ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Accessibility crash course!

Vincenzo Rubano

Usability and User Experience Design

Università di Bologna,
A.Y. 2023-2024.

Outline

- Brief intro to disabilities and assistive technologies
- Why accessibility?
- International standards and guidelines
- Accessibility testing

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Accessibility

What is web accessibility?

Disability types

Visual: visual impairments including blindness, various common types of low vision, poor eyesight, and color blindness.

Motor/mobility: such as difficulty or inability to use the hands, including tremors, muscle slowness, loss of fine muscle control, etc.

Auditory: deafness or hearing impairments, including individuals who are hard of hearing.

Seizures: photo epileptic seizures caused by visual strobe or flashing effects.

Cognitive and intellectual: Developmental disabilities, learning difficulties (dyslexia, dyscalculia, etc.), and cognitive disabilities (PTSD, Alzheimer's) of various origins, affecting memory, attention, developmental "maturity", problem-solving and logic skills, and so on.

Accessibility is much more!

But accessibility does not benefit only people listed in the previous slide, it extends to anyone who is experiencing any permanent, temporary or situational disability.

Temporary disability: a broken wrist makes mouse navigation not an option.

By situational disability we mean someone who may be experiencing a boundary based on the current experience (e.g. partial sight due to sun lighting, being one handed due to carrying a baby).

Why should you care?

- social and ethics
- legal reasons
- business

Social implications

Accessibility is a Civil Right, recognized by the United Nations Convention on the Rights of Persons with Disabilities (CRPD).

Any website, application or system that is not accessible can be considered a discrimination, as it prevents groups of people from using it.

Inaccessible systems impacts negatively on dignity, autonomy, full and effective participation, equal opportunity, and much more of large groups.

Legal reasons

Accessibility laws and polices are in place to enforce the creation of accessible content all over the world (Australia, US, Canada, European Union, Italy, and more).

Companies with inaccessible websites and/or applications can be (and are) sued for that (US, EU coming soon, eventually).

In Italy, websites and mobile applications developed on behalf of public administrations, or companies that provide services on their behalf, have to be accessible. Bodies in the public field (including schools, museums, universities, etc) cannot purchase inaccessible ICT solutions.

Business

It has been estimated that The total after-tax disposable income for working-age people with disabilities in the US is about $490 billion, which is similar to that of other significant market segments, such as African Americans ($501 billion) and Hispanics ($582 billion). Simply put, inaccessible systems are missing on a significant market segment.

People with disabilities are not a solitary market; as they are surrounded by family members and friends who also recognize the value in products and services that accommodate all people in society.

Getting sued for accessibility reasons costs money, and you'll have to pay for accessibility remediation in any case.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Assistive Technologies

I/O assumptions

When we design computer systems we often make several assumptions:

- interface will be driven by mouse clicks;
- keyboard will be used for text input (only);
- output will be sent via the screen.

Is this really the case?

Multimodal I/O

Of course not. Users might be interacting with a system in completely unexpected ways, leveraging extremely different input and output devices and systems.

A system should be designed and implemented to behave correctly in such scenarios, offering to anyone the same user experience. Or, in other words, to let everyone access it!

Assistive technologies are such an example of multimodal I/O.

Assistive technologies

Assistive technologies (AT) [1] are assistive, adaptive, and rehabilitative devices for people with disabilities or the elderly population.

People who have disabilities often cannot perform activities of daily living (ADL) such as toileting, mobility (ambulation), eating, bathing, dressing, grooming, and personal device care as you usually do.

Assistive technologies can ameliorate the effects of disabilities that limit the ability to perform ADLs and promote greater independence by enabling people to perform tasks they were formerly unable to accomplish, or had great difficulty accomplishing, by providing enhancements to, or changing methods of interacting with, the technology needed to accomplish such tasks.

