University in Prague
What it means accessible?

- **Visual impairment**
  - control: keyboard navigation
  - presentation: audio/haptic; large graphics
  - data input: keyboard, gestures

- **Motor impairment**
  - control: large controls for direct manipulation (mouse, touch), keyboard navigation
  - presentation: <no limitations>
  - data input: keyboard, virtual keyboard, voice, direct manipulation via special devices (joystic, force feedback mouse)

- **Hearing impairment**
  - control: <no limitations>
  - presentation: visual/haptic
  - data input: not a speech
What it means accessible?

- What about combinations?
  - Deaf-blind impairment
  - Motor-blind impairment
  - Motor-deaf impairment

- Guidelines and recommendations
  - https://www.w3.org/WAI/
  - http://webaim.org/
Accessible RIA

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

https://www.w3.org/WAI/intro/wcag.php
https://www.w3.org/TR/WCAG20/
What is Rich Internet Application (RIA)?
- New approach on Internet
- Websites look more like desktop applications
  - Online documents
  - E-mail
  - Instant messaging
  - Calendar
  - Social networking

Number of RIA rapidly increasing

Possible benefits for handicapped people working with Internet
RIA +/-

- **RIA pros**
  - Higher comfort
    - Suggestions
    - Immediate feedback
  - Better user experience
    - Drag&Drop
    - Sophisticated components (tree, collapsible panel)
  - Dynamics
    - Chat
    - Shared data
    - Collaboration

- **RIA cons**
  - Not transparent
    - Many concurrent changes
    - Complex components
  - Unclear structure
    - Content (not in HTML)
    - Semantics in code of scripts
  - Problematic keyboard navigation
  - Difficult controlling of components
  - Dependency on JavaScript
WAI-ARIA

- RIA is less accessible than static web pages
  - How this can be solved?
- Web Accessibility Initiative (WAI)
  - part of W3C consortium
- Accessible Rich Internet Application suite of W3C (WAI-ARIA)
  - Recommendation since March 2014
  - How to make RIA accessible
    - Additional metadata
    - Bring back the semantics into HTML code
  - No restrictions on current functionality
a11y issues tackled by ARIA

- Keyboard-only operation impossible
- Orientation in content
  - Missing appropriate labels
  - Hiding of some content without announcement
    - Tab panels
    - Collapsible panels
- Controlling of UI components
  - Wrong or missing tab order
  - Focus stacking or disappearing
  - Wrong or missing status announcement
    - Checkbox
    - Radio Button
    - Slider
Creation of ARIA
Modern RIA applications are build from components

- Tree
- Tabs
- Accordion
- Grid
- etc.
Rich world of RIA

- Web environment is extremely variable
- Accessibility depends on:
  - Type of OS
    - Windows, Linux, Mac, …
  - Type of Web browser
    - Firefox, IE, Safari, Chrome, …
  - Type of Screen reader
    - Jaws, NVDA, Orca, …

- No configuration is 100% ARIA compliant
Three steps towards accessible RIA

1. Accessibility of RIA components
2. Accessibility of RIA applications
3. Testing of application accessibility
Three steps towards accessible RIA

1. Accessibility of RIA components
2. Accessibility of RIA applications
3. Testing of application accessibility
Offline component prototype
Offline component prototype

\[\text{CSS} \quad <\text{HTML}> \quad \text{JS} \quad + \quad <\text{ARIA}> \quad = \quad <\text{HTML}> \quad \text{JS} \quad <\text{ARIA}>\]

? \quad + \quad \text{Device} \quad \rightarrow \quad \text{Device} \quad \rightarrow \quad \text{Device}
Accessibility of RIA components

1. Create offline component prototype
2. Simplify the component architecture
3. Add WAI-ARIA attributes into offline component prototypes
   – Implementing ARIA attributes
   – Implementing keyboard navigation
4. Implement changes back to the server
   – Test whether results are accessible
Three steps towards accessible RIA

1. Accessibility of RIA components
2. **Accessibility of RIA applications**
3. Testing of application accessibility
Issues to be implemented

- Navigation on the page
- Relationships between components
- Dynamic changes of presented information

- Created set of 11 heuristics based on Nielsen’s usability heuristics
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. Distinguish all components
8. Define complete keyboard operation and where possible, standardize
9. Define document structure with ARIA landmarks
10. Provide a logical tab order
11. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind

2. Screen readers and another assistive technologies use several browsing modes. Make sure all parts of the web page are accessible at least with “virtual cursor” and “forms mode”. In forms mode all information in the form area must be linked to one of the form elements as a label or description.

