Webtoegankelijkheid

Radix

BOSA
Federale overheidsdienst
Beleid en Ondersteuning
Service public fédéral
Stratégie et Appui

1W

Context

Web Accessibility

The practice of making content and applications accessible to people with various disabilities, including **blindness, low vision, deafness, hearing loss, learning disabilities, cognitive impairments, limited mobility, speech impairments, and photosensitivity.**

Legal context

The European Directive on accessibility of government websites and mobile applications has been in force since December 2016. This directive, which is binding, stipulates a number of things:

1. That government websites and mobile applications must be accessible for everyone

2. How accessibility is defined

3. That every website and mobile application should have an accessibility statement

4. How to monitor and report

The goal

A tool that can check the accessibility of websites and mobile applications that have been developed: the accessibility Checker. This checker is partly automated, but still needs a lot of manual intervention. The goal is to extend this tool with AI in order to automate it even more.

The goal is to identify problems and then signal them to the business owner of the website or application on which the problem occurs. It is explicitly NOT the goal to also resolve an identified problem.

The accessibility checker

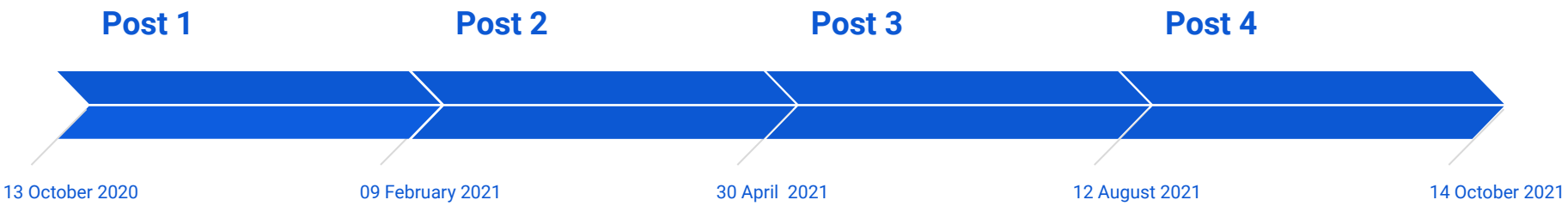
Tests that are currently performed by AC: https://openfed.github.io/AccessibilityCheck/overview/index.html

This javascript tool is based on the HTML Codesniffer Accessibility Auditing tool. Based on a set of test routines encoded in javascript, official accessibility criteria are checked with three possible reporting levels depending on the severity and reliability of the detection:

- level 0 : no notification - some criteria are difficult or impossible to test via automated testing and not linked to a specific HTML element
- Level 1 : "Notice" - situations where the validity of a certain criterion is not automatically testable, but a human validation on specific HTML-elements can be done.
- Level 2: "Warning" - situations where the validity of a certain criterion cannot be determined with certainty, but the situation suggests that the criterion is not met.
- Level 3: "Error" - situations where the test determines with certainty that a criterion is not met.

Project Overview

Timing



Post 1

Post 1

Identify and analyse contrast-related criteria through image recognition

WCAG 1.4.11 - Non-text contrast

The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):

- **User Interface Components** Visual information required to identify user interface components and states, except for inactive components or where the appearance of the component is determined by the user agent and not modified by the author;
- **Graphical Objects** Parts of graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.

WCAG 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:

- **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 contrast requirement.

Post 1: Solution

Identify and analyse contrast-related criteria through image recognition

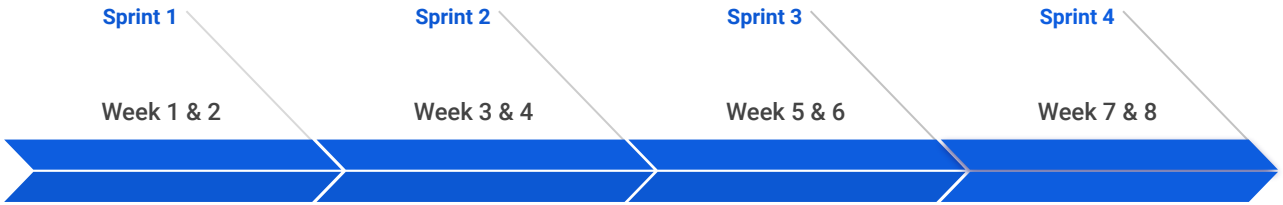
WCAG 1.4.3 – Contrast (Minimum)

