Accessibility

MIS – SS 2023

OLD video

Accessibility

- Almost one in five of the world's population lives with some kind of recognized disability.
- Sooner or later, everyone will develop at least some limitations in vision, hearing, dexterity or learning. To improve usability for those of us with sensory or physical limitations, phones and tablets have features for accessibility, which are continually improving and becoming more prevalent as technologies advance.

https://webaim.org/intro/index#video

Accessibility

Important

The major categories of disability types are:

 **Visual**

Blindness, low vision, color-blindness

 **Hearing**

Deafness and hard-of-hearing

 **Motor**

Inability to use a mouse, slow response time, limited fine motor control

 **Cognitive**

Learning disabilities, distractibility, inability to remember or focus on large amounts of information

More than “Mobile”

- “Mobile accessibility” refers to making websites and applications more accessible to people with disabilities when they are using mobile phones and other devices.
- WAI’s work in this area addresses accessibility issues of people using a broad range of devices to interact with the web, including:
  - phones and tablets
  - digital TVs
  - wearables such as smart watches
  - devices in car dashboards and airplane seatbacks
  - devices in household appliances
  - other “Internet of Things”
- It address a wide range of issues:
  - touchscreens
  - small screen sizes
  - different input modalities, including speech and 3D touch enabled by pressure sensors
  - device use in a different settings, such as bright sunlight
  - and more

WAI

- The W3C Web Accessibility Initiative (WAI) develops standards and support materials to help you understand and implement accessibility.
- https://www.w3.org/WAI/

WAI's accessibility standards address mobile accessibility

- **Web Content** Accessibility Guidelines (WCAG) covers web pages and web applications, including content used on mobile devices.
- **User Agent** Accessibility Guidelines (UAAG) covers web browsers and other "user agents", including mobile browsers.
- **Authoring Tool** Accessibility Guideline (ATAG) covers software used to create web pages and applications, including for mobile.
- WAI-ARIA (Accessible Rich Internet Applications) defines ways to make web content more accessible, especially dynamic content and advanced user interface controls. It applies to web applications and to accessing websites with mobile devices.

WCAG

- WCAG 2.0 was published on 11 December 2008. WCAG 2.1 was published on 5 June 2018.
- WCAG is primarily intended for:
  - Web content developers (page authors, site designers, etc.)
  - Web authoring tool developers
  - Web accessibility evaluation tool developers
  - Others who want or need a standard for web accessibility, including for mobile accessibility

WCAG: Four Principles of Accessibility

- 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)
- 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)

If any of these are not true, users with disabilities will not be able to use the Web.

https://www.w3.org/WAI/WCAG21/Understanding/

Perceivable

- Guideline 1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.
- Guideline 1.2 Time-based Media: Provide alternatives for time-based media.
- Guideline 1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
- Guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.

Operable

- Guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard.
- Guideline 2.2 Enough Time: Provide users enough time to read and use content.
- Guideline 2.3 Seizures and Physical Reactions: Do not design content in a way that is known to cause seizures or physical reactions.
- Guideline 2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are.
- Guideline 2.5 Input Modalities: Make it easier for users to operate functionality through various inputs beyond keyboard.

Understandable

- Guideline 3.1 Readable: Make text content readable and understandable.
- Guideline 3.2 Predictable: Make Web pages appear and operate in predictable ways.
- Guideline 3.3 Input Assistance: Help users avoid and correct mistakes.

Robust

- Guideline 4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies.

ATAG

- Authoring tools are software and services that “authors” (web developers, designers, writers, etc.) use to produce web content (static web pages, dynamic web applications, etc.). Examples of authoring tools are listed below under “Who ATAG is for”.
- The Authoring Tool Accessibility Guidelines (ATAG) documents explain how to:
  - make the authoring tools themselves accessible, so that people with disabilities can create web content, ***and***
  - help authors create more accessible web content — specifically: enable, support, and promote the production of content that conforms to Web Content Accessibility Guidelines (WCAG).

UAAG

- The User Agent Accessibility Guidelines (UAAG) documents explain how to make user agents accessible to people with disabilities. User agents include browsers, browser extensions, media players, readers and other applications that render web content. Some accessibility needs are better met in the browser than in the web content, such as text customization, preferences, and user interface accessibility. A user agent that follows UAAG 2.0 will improve accessibility through its own user interface and its ability to communicate with other technologies, including assistive technologies (software that some people with disabilities use to meet their requirements). All users, not just users with disabilities, will benefit from user agents that follow UAAG 2.0.