[Source: Assistive technology | Wikipedia]

Assistive technologies: some examples

- white canes and/or guide dogs, that allow blind people to move independently in their surroundings avoiding obstacles;
- wheelchairs, to provide independent mobility for those who cannot walk,;
- assistive eating devices, that can enable people who cannot feed themselves to do so;
- hearing AIDs, devices designed to make sounds audible to a person with hearing loss;

Assistive technologies for the visually impaired

Let’s examine some assistive technologies that can be used by blind and visually impaired people to interact with computer systems, illustrating how they can be seen as different I/O means from a usability and user experience point of view. How do they change the perception of a system?

Main visual impairment ATs:

- refreshable Braille display;
- screen reader;
- screen magnifier;

Refreshable Braille display

A refreshable braille display or braille terminal is an electro-mechanical device for displaying Braille characters, usually by means of round-tipped pins raised through holes in a flat surface. Each area for displaying a character is called a “cell”: typical Braille displays contain 40 or 80 cells. Each cell can use up to 8 points for representing a character.

Let’s see it in action!

Screen reader

A screen reader is a software application that attempts to convey what people with normal eyesight see on a display to their users via non-visual means, like text-to-speech, sound icons or a Braille terminal.

Let’s see it in action!

Main screen readers

Name	Operating system	License
Jaws	Windows	Commercial
NVDA	Windows	GPL v 2
Orca	Linux, ambiente grafico	GPL
TalkBack	Android	Built-in
VoiceOver	iOS	Built-in
VoiceOver	Mac OS	Built-in
Chrome Vox	Google Chrome, Chrome OS	N/A

Screen magnifier

A screen magnifier interfaces with a computer's graphical output to present enlarged screen content. By enlarging part (or all) of a screen, people with visual impairments (with some functional vision) can better see words and images.

The simplest form of magnification presents an enlarged portion of the original screen content, the focus, so that it covers some or all of the full screen. This enlarged portion should include the content of interest to the user and the pointer or cursor, also suitably enlarged. As the user moves the pointer or cursor the screen magnifier should update the enlarged content. If this tracking is jerky or flickers it is likely to disturb the user.

Common features in screen magnifiers

Ranges of 1- to 16-times magnification are commonly used. The greater the magnification the smaller the portion of the original screen content that can be viewed, so users will tend to use the lowest magnification they can manage. Additional features are commonly provided for people with particular sight difficulties:

- Color inversion, typically turning text from black-on-white to white-on-black. This can reduce screen glare.
- Smoothing. Text can become blocky and harder to recognize when enlarged, thus screen magnifiers use interpolation to smooth the text to compensate.
- Cursor customization, highlighting mouse and text cursor positions to make them more visible.
- Different magnification modes. Screen magnifiers can alter how they present the enlarged portion: covering the full screen, providing a lens that is moved around the un-magnified screen, or using a fixed magnified portion.
- Crosshairs (with customizable size, color and opacity), to make the use of a pointing device easier when the mouse pointer is hard to see even if using magnification.

Assistive technologies for motor impairments

Motor impairment assistive technologies:

- mouth stick;
- head wand;
- single switch access;
- sip and puff switch;
- oversized trackball mouse;
- adaptive keyboard;
- eye tracking systems;
- voice recognition systems;