1. Render website using headless browser and take a screenshot (end user's screen size & zoom level)
    a. Note that all of the page's contents should be in the screenshot
2. Compute locations of text pixels (Tesseract: image_to_boxes function returns boxes of the characters)
    a. Letter pixel color = mode of box
    b. Background pixel color = mode of edge around box
3. Calculate the contrast ratio for each character
4. Summarize contrast ratios

WCAG 1.4.11 - Non-text contrast

1. List all non-disabled interactive elements + graphical objects
2. For each element: take screenshot
3. Cluster visually similar colours
    a. CIE-lab
4. Compute a weighted contrast using (3)
    a. E.g. 2 black pixels should not affect the whole computation

Post 1



Integration

- Integrate our Python SDK into AC
- Take screenshot of screen

Develop components

- detect text in screenshot
- Find the edges
- Calculate contrast

Integration

- Integrate into one pipeline
- Test

Iterate until acceptance

Future improvements

Ways to improve the usability of the Accessibility Checker:

1. **Also check changeable content:** For now the web page to be checked is rendered in the back-end in order to find contrast issues. This means it does not the the page exactly as the user sees it.
    - It will not see changeable content in there different versions (for example sliders)
    - different states of the web page (for example collapses, extensions...)
    - different screen sizes
        - -> could be solved by finding the hardcoded breakpoints in the css. Then we can render the web page for these different breakpoints in the back-end.
2. **Handle authentication and cookies:** in the current solution it is impossible to check pages that need authentication. Also there is a problem with the 'cookies settings' overlay that almost all website have.
        - -> Take screenshot from client-side / fully run solution locally / work with encrypted server
3. **Make solution run faster:** The goal of this solution was to test the strength of using AI. This solution has not been optimized in terms of speed. We can list an overview of which components are slow and which low hanging fruit you can implement to improve this.
    - Slow components:
        - -> Creating 2 headless Chrome browsers needed to take 2 screenshots

Ways to improve checking the contrast of the interactive components:

- **Check contrast within an interactive component:** for now, we only check the contrast between a component and its background. For some components (for example slider) it is also important to check the contrast within the component. Otherwise the user will not be able to see if a slider is turned on or off.
- **Better detect interactive components**: now we only detect a simple subset of interactive components (= submit and input). Often these components are made not using the standard html elements. These are not detected at the moment
        - -> Train a model to detect interactive components.
- Take into account the ACT-rules, https://act-rules.github.io/rules/afw4f7: "This rule checks that the highest possible contrast of every text character with its background meets the minimal contrast requirement." – so we would need to compare the brightest and the darkest colours in the detected boxes, rather than the "most used".

Post 2

Post 2

Identify language

WCAG 3.1.1 Language of Page

The default human language of each Web page can be programmatically determined.

WCAG 3.2.1 Language of Parts

The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

Identify language:

What problems will we solve

WCAG 3.1.1 Language of Page

The default human language of each Web page can be programmatically determined.

❗ Errors:

- no language attribute set
- wrong language set

WCAG 3.2.1 Language of Parts

The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

❗ Errors:

- no language attribute set
- wrong language set
- header/footer language of multilingual website is in Dutch, while contents are in English or French

Identify language:

General remarks

- We will only focus on Dutch, English, French and German
- check all html elements and attributes
- how do we define a grammatically correct sentence?
  - Minimum of 4 words