WAI-ARIA

- **WAI-ARIA, the Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. It especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.** Currently certain functionality used in Web sites is not available to some users with disabilities, especially people who rely on screen readers and people who cannot use a mouse. WAI-ARIA addresses these accessibility challenges, for example, by defining new ways for functionality to be provided to assistive technology. With WAI-ARIA, developers can make advanced Web applications accessible and usable to people with disabilities.

WAI-ARIA Example

- Web sites are increasingly using more advanced and complex user interface controls, such as tree controls for Web site navigation like the example in Figure 1. To provide an accessible user experience to people with disabilities, assistive technologies need to be able to interact with these controls. However, the information that the assistive technologies need is not available with most current Web technologies.
- Another example of an accessibility barrier is drag-and-drop functionality that is not available to users who use a keyboard only and cannot use a mouse. Even relatively simple Web sites can be difficult if they require an extensive amount of keystrokes to navigate with only a keyboard.
- Many Web applications developed with Ajax (also known as AJAX), DHTML, and other technologies pose additional accessibility challenges. For example, if the content of a Web page changes in response to user actions or time- or event-based updates, that new content may not be available to some people, such as people who are blind or people with cognitive disabilities who use a screen reader.

WAI-ARIA Solutions

- WAI-ARIA provides Web authors with the following:
  - Roles to describe the type of widget presented, such as “menu”, “treeitem”, “slider”, and “progressmeter”
  - Roles to describe the structure of the Web page, such as headings, regions, and tables (grids)
  - Properties to describe the state widgets are in, such as “checked” for a check box, or “haspopup” for a menu.
  - Properties to define live regions of a page that are likely to get updates (such as stock quotes), as well as an interruption policy for those updates—for example, critical updates may be presented in an alert dialog box, and incidental updates occur within the page
  - Properties for drag-and-drop that describe drag sources and drop targets
  - A way to provide keyboard navigation for the Web objects and events, such as those mentioned above

Personalization

- Personalization involves tailoring the user experience to meet the needs and preferences of the individual user.
- As web technology broadens in scope and reach, and understanding of users' needs increases through research and experience, the set of accessibility considerations for websites becomes larger.
  - Rather than developing complex solutions for a wide range of users, designing sites in a way that they can be personalized to the needs of each user provides more optimal accessibility.
- Personalization enables users to use adaptive widgets and user preferences to customize their user experience. It enables content authors to provide a default design and enable user personalization with minimal work.

Personalization

- Some people are easily **distracted or overwhelmed by lots of information** on a website.
  - They have difficulty finding and focusing on key information. Personalization allows users to hide extraneous information so they only get what they need to understand and use the main content.
- Some people have **difficulty understanding numbers** (“dyscalculia”).
  - Personalization allows users to change numeric information. For example, a temperature of 32°F/0°C is replaced with a picture of a person wearing a hat, scarf, and mittens, and the text “very cold”.
- Some people with severe language impairment **cannot read text**. They use symbols to represent words. Personalization allows users to change text to symbols.