Mouth stick

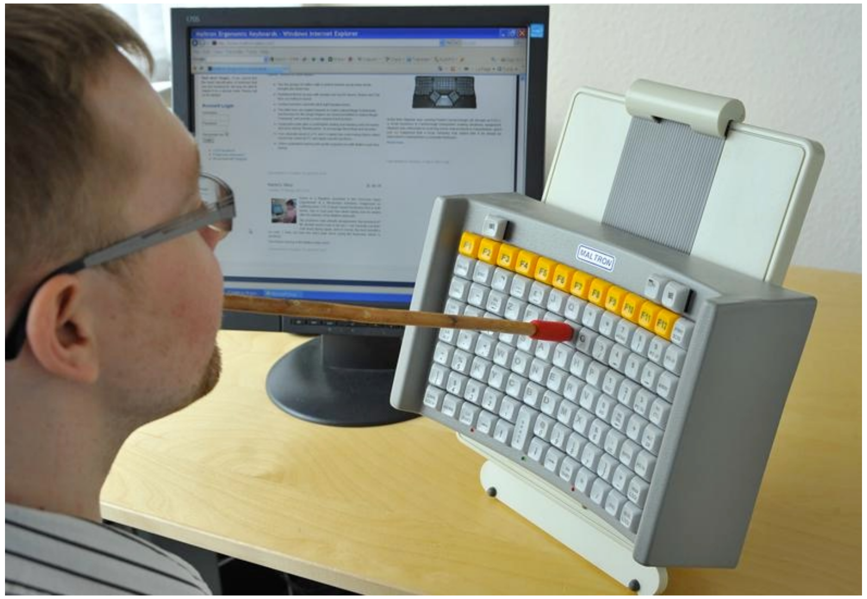
A mouth stick is just what its name implies: a stick that is placed in the mouth.

Due to its simplicity and low cost, the mouth stick is one of the most popular assistive technologies.

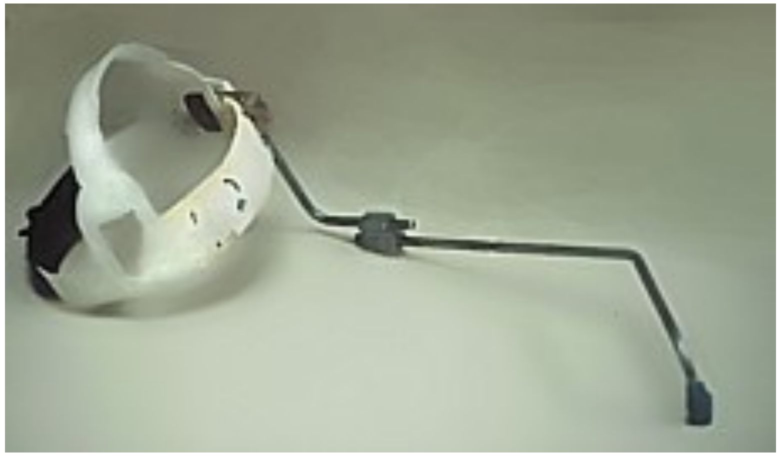
In many cases there is a rubber tip at the end of the mouth stick to give the tip better traction, and a plastic or rubber feature at the other end that the person inserts into the mouth.

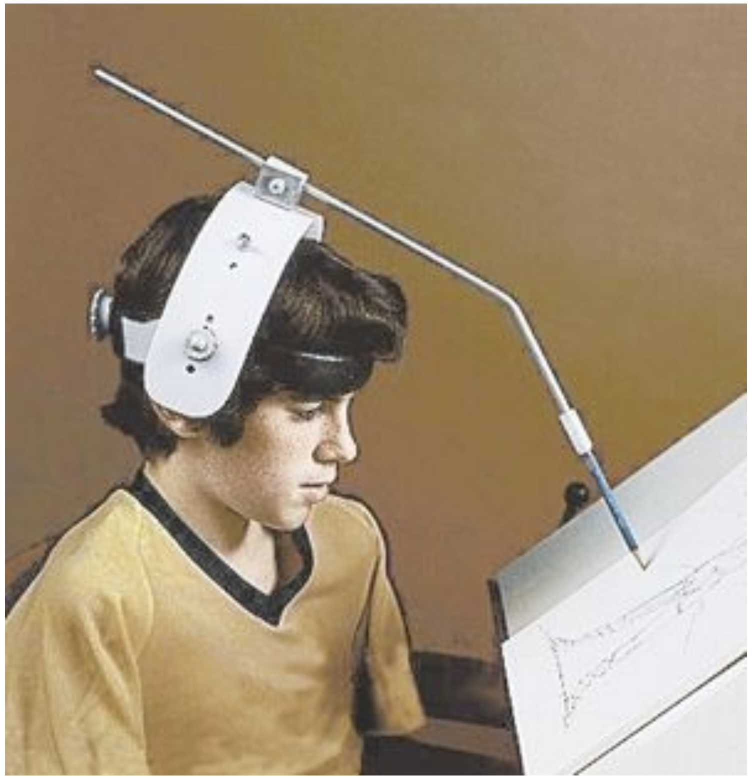
Someone with no use of the hands could use a mouth stick to type and perhaps to manipulate a trackball mouse, depending on the amount of control that the person has with the mouth stick, and on the amount of patience that the person has if these movements are difficult.