Post 2: Solution

Identify language

WCAG 3.1.1 Language of Page

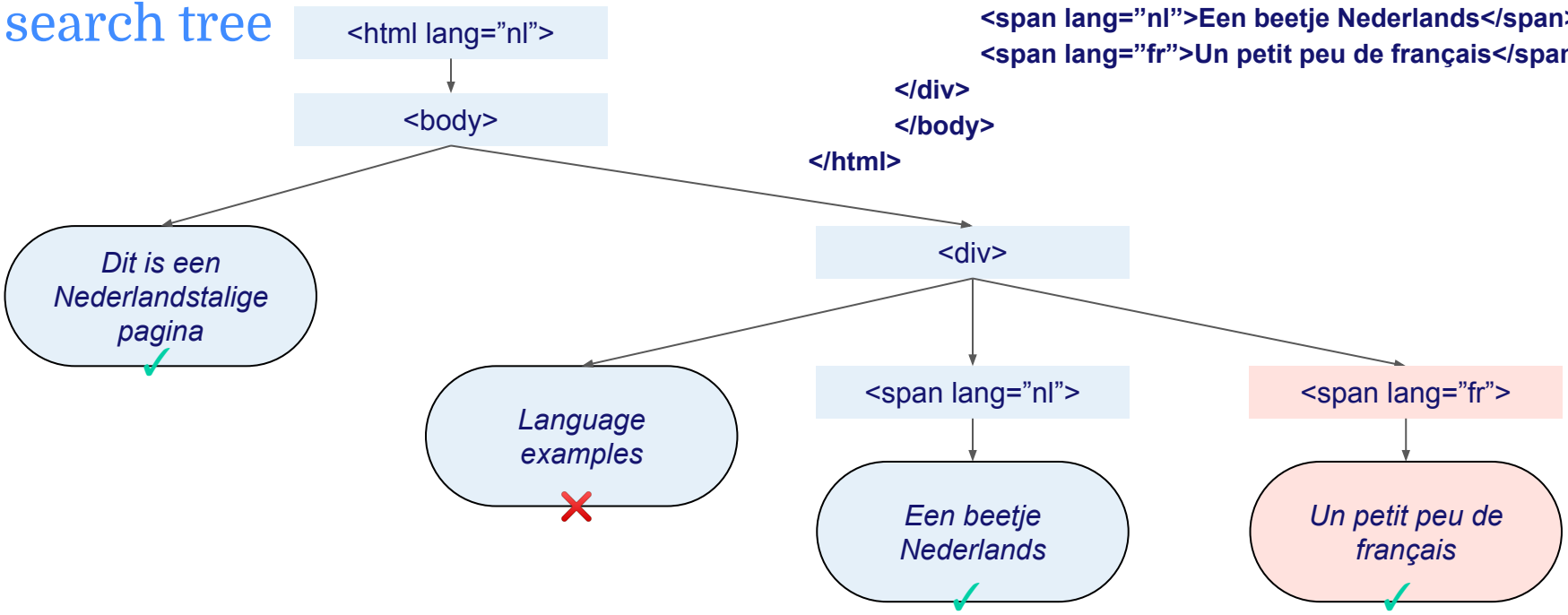
WCAG 3.2.1 Language of Parts

1. Find the visible text elements in HTML code
2. For each text element:
    a. Use Python library to determine the language, e.g. `langid`, `langdetect`, `FastText`
    b. Compare the predicted language to the language attribute of that text element
    c. Use probability thresholds to determine the verdict
3. Communicate back to the AC

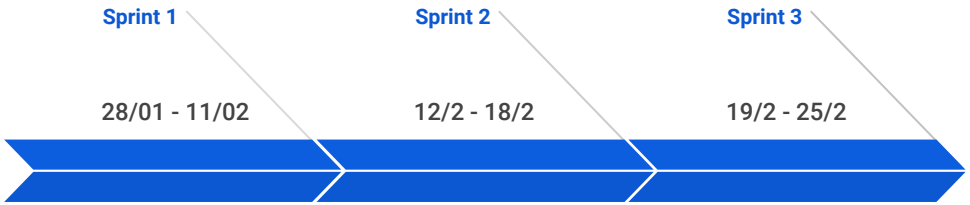
Post 2: Solution

Identifying language is a search tree

```
<html lang=”nl”>
        <body>
        Dit is een Nederlandstalige pagina.
        <div>
                Language examples
                <span lang=”nl”>Een beetje Nederlands</span>
                <span lang=”fr”>Un petit peu de français</span>
        </div>
        </body>
</html>
```



Post 2



Develop components

- Define language of a page
- Compare language of a page with label
- Define language of each html element
- Compare language of html element with label

Integration

- integrate into tool
- implement feedback

Testing

- Test tool and try to go to acceptance

Identify language:
Possible future iterations

1. Check language of
   - columns or rows in a table (cfr. Belgian law websites)
   - PDF files
   - Subtitles of videos
2. aria label -> warning when less than 4 words
3. Train fasttext model on Belgian websites
4. Non-text attributes with other language tag as the surrounding text, should give a warning. For example a phone number should probably have the same language as its context.

Post 3

Post 3

Identify images

WCAG 1.1.1 Non-Text (A)

All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below.

- **Controls, Input** If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to Success Criterion 4.1.2 for additional requirements for controls and content that accepts user input.)
- **Time-Based Media** If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to Guideline 1.2 for additional requirements for media.)
- **Test** If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content.
- **Sensory** If non-text content is primarily intended to create a specific sensory experience, then text alternatives at least provide descriptive identification of the non-text content.
- **CAPTCHA** If the purpose of non-text content is to confirm that content is being accessed by a person rather than a computer, then text alternatives that identify and describe the purpose of the non-text content are provided, and alternative forms of CAPTCHA using output modes for different types of sensory perception are provided to accommodate different disabilities.
- **Decoration, Formatting, Invisible** If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.