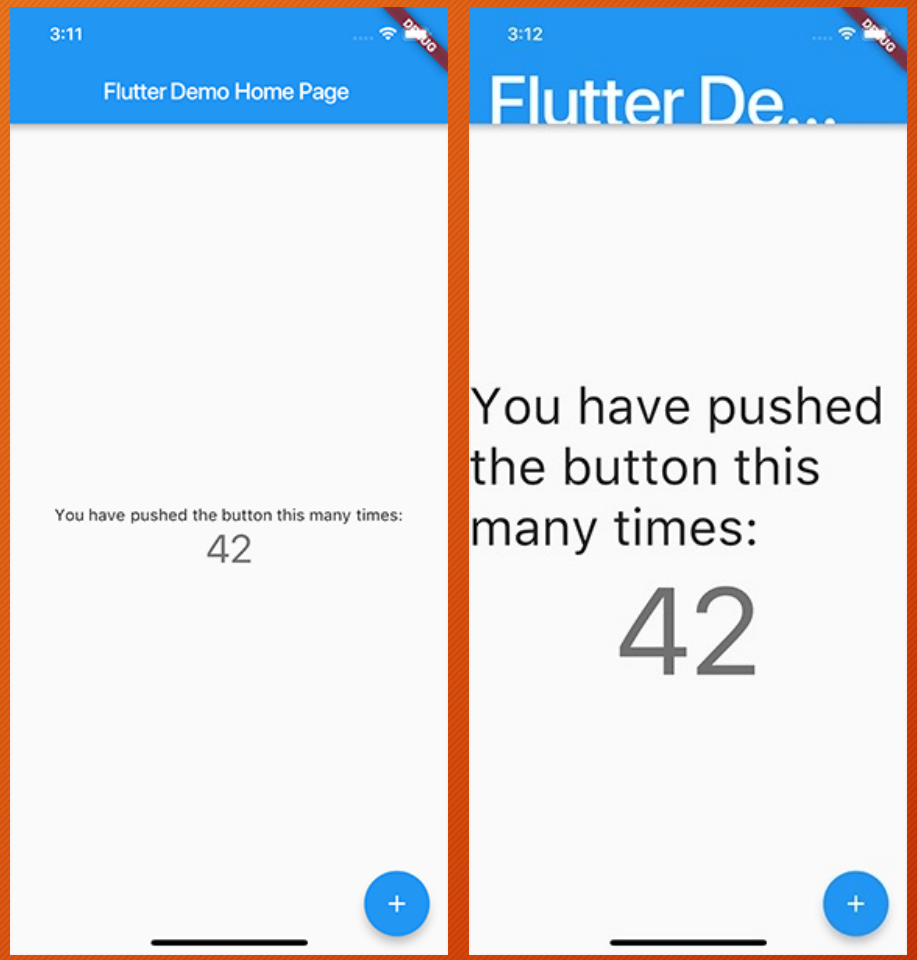
Personalization

- **Content developers** can add syntax to support personalization.
- **User agents** such as browser extensions and assistive technology can use the syntax to manipulate the content to meet the user's need. User agents can also use user preferences for different interface options.

Flutter & Accessibility

- Large fonts
  - Both Android and iOS contain system settings to configure the desired font sizes used by apps. Flutter text widgets respect this OS setting when determining font sizes.
- Screen readers
  - Screen readers (TalkBack, VoiceOver) enable visually impaired users to get spoken feedback about the contents of the screen.
- Sufficient contrast
  - Sufficient color contrast makes text and images easier to read. Along with benefitting users with various visual impairments, sufficient color contrast helps all users when viewing an interface on devices in extreme lighting conditions, such as when exposed to direct sunlight or on a display with low brightness.

Font size



Font sizes are calculated automatically by Flutter based on the OS setting. However, as a developer you should make sure your layout has enough room to render all its contents when the font sizes are increased. For example you can test all parts of your app on a small-screen device configured to use the largest font setting.

Contrast

- 1.4.3 **Contrast (Minimum):** The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA)
  - **Large Text:** Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;
  - **Incidental:** Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
  - **Logotypes:** Text that is part of a logo or brand name has no minimum contrast requirement.