3. Provide text alternative for all non-textual elements

4. Use headings to mark important areas

5. Handle hidden section appropriately

6. Communicate important information and feedback as soon as possible

7. Create proper linkage of controls, labels and messages

8. Distinguish all components

9. Define complete keyboard operation and where possible, standardize

10. Define document structure with ARIA landmarks

11. Provide a logical tab order

12. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. **Provide text alternative for all non-textual elements**
3. Icons and other similar visual elements that carry information to the user should have a textual alternative available. The only exception is when a non-textual element is used for decoration or layout purposes.
4. Communicate important information and feedback as soon as possible
5. Create proper linkage of controls, labels and messages
6. Distinguish all components
7. Define complete keyboard operation and where possible, standardize
8. Define document structure with ARIA landmarks
9. Provide a logical tab order
10. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. **Use headings to mark important areas**
4. Headings are the only elements with various levels of importance. They are often **used to scan the content** and should be used when possible to denote sections.
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. Distinguish all components
8. Define complete keyboard operation and where possible, standardize
9. Define document structure with ARIA landmarks
10. Provide a logical tab order
11. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. **Handle hidden section appropriately**
5. When showing larger section **move focus** to the section. When showing a **tooltip** all content should be **connected as description**.
6. Define complete keyboard operation and where possible, standardize
7. Define document structure with ARIA landmarks
8. Provide a logical tab order
9. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. **Communicate important information and feedback as soon as possible**
6. Use **on-the-fly validation** where possible. Use **live regions** to communicate asynchronous messages.
7. Define document structure with ARIA landmarks
8. Provide a logical tab order
9. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. **Connect** menus with corresponding *dynamically loaded sections* using aria-controls.
8. Define document structure with ARIA landmarks
9. Provide a logical tab order
10. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. **Distinguish all components**
8. **All components** that have their **Roles** identified in WAI-ARIA should be marked using appropriate Role.
9. 
10. Provide a logical tab order
11. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. Distinguish all components
8. Define complete keyboard operation and where possible, standardize

9. **Use design patterns** defined in WAI-ARIA or DHTML
10. Style Guide to determine the proper **keyboard navigation** before implementing your own.
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. Identify as many common structure parts as possible and apply WAI-ARIA landmark roles to them.
8. Define document structure with ARIA landmarks
9. Provide a logical tab order
10. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. **Menus** should be close in the means of tab order to the sections they are affecting. Tab order is important as it is used to quickly scan the page for interactive components. If the tab order is faulty, the mental model of the web page will likely be incorrect.
7. Create proper linkage of controls, labels and messages
8. Distinguish all components
9. Define complete keyboard operation and where possible, standardize
10. **Provide a logical tab order**
11. Use buttons for functions and links for linking
Heuristics

1. Design with screen reader modes in mind
2. Provide text alternative for all non-textual elements
3. Use headings to mark important areas
4. Handle hidden section appropriately
5. Communicate important information and feedback as soon as possible
6. Create proper linkage of controls, labels and messages
7. Make clear distinction between buttons and links. For all functions that are available on the page use buttons. For navigation purposes and for linking to other pages or anchoring, use links.
8. Define complete keyboard operation and where possible, standardize
9. Define document structure with ARIA landmarks
10. Provide a logical tab order
11. Use buttons for functions and links for linking
Three steps towards accessible RIA

1. Accessibility of RIA components
2. Accessibility of RIA applications
3. Testing of application accessibility
Testing of application accessibility