Post 3

Identify images
EXTRA INFORMATION

The current tools check for the **presence/absence of such a an alternative text**, but can not or only rudimentarily check whether the description is also "equivalent" to the image itself. Graphic elements that are purely "decorative" (e.g. a visual "curl" in the margin of a text) are also explicitly expected to contain an *empty* alternative description. Image recognition techniques can be used to find a "match" between the present description for images and their content, or just to find a negative match (e.g. a picture of a tree where the textual description would be "cat").

Note that even a human being would not be able to correctly assess a description of an image and its representation in 100% of cases, as this would require a perfect understanding of the context. In an article on pets, describing an image as a “dog” may be perfectly acceptable, but in an article on measures against violent dog breeds, images of a pit bull or a poodle are not served with a general formulation "dog". Therefore, we do not expect a solution that covers this case with perfect accuracy, but it should be possible to cover at least 50% of mislabeled images and less than 5% of false detections.

Post 3: solution

Compare alternative text with content on images

Flowchart:

- Take alternative text
- Translate it to English
- Feed text and image to CLIP model
- Check similarity from CLIP model
- Decide whether it similar enough using fine tuned threshold.

guacamole (90.1%) Ranked 1 out of 101 labels

✓ a photo of **guacamole**, a type of food.

× a photo of **ceviche**, a type of food.

× a photo of **edamame**, a type of food.

× a photo of **tuna tartare**, a type of food.

× a photo of **hummus**, a type of food.

television studio (90.2%) Ranked 1 out of 397

✓ a photo of a **television studio**.

× a photo of a **podium indoor**.

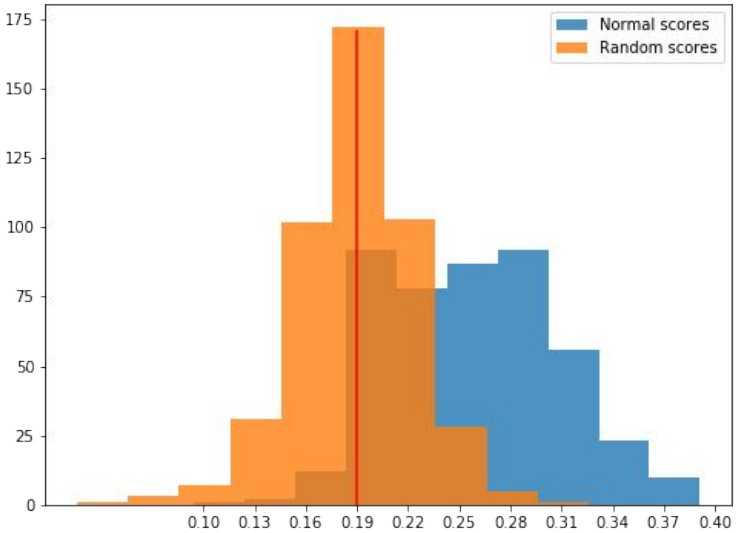
× a photo of a **conference room**.

× a photo of a **lecture room**.

× a photo of a **control room**.

Post 3: solution

Determine threshold



Flowchart:

- The distribution of scores for a constructed dataset of image-text pairs is shown in blue
- The distribution of scores where text and images were randomly paired up is shown in red.
- We decided to take 0.19 as a threshold which resulted in 48% of wrong pairs are predicted as correct and 5% good pairs are predicted as wrong.

Post 4

Post 4

Extract text from images

WCAG 1.1.1 Non-Text (A)
see post 3

WCAG 1.4.5 Images of text
If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following:

- **Customizable** The image of text can be visually customized to the user's requirements;
- **Essential** A particular presentation of text is essential to the information being conveyed.

Decorative vs non-decorative

Using some rules to identify whether an image is decorative or not

For full flow decision tree, check the attached document ‘Decorative images’

Decorative vs non-decorative

Using some rules to identify whether an image is decorative or not

Flowchart:

1. Is the image clickable? -> if yes, it is definitely non-decorative
2. Does the image contain text? -> if yes, it is probably non-decorative
    a. If yes: compare image with caption to see if text corresponds with caption

Post 3 and 4:

Possible future iterations

Our main struggle was the identification of decorative vs non-decorative. We did not find a good rule based approach. A possibility could be to train a model, but for that we would need a dataset of decorative and non-decorative labeled instances.

We also found that identifying whether an image is decorative or not, is impossible when the context is not taken into account. By context we mean the text surrounding the image. So a future improvement could be to train a more complex model that not only looks at the image itself, but also at the surrounding text.

In general the AC needs to be optimised in terms of performance, before it can be productionised. For now it is a little bit too slow.