[Source: Motor disabilities assistive technologies | WebAIM]



Head wand





Very similar in function to mouth sticks, except the stick is strapped to the head.

A person moves the head to make the head wand type characters, navigate through web documents, etc.

Fatigue can be an issue when a lot of keystrokes are required in order to accomplish a task.

[Source: Motor disabilities assistive technologies | WebAIM]

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Single switch access

People who have very limited mobility use this type of device.

If a person can move only the head, for example, a switch could be placed to the side of the head that would allow the person to click it with head movements.

This clicking action is usually interpreted by special software on the computer, allowing the user to navigate through the operating system, web pages and other environments.

Some software facilitate the typing of words by using an auto-complete feature that tries to guess what the person is typing, and allowing the person to choose between the words that it guesses.

Source: [5]

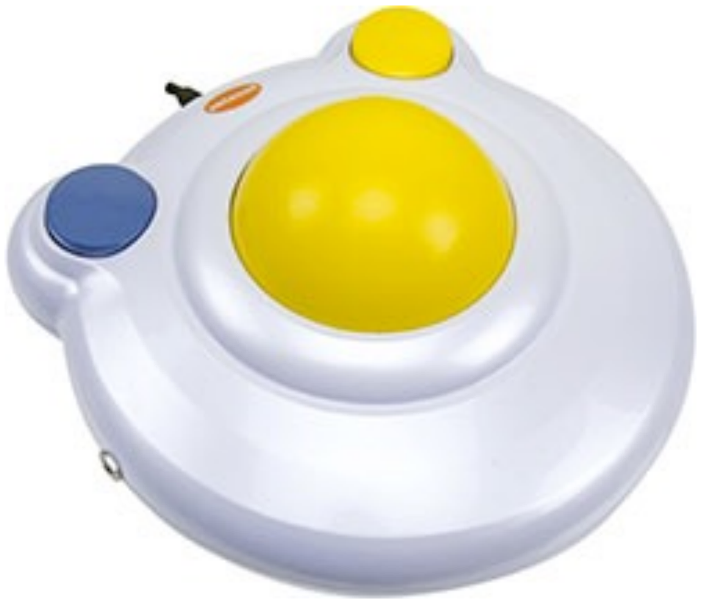
Sip and puff switches

Similar in functionality to the single switch, sip and puff switches are able to interpret the user's breath actions as on/off signals, and can be used for a variety of purposes, from controlling a wheelchair to navigating a computer.

The hardware can be combined with software that extends the functionality of this simple device for more sophisticated applications.

[Source: Motor disabilities assistive technologies | WebAIM]