- Developer is typically NOT:
  - Blind user
  - Used to operate screen reader

- Need for accessibility testing with blind users

- Early stages of development means:
  - Poor accessibility
  - Need for support of accessibility testing
View of blind user

- User sees some components just partially or they seem missing
View of developer

- Developer sees all the components
View of user with description

Datepicker (E)
Tablist (A)
  Tab 1
    Collapsible panel (C)
      Panel 1
        Tree view (B)
      Panel 2
    Grid (D)
  Tab 2
    Grid
RIA accessibility - summary

- RIA accessibility is complicated and complex process
  - Has to be treated in phases

- Testing is complicated
  - Support of blind tester needed
Simulation and inspection tools
A11Y checkers

- Simulates color blindness and other visual impairments

- Web applications
  - http://webaim.org/resources/contrastchecker/
  - http://www.snook.ca/technical/colour_contrast/colour.html

- Firefox extensions
  - Web Developer, Juicy Studio Accessibility Toolbar
  - Color Checker, No Color, WCAG Contrast checker

- Chrome extensions
  - Accessibility Developer Tools
Screen readers

- **MS Windows**
  - NVDA
  - JAWS
- **Linux**
  - ORCA
- **Apple OS**
  - VoiceOver
- **Android**
  - TalkBack
- **Simulation in FireFox**
  - Fangs (screen reader emulation), Screen-reader-simulator
DIAS: Disability Impairment Approximation Tool

- Simulation of impairments
- Inspection of the code
- Integration with IDE (NetBeans)
- http://sourceforge.net/projects/diasnb/
WaaT: Web A11y assessment tool

- Assesses the a11y of web applications
- Integrated with IDE (NetBeans)

http://sourceforge.net/projects/waat/
MIS tool

- Simulates visual impairments
- Simulates mobile environment
Computer vs mobile environment

- **Computer**
  - Keyboard, mouse
  - Large LCD monitor
  - Steady environment (indoors)
    - Artificial lightning
    - Fixed position
    - Planned activity

- **Mobile**
  - Touch screen, HW buttons
  - Small display
  - Changing environment (outdoor)
    - Sun, darkness
    - Movements
    - Frequent unpredictable interruption
Simulation of mobile environment in office?

- More factors should be taken into account
- Combination of factors also important
- Difficult to simulate in office environment

Hard to imagine

Simulation desired
Mobile Impairment Simulation tool

- Filter overlay window
- Independent on mobile platform
  - Android
  - BlackBerry
  - iOS
  - Symbian
  - Windows 7
  - ...
Categories of simulated issues

- **Visual impairment**
  - Tunnel vision
  - Blurred vision
  - Color blindness

- **Occlusion of the display**
  - Finger occlusion

- **Reflection on the display**
  - Static reflections
  - Display tremor

- **Combined simulations**
Visual impairment

- Blurred vision
- Tunnel vision
- Color blindness
Occlusion of the display

Real world

Simulation
Reflection on the display

Real world

Simulation
Homework 1: Bad designs

- Check examples of bad and good designs
  - http://www.baddesigns.com/
  - Bad vs. Good Design
    - https://www.interaction-design.org/literature/article/bad-design-vs-good-design-5-examples-we-can-learn-frombad-design-vs-good-design-5-examples-we-can-learn-from-130706

- Check UI design guidelines
  - ARIA design (earlier in this presentation)
  - Nielsen’s 10 rules of thumb

- Write a composition on
  - what are the reasons leading to bad design
  - document your own experience with bad design
    - problem description, photos, recommendations for improvements
Homework 2: A11Y web inspection

- **Install a11y tools**
  - Firefox/Chrome extensions (slide 41)
  - Screen readers (slide 42)

- **Check several webs**

- **Try to improve web accessibility**
  - use extensions to edit HTML and CSS on the client side

- **Write a report on accessibility of selected web**
  - what are the issues (readability, control, navigation)
  - recommendations for improvements
Thank you

Zdeněk Míkovec
Czech Technical University in Prague
xmikovec@fel.cvut.cz