https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html

Procedure

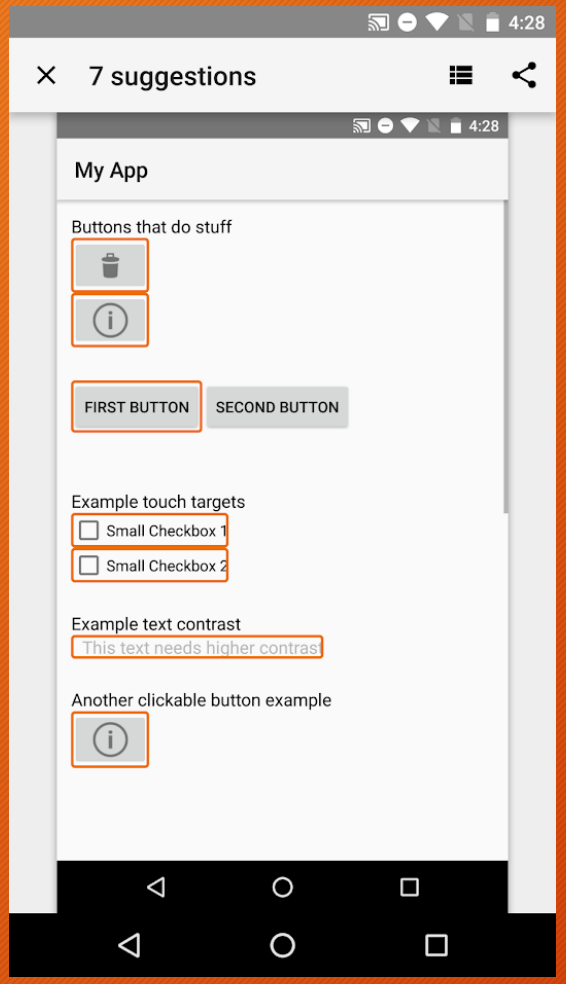
1. Measure the relative luminance of each letter (unless they are all uniform) using the formula:
    1. L = 0.2126 * **R** + 0.7152 * **G** + 0.0722 * **B** where **R**, **G** and **B** are defined as:
        1. if R_{sRGB} <= 0.03928 then **R** = R_{sRGB} /12.92 else **R** = ((R_{sRGB} +0.055)/1.055) ^ 2.4
        2. if G_{sRGB} <= 0.03928 then **G** = G_{sRGB} /12.92 else **G** = ((G_{sRGB} +0.055)/1.055) ^ 2.4
        3. if B_{sRGB} <= 0.03928 then **B** = B_{sRGB} /12.92 else **B** = ((B_{sRGB} +0.055)/1.055) ^ 2.4
    2. and R_{sRGB}, G_{sRGB}, and B_{sRGB} are defined as:
        1. $R_{sRGB} = R_{8bit}$ /255
        2. $G_{sRGB} = G_{8bit}$ /255
        3. $B_{sRGB} = B_{8bit}$ /255
    3. The "^" character is the exponentiation operator.

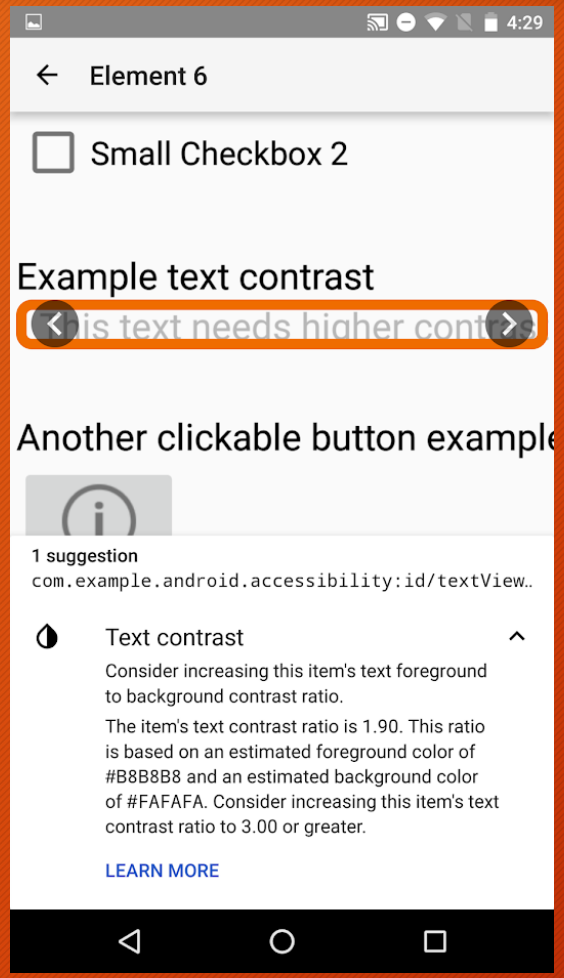
    Note: For aliased letters, use the relative luminance value found two pixels in from the edge of the letter.
1. Measure the relative luminance of the background pixels immediately next to the letter using same formula.
2. Calculate the contrast ratio using the following formula.
    1. (L1 + 0.05) / (L2 + 0.05), where
        1. L1 is the relative luminance of the lighter of the foreground or background colors, and
        2. L2 is the relative luminance of the darker of the foreground or background colors.

Accessibility scanner

- Android
  - Accessibility scanner (https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor&hl=en)
- iOS
  - Xcode > Open Developer Tools > Accessibility Inspector

Accessibility scanner





Accessibility Scanner suggests improvements such as

- enlarging small touch targets,
- increasing contrast, and
- providing content descriptions

so that your app can be more easily used by individuals with accessibility needs.

Designing for accessibility can allow you to reach a larger audience and provide a more inclusive experience.