Oversized trackball mouse

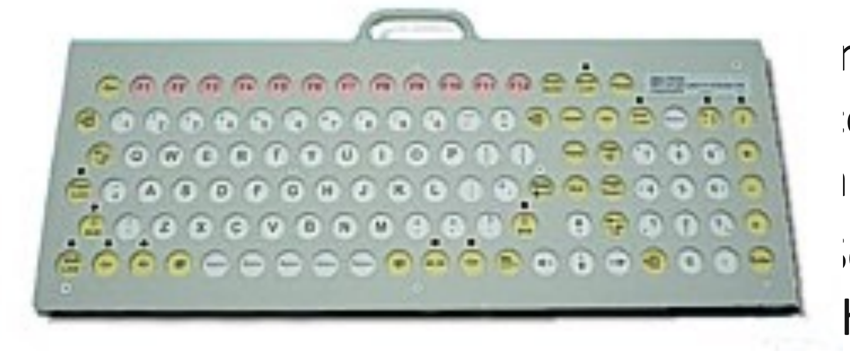


A trackball mouse is not necessarily an assistive technology, as some people without disabilities simply prefer it to the standard mouse. But it is often easier for a person with a motor disability to operate than a standard mouse. Someone may, for example, use a trackball mouse in conjunction with a head wand or mouth stick, as it is much easier to manipulate a trackball with these devices compared with a standard mouse.

Someone with tremors in the hands may also find this kind of mouse more useful because once the person moves the mouse cursor to the right location, there is less danger of accidentally moving the cursor while trying to click on the mouse button. A person with tremors in the hands could also manipulate the trackball mouse with a foot, if there is enough motor control in the feet.

[Source: Motor disabilities assistive technologies | WebAIM]

Adaptive keyboard



n cases where a person does not have reliable muscle ontrol in the hands for precision movements, an daptive keyboard can be useful.

ome adaptive keyboards have raised areas in between he keys, rather than lowered areas, to allow the person to first place the hand down on the keyboard, then slide the finger into the correct key.

Keyboard overlays are also available as an adaptation to standard keyboards, which achieve the same results. In some cases, adaptive keyboards come with specialized software with word-completion technology, allowing the person to type with fewer keystrokes, since typing can be rather laborious and slow otherwise.

Source: [5]

Eye tracking systems

Eye tracking devices can be a powerful alternative for individuals with no control, or only limited control, over their hand movements.

The device follows the movement of the eyes and allows the person to navigate through the web with only eye movements.

Special software allows the person to type, and may include word-completion technology to speed up the process.

These systems can be expensive—usually in the thousands of US dollars—so they are less common than the less sophisticated devices, such as mouth sticks and head wands.

[Source: Motor disabilities assistive technologies | WebAIM]

Voice recognition systems

These systems allow a person to control the computer by speaking. This assumes that the person has a voice that is easy to understand. Some people with motor disabilities—those with cerebral palsy in particular—may have a difficult time speaking in a way that the software can understand them, since the muscles that control the voice are slow to respond, and speech is often slurred, despite the fact that these people do not have any slowness in their mental capacity.

[Source: Motor disabilities assistive technologies | WebAIM]

Let’s see them in action!

How do these systems work?

Most of the assistive technologies we examined work through or emulating the keyboard. This implies that it is critical for a system to be accessible to the keyboard and navigable with as few keystrokes as possible. But that's only a (good and essential) starting point to support all users with disabilities!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Welcome to the accessibility world

Towards official guidelines

In order to guarantee that a system can be used by everyone, independently from the assistive technology he/she needs, more complex guidelines need to be introduced.

Given the variety of technologies we can use nowadays to implement new systems, such guidelines should be abstract enough to be valid for each of them, but still be concrete so as to make it possible implementing such recommendations.

WCAG 2.2

Web Content Accessibility Guidelinhes (WCAG) 2.2 is a W3C recommendation that contains a set of guidelines to be satisfied by each system to be considered accessible. Such guidelines are organized around 4 fundamental principles. For each guideline, success criteria (testable statements) are provided, specifying what to test and the expected results yet in a technology independent way.

Conformance to WCAG 2.1 can be in three different levels (A, AA, AAA) depending on what success criteria the system satisfies.

Note that additional support documents (Techniques for WCAG 2.1) are provided to offer practical examples on how to meet success criteria in specific, technology dependent ways.

WCAG 2.2 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 2.2 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]

Web accessibility by examples I

Coding a site with semantically meaningful HTML, textual equivalents provided for images and links named meaningfully, helps blind users using screen readers and Braille displays.

Large and/or enlargeable text and images make easier for users with poor sight to read and understand the content.

