Accessibility for screeanreader et al

Werkzeuge:

- Screenreader: NVDA (https://www.nvaccess.org/download/)
- Chrome addon: WAVE Evaluation Tool (https://chrome.google.com/webstore/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh?hl=pl)
- Gibt sicher auch was für Firefox

NVDA Erklärung:

NVDA liest Bildschirmtexte vor und benutzt dazu die Voice des Betriebssystem. Zur Steuerung wird eine „NVDA-Taste“ festgelegt (per default „Einfügen“, im folgenden mit ■ gekennzeichnet).

NVDA hat dabei zwei Modi, mit denen man mit ■+Leertaste wechseln kann. Im ersten Modus werden alle Tasten der Tastatur zur Steuerung von NVDA verwendet, im Zweiten kann ganz normal geschrieben werden. Im Regelfall ist der Erste Modus aktiv, der zweite wird nur zum ausfüllen von Textfeldern benutzt (NVDA sollte im Regelfall automatisch wechseln).

Auf der Website wird standardmäßig mit den Pfeiltasten navigiert. ↓ liest dabei das nächste Element vor, ↑ das vorherige. Mit → und ← kann das aktuelle Element Buchstabe für Buchstabe vorgelesen werden.
Mit der Taste „H“ kann zur nächsten Überschrift gesprungen werden. Mit „Shift+H“ zur vorherigen.
Mit „■+S“ kann zwischen den drei Sound-Modi geswitcht werden: Voll (Standard), Stumm und nur Signaltöne.

WAVE:

Wave zeigt auf einer Website Fehler in der accessibility an. Dabei wird unterschieden in Fehler, Kontrast und Alerts. Im Details Reiter wird ausgeführt was genau die Fehler sind und auch wo auf der Seite sie sind (dabei kann es durchaus vorkommen, dass man das entsprechende Feld trotzdem nicht findet…). Über das „i“ neben den Fehler wird der Reference-Tab aufgerufen, der genauer sagt was falsch ist, warum das nicht gut ist und einen Lösungsvorschlag macht.
Der Struktur-Reiter ist recht selbsterklärend, im Kontrast Reiter kann bei Kontrastfehlern direkt eine Farbe gewählt werden die passt. Der Hex-Farbcode kann so im CSS benutzt werden.

Ein paar Notes: Gelegentlich hatte ich Fehler, die entweder Wave scheinbar nicht korrekt erkannt hat oder die zu keinem Problem geführt haben. Einfach die Seite mal mit dem Screenreader testen, wenn alles vorgelesen wird, passt es. Klassische Sachen waren, dass NVDA falsche Heading-Reihnfolgen monierte, die aber stimmten, oder fehlende (ARIA)-Label, die aber kein Problem in der Benutzung darstellten.
Außerdem, in der Lokal Version der Seite werden sehr viele Links mit „empty“ geflagged, kann ignoriert werden.

ARIA:

Aria ist eine Gruppe von Attributen, die im HTML benutzt werden kann. Diese können von Screenreader ausgelesen werden und geben diesen spezielle Anweisungen.

Worauf muss beim coden geachtet werden:

- Headings vollständig und in richtiger Reihenfolge
- Felder mit Labels oder ARIA-Labels versehen
- Jede klickbare Grafik hat eine semantische Bedeutung. Sollte sie keine haben muss die Grafik übers CSS eingebunden sein
- Kein Autofokus, verwirrt den Screenreader nutzer

Wichtige Aria-Attribute (https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes):

- Aria-Hidden: Das Element ist für den Screenreader unsichtbar (und wird daher nicht vorgelesen). Sinnvoll bei doppeltem Text, oder in Kombination mit Label/labeledby
- Aria-Label: String der vom Screenreader vorgelesen wird. Wenn die sichtbaren Texte den Kontext nicht ausreichend erklären oder nicht vorhanden sind. Liegen die vorzulesenden Texte bereits im DOM kann Aria-Labeledby verwendet werden
- Aria-labeledby: Wie Label, aber verweist auf eine ID im DOM
- Aria-describe/describedby: Wie label/labeledby. Labels sollten kurz und prägnant sein, Beschreibungen können länger und ausführlicher sein
- Aria-live: Informiert den Screenreader, dass der Inhalt dynamisch geändert wird. Liefert eine „Wichtigkeit“ mit wobei normalerweise ‚polite‘ genutzt werden sollte. ‚assertive‘ unterbricht den Screenreader und sollte nur genutzt werden, wenn absolut nötig.

Beispiele:
Aria-Hidden: in laya-la-feedback view.vue

```
<div class="d-flex justify-content-between">
  <b
    v-for="cat in categoriesLocal"
    :key="cat"
    aria-hidden="true"
  >{{ cat }}</b>
</div>
```

Cat wird nicht vorgelesen. Cat sind Überschriften über einem Slider und ist nur verwirrend, wenn sie vorgelesen werden, bevor der Silder announced wird. Außerdem wird cat im slider selbst vorglesen.

Aria.Label: in laya-la-feedback view.vue

```
<input
  v-model.number="choice[i]"
  type="range"
  class="custom-range"
  min="0"
  :max="categoriesLocal.length-1"
  :aria-valuenow="choice[i]"
  :aria-valuetext="categoriesLocal[choice[i]]"
  :aria-label="y18n('layaLaFeedback.label.slider')"
>
```

Auf den slidern wird normalerweise die Position des Array vorgelesen. Wir wollen aber den Inhalt des Array.

Aria-labelby: in laya-la-feedback edit.vue

```
<i
  id="questionmark"
  v-b-tooltip.left
  class="fas fa-question-circle"
  :title="y18n('showTip')"
  aria-labelledby="tooltipText"
  aria-live="polite"
  @click="toggleTip"
>
```

Das Feld mit der id “tooltiptext“ soll vorgelesen werden. In diesem Fall, weil es erst durch drücken eines Button sichtbar wird

Aria-Live (siehe aria-labelby):
Die Änderung im Feld ‚‘tooltiptext‘ soll vorglesen werden. Änderung hier: es wird angezeigt.
Polite: Der Screenreader wird nicht unterbrochen.