Latest tech, great!
Accessibility, not so much.

Oana Mangiurea
Senior UX Designer @ Grafana Labs
@Caelea
Why should you care?

“Computer accessibility refers to the accessibility of a computer system to all people, regardless of disability type or severity of impairment.”

One billion people, or 15% of the world's population, experience some form of disability.

At least 10% of the adult population has a vision impairment.
Plus you get...

- A tool or app that is easier to use by everyone
- A bigger user or contributor base, as more people can use your project
- Advantage over your competitors
- **Cater for clients** whose businesses have a mandatory accessibility requirement for the software that they use, like federal agencies
- Potentially expose areas in UI or user flow that need improvement
- *Plus it’s 2021… it’s about time we not only talk about inclusiveness, but act on it*
Where can you start?
Learn about accessibility (a11y)

www.a11yproject.com

-> great resources & checklist
Free tools for designers

Color contrast checkers
- Stark - Figma, Sketch, Adobe XD
- A11y - Color Contrast Checker - Figma
- Cluse - Sketch

Color blindness simulators
- Able – Friction free accessibility - Figma
- Stark - Figma, Sketch, Adobe XD
Browser tools

Firefox Accessibility Tool - all in one tool
● color blindness simulator
● color contrast checker - use as inspect or entire page
● audit for contrast, keyboard navigation and text label issues
● shows tabs order for navigation

Google Lighthouse - runs like an audit
Other tools

**Wave** - a11y evaluation tool. Chrome & Firefox extensions

**Color blindness simulator**

*Colorblindly* - extension

*Cobis* - online tool

**Color contrast checker**

*Colour Contrast Checker* - extension & online tool

*Coolors* - extension & online tool
Supportive assistive technology

- Orca (Linux/Gnome)
- Emacspeak (Linux)
- VoiceOver (Mac OS X)
- JAWS (Windows)
- NVDA (Windows)
- Dolphin Supernova (Windows)
- ZoomText (Windows)

Why don’t you try it yourself?

Close your eyes and try to send an email using a screen reader. Don’t cheat!
Automated testing with Pa11y

Pa11y runs accessibility tests on your pages via the command line or Node.js, so you can automate your testing process.

With Pa11y you can define routes to be tested, and be able to take screenshots of those routes through actions which help you go through the app.

FOSS - LGPL-3.0 licensed
Overwhelming right?
Let’s break it into pieces

01 Audit your project
Use one of the tools presented to see where your project stands today

02 Identify main issues
Write down all important or repetitive issues

03 Estimate time/effort
Give each a rough time or effort estimate

04 Create a plan
Be it a roadmap, prioritization matrix or anything that works for you

05 Start small
Go after the low hanging fruit. What’s the easiest fix you can do?

06 Involve community
- test your GUI with them
- invite them to open accessibility related tickets
I recently tested the projects I was working on:

<table>
<thead>
<tr>
<th>Feature</th>
<th>HTML</th>
<th>Color Blindness</th>
<th>Color Contrast</th>
<th>Tab Navigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easystart V2</td>
<td>1 Error*</td>
<td>OK</td>
<td>OK</td>
<td>Unusable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Can't select integrations, can't scroll.</td>
</tr>
<tr>
<td>Integrations Management</td>
<td>5 Errors*</td>
<td>OK</td>
<td>OK</td>
<td>Unusable</td>
</tr>
<tr>
<td></td>
<td>50 Alerts</td>
<td></td>
<td></td>
<td>Can't select integrations.</td>
</tr>
<tr>
<td>Synthetic Monitoring (setup)</td>
<td>6 Errors*</td>
<td>OK</td>
<td>Ok</td>
<td>Hardly usable</td>
</tr>
<tr>
<td></td>
<td>1 Alert</td>
<td></td>
<td></td>
<td>Can create a check. Can't use probes or config.</td>
</tr>
<tr>
<td>Alerting UI</td>
<td>16 Errors</td>
<td>OK</td>
<td>2 Issues</td>
<td>Somewhat usable</td>
</tr>
<tr>
<td></td>
<td>4 Alerts</td>
<td></td>
<td></td>
<td>Can create rules, but can't edit as the button is skipped. Can create silences.</td>
</tr>
<tr>
<td>Loki (querying logs in explore)</td>
<td>7 Errors</td>
<td>OK</td>
<td>5 Issues</td>
<td>Unusable</td>
</tr>
<tr>
<td></td>
<td>9 Alerts</td>
<td></td>
<td></td>
<td>Tab gets stuck in query input. Can't see half of selection items.</td>
</tr>
</tbody>
</table>
Main issues

- **Navigation:**
  - impossible to use the app with keyboard only
  - impossible to use the app with screen readers

- **Structural elements:**
  - page headings levels are sometimes skipped - *used to facilitate keyboard navigation by users of assistive technology*
  - missing page regions or ARIA landmarks - *used to identify significant page areas*

- **Forms (design system related):**
  - using empty form labels for on/off switches
  - missing fieldset for button-groups
  - missing form labels

- **Color contrast:**
  - code highlight issues - red on black has too low contrast
Fixes everywhere

Small issues:

- Define ARIA landmarks ~ 2 days / feature
- Open Github issues for form elements related to the design system. Assign it to yourself if you can fix it and you have time
- Make sure we use labels for all dropdowns, button-groups, etc or that we have appropriate aria-labels ~ 1 day / feature
- Use proper level headings ~ 1 day / feature

We can fix them in under a week!
Fixes everywhere

**Make it accessible by keyboard**

- **Visibility of the focus indicator**
  Users need to be able to easily distinguish the keyboard focus indicator from other features of the visual design.

- **Persistence of focus**
  It is essential that there is always a component within the user interface that is active.
  **Ex:** if the user closes a dialog the active element may be hidden or removed from the DOM. In this case the active/focus element should be on the button that triggered that dialog window.

- **Predictability of movement**
  Usability of a keyboard interface is heavily influenced by how readily users can guess where focus will land after a navigation key is pressed.

- **Avoiding keyboard traps**
  TAB takes you forward to the next focusable item and that SHIFT+TAB takes you backwards.
  **Ex:** client-side form validation that forces users back into the incomplete field, preventing them from moving on the next field.

- **Bypass Blocks**
  Provide a ‘Skip to Content’ link (ex: skip sidebar menu)

~ 1 week - 2 months, depending on the feature/area
Final steps

Test with pa11y, our own keyboards and voiceover tools

Get feedback from community on the changes

Keep learning, keep fixing and write semantic HTML!
Thank you!
Let’s take some questions.