Having colored and underlined or otherwise differentiated links ensures that color blind users will be able to notice them.

Large clickable links and areas help users who cannot control a mouse with precision (or use the website with a touch screen device).

Web accessibility by examples II

Not coding in a way that hinders navigation by means of the keyboard alone, or a single switch access device alone, helps users who cannot use a mouse or even a standard keyboard.

Providing closed captioned videos, a transcript and/or a sign language version of them, deaf and hard-of-hearing users can understand it.

When flashing effects are avoided or made optional, users prone to seizures caused by these effects are not put at risk.

Writing content in plain language and illustrating it with instructional diagrams and animations, can make users with dyslexia and learning difficulties understand it better.

A multi step process

By definition, it is clear that, for a system to be accessible, multiple phases of its life cycle are involved:

- *design*, as important decisions have to be made even before writing the first line of code; it’s much simpler to make an easy-to-use interface accessible rather than a complex one;
- *development*, as implementing the design (i.e. coding) can introduce accessibility issues
- *editing*, as content within the system should be accessible, or your efforts (design and implementation) are vanished.

Design

Every design decision has the potential to include or exclude customers. Inclusive design emphasizes the contribution that understanding user diversity makes to informing these decisions, and thus to including as many people as possible. User diversity covers variation in capabilities, needs and aspirations.

Source:What is Inclusive Design

Development

Code chicks in. The implementation phase can introduce barriers as well. The design must be implemented leveraging existing technologies known to be accessible, or adopting all mechanisms required to make them so.

Support documents explaining how to comply with WCAG 2.1 in specific scenarios are available, e.g.Techniques for WCAG 2.1

Content

You could have the most accessible system, but your efforts vanish when content is not authored (edited) to be accessible. Examples include:

- attaching inaccessible documents (scanned PDF files without OCR),
- screenshots without descriptions that can make you understand their content,
- writing texts that cannot be understood by everyone (e.g. using information that can be related only to one sense),
- not providing enough context for content to be understood in case of a disability.

WCAG 2.2 conformance levels

Levels of compliance to WCAG 2.2:

A

- lowest level of conformance;
- removes major barriers for blindness, deafness and motor disabilities.

AA

- next level of conformance (includes A);
- removes major barriers for low vision users;
- offers a little help for cognitive disabilities.

AAA

- highest level of conformance (includes A and AA);
- not recommended to be required as a general policy for entire sites because it is not possible to satisfy all Level AAA requirements for some content.

How do we test for conformance?

Testing for conformance to accessibility guidelines can be automated to some extent, but still requires manual user testing to be fully assessed.

Let’s consider a simple success criterion: all non-decorative images should have a descriptive alternative text. Checking that an image has an alternative text associated to it is trivial, but ensuring that it is descriptive for that image is not (yet). Also distinguishing what images are decorative and what not can be complicated, even for humans.

Myths and facts I

Myth: an accessible interface is ugly and boring.

Fact: you can implement sophisticated and beautifully crafted interfaces, yet accessible! An accessible design is more useable, but that’s something for another topic!

Myth: accessibility is expensive!

Fact: yes, but only if you consider it as an afterthought. Remediating inaccessible designs require much more efforts, time and knowledge (thus money) than creating an accessible equivalent of it, and the end result might be (generally speaking is) not as good as it could.

Myths and facts II

Myth: accessibility benefits too few people.

Fact: it is estimated that around 10% of the population worldwide has a disability that affects internet usage. Are about 700 million people too few? And you need to add to the number people affected by temporary and situational disabilities! And like it or not, with age our hearing, sight and dexterity diminish, changing our ability to use the Internet.

Myth: accessible interfaces are static.

Fact: highly dynamic and sofisticated websites (even desktop like applications) can be made accessible, just special attention is required. Welcome to WAI-ARIA!

WAI-ARIA

Accessible Rich Internet Applications (WAI-ARIA) 1.1 is a W3C recommendation that provides an ontology of roles, states, and properties that define accessible user interface elements and can be used to improve the accessibility and interoperability of web content and applications. Designed to allow an author to properly convey user interface behaviors and structural information to assistive technologies in document-level markup.

It is a critical tool for making accessible desktop-like web applications, as there are (many) advanced widgets (menu-bars, tabs and tab panels, toolbars, etc) that are not part of HTML (yet).

Role, properties and states

You can use WAI-ARIA by leveraging specific attributes to be applied on any HTML element:

- the *role* attribute, that specifies the role (semantics for) the element (button, checkbox, tree, tablist, tab, etc);
- properties (*aria-label, aria-labelledby, aria-valuenow*, etc), attributes that are essential to the nature of a given object, or that represent a data value associated with it. A change of a property may significantly impact the meaning or presentation of an object;
- states (*aria-checked, aria-selected*, etc), dynamic properties expressing characteristics of an object that may change in response to user action or automated processes. States do not affect the essential nature of the object, but represent data associated with the object or user interaction possibilities.

Rules of ARIA

1. If you can use a native HTML element] or attribute with the semantics and behavior you require already built in, use that. Exceptions:
   - if the feature is available in HTML but it is not implemented or its implementation does not provide accessibility support;
   - If the visual design constraints rule out the use of a particular native element, because the element cannot be styled as required.
2. Do not change native semantics, unless you really have to. Note that if a non-interactive element (e.g. span) is used as an interactive one (e.g. button), the developer must implement the appropriated behavior using JavaScript.
3. faAll interactive ARIA controls must be usable with the keyboard. Support should be implemented by the developer.
4. Do not use role="presentation" or aria-hidden="true" on a focusable element , or focus might end up in the middle of nowhere.
5. All interactive elements must have an accessible name.

ATAG 2.0

Authoring tools accessibility guidelines (ATAG) 2.0 is a W3C recommendation specifically crafted for ensuring accessibility of authorhing tools such as:

- web page authoring tools (i.e. WYSIWYg HTML editors);
- software for generating websites (i.e. CMS systems);
- software that converts contents to web technologies;
- multimedia authoring tools;
- websites whose users can add content (i.e. social networks).

ATAG 2.0 II

ATAG 2.0 is divided in two main parts:

- part a, that is about making authoring tools accessible so that people with disabilities can use them;
- part b, that is about helping authors produce accessible content, i.e. content conforming to WCAG 2.1.

Like in WCAG 2.1, in ATAG we find guidelines organized around key principles, whose satisfaction can be assessed by success criteria compliance on same 3 levels (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.

(very) helpful resources

Shamelessly self-advertising A11a... A structured, cathegorized collection of accessibility resources available on the Internet. You can find it at https://a11a.disi.uhnibo.it

Questions?

???

Tips and tricks

Here are some tips and tricks that can be useful to implement your project, and create accessible web applications in general.

Tip I: mistrust the authority

If you are using a framework of UI components (bootstrap, angular-material, element-ui, etc), do not assume that those components will be accessible. Always verify their accessibility, and eventually work around their issues; contributing fixes to project is recommended, but it’s up to you! Choose a different framework if necessary.

Tip II: test with a screen reader

There’s a strong evidence that browsing a web application and interacting with it by means of accessibility, offers the most thorough accessibility review. It does not cover any aspect, but it’s a good starting point!

Tip III: pick an easy to use screen reader for testing

If you decide to test your web application with a screen reader, make sure to know how to use it (main features, keyboard shortcuts, etc). Seems obvious but, *absolutely* be sure to know how to disable it: screen readers often change the way a computer is controlled, thus can be considered invasive; make sure you know how to control them. Chrome Vox is a great option on that point, as it offers a great introductory interactive tutorial and is a browser extension.

Tip III: automated testing

Always review issues reported by automated tools, as in some cases they might not be actual errors. Distinguish between issues reported as errors, and issues that are reported as potential errors (they could or couldn't be, but the automated tools could not infer an answer). When uncertain focus on errors!

Accessible design cheatsheet

How should you design an interface to maximise the chances of it being accessible? Let’s see.

Distinguish design patterns and widgets

Identify design patterns required to visualize the data, and widgets to represent, input or otherwise interact with it. Try to compose your interface with as few widgets as possible, be consistent.

Let “Element” be each necessary widget or pattern.

Native elements

Is Element available as a native interface element on the platform your interface will be executed on?

Great, use it… Do not try to emulate it, unless you have a very very very good reason to do so (probably you don’t). Let e be such interface element.

Does e require specific information to be accessible? You should be able to answer this question with a good understanding of WCAG 2.1 principles and guidelines, but you could also find out by looking at “Techniques for WCAG”. Rule of thumb: if it is an interactive element (e.g. form widget), it does. Provide such information so that it is meaningful.

Is e an image?

If e is a static image (i.e. interacting with it does not initiate any action that alters the application state), ask yourself: “is e a decorative image?”

If so, convey such information to assistive technologies, its description is not important.

Otherwise, make sure a meaningful description is associated to it. Test: prevent your user agent from displaying images. Can you make sense of what’s shown in those pictures by relying on their description?

Focus handling

The more dynamic and sophisticated your application is, the more focus handling importance increases! Whenever interface state changes occur ask yourself:

1. Where is the focus?
2. Where should it be?

Incorrect or missing focus handling causes assistive technology users to be disoriented (a modal dialog is missed, but focus is not placed on the element that triggered its opening), and UI state changes to be unnoticed (e.g. a modal dialog appears, but focus is not moved to its first focusable child).

Moving focus to an element has the potential for its preceding siblings to end up being unnoticed by assistive technology users, so choose wisely when to do that. Rule of thumb: no autofocus on appearance of a page/view/screen, unless required by its design pattern.

About external frameworks

Realistically, chances are that Element is a component provided by an external framework, or an HTML element enhanced by that. You still need to ensure it is accessible, and fix or workaround its accessibility issues in your code. Pick up a framework known for being a good starting point in this regard. In any case, expect to do some work on this front. Remember, you’re always responsible for whatever you deliver.

About styling

Feel free to style your elements as you desire, but keep in mind accessibility principles. Pay special attention to color contrast, font sizes and make your layout as responsive as possible to respond to font size changes, zooming, etc. Choose fonts that make content more readable (also keeping in mind the context). Keep in mind readability rules, they’re important for accessibility too!

Automated testing

Automated accessibility testing tools can help you identify accessibility issues, use them! Be careful to choose reliable ones.

Make sure to run such tools for each variation in your interface (e.g. when a form trigger errors or doesn’t, a modal is presented or not, a menu is expanded or collapsed, etc).

Manual testing I

Manual accessibility testing is essential.

Can your application be fully operated by using only the keyboard? Check for that!

Try using a screen reader to interact with your application: does the navigation flow make sense? Is your interface operable, understandable and perceivable? Are state changes to it, both manually or automatically initiated, too?

Screen readers are available for most platforms, make sure you fully understand their features and how to use them before testing your interface: this ensures you won’t consider as accessibility issues problems that depends on the fact you don’t know how to use the assistive technology properly.

Manual testing II

Pay attention to identify information conveyed only by colors, even if hopefully at this point you shouldn’t have any. If found, iterate this process on that particular case to find an accessible representation.

Try enlarging font sizes (2x up to 4x at least). Does your interface scale nicely to accomodate for this change?

Top priorities

- Keyboard support
- Form labels
- Focus handling
- Text alternatives
- Color contrast and font sizes

Don’t forget about content

As we said, design and development are just two of the major three components involved in making a system accessible.

People with disabilities can use technology, but with adaptations (assistive technologies).

Do not require actions that a disabled person cannot perform (i.e. reach up for something located in a high position for a person with a wheelchair, distinguish in between colors for blind people, etc).

Keep in mind WCAG 2.1 principles

Questions?

???

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA