Technical specification for the feature: Testing the content of pages

This is the technical specification for the Testing the content of pages feature. The feature was created as part of an EU-funded project called We4Authors Cluster. All features, their technical specification and their video documentation are published on www.accessibilitycluster.com.

If you have any questions or comments, please do not hesitate to reach out to research@funka.com.

The technical specification contains several perspectives:

- the web interface of the feature, based on the experience of the end user or, when it comes to the testing-features, the web author;
- the code of the feature ensuring accessibility by default;
- a description highlighting the key elements of the code;
- reference to a video documentation;
- recommendations for implementation.

Specifications of the feature: Testing the content of pages

When creating content, the web author usually combines text and images with lists, blocks, objects and functionalities to compile a full page. By allowing the web author to test the page before publishing, many unnecessary accessibility problems can be avoided. In this feature, there is also a built-in auto-repair functionality that suggests fixes to the identified accessibility issues and the results do not require programming skills.

There are many automatic accessibility testing tools on the market; open source as well as commercial. They vary in quality and complexity, but more importantly, they focus on different target audiences. Many of the tools are made specifically for developers, to support their work while creating a website. Others help designers to choose the right combination of colours and the like. Some tools are made to support web authors to publish in an accessible way.

There are also many other kinds of tools that may help the authors, like spell checks and crawlers that look for broken links etc.

CivicActions maintains a list of available open source tools for evaluating page-level accessibility: https://accessibility.civicactions.com/guide/tools#page-level-evaluation

As with all automatic accessibility testing, these tools can only test parts of the requirements, and some point to issues that need to be checked manually.

By building an API-based service using an automatic accessibility testing tool core service, the page specific content can be checked for certain accessibility issues. This way, only web author-relevant failures would be presented, so that the author can concentrate on remediating these.

The main goal of this feature is to help the web author make sure different objects or parts of the web page are checked separately and that the end result is accessible - before publishing the page.

Web interface: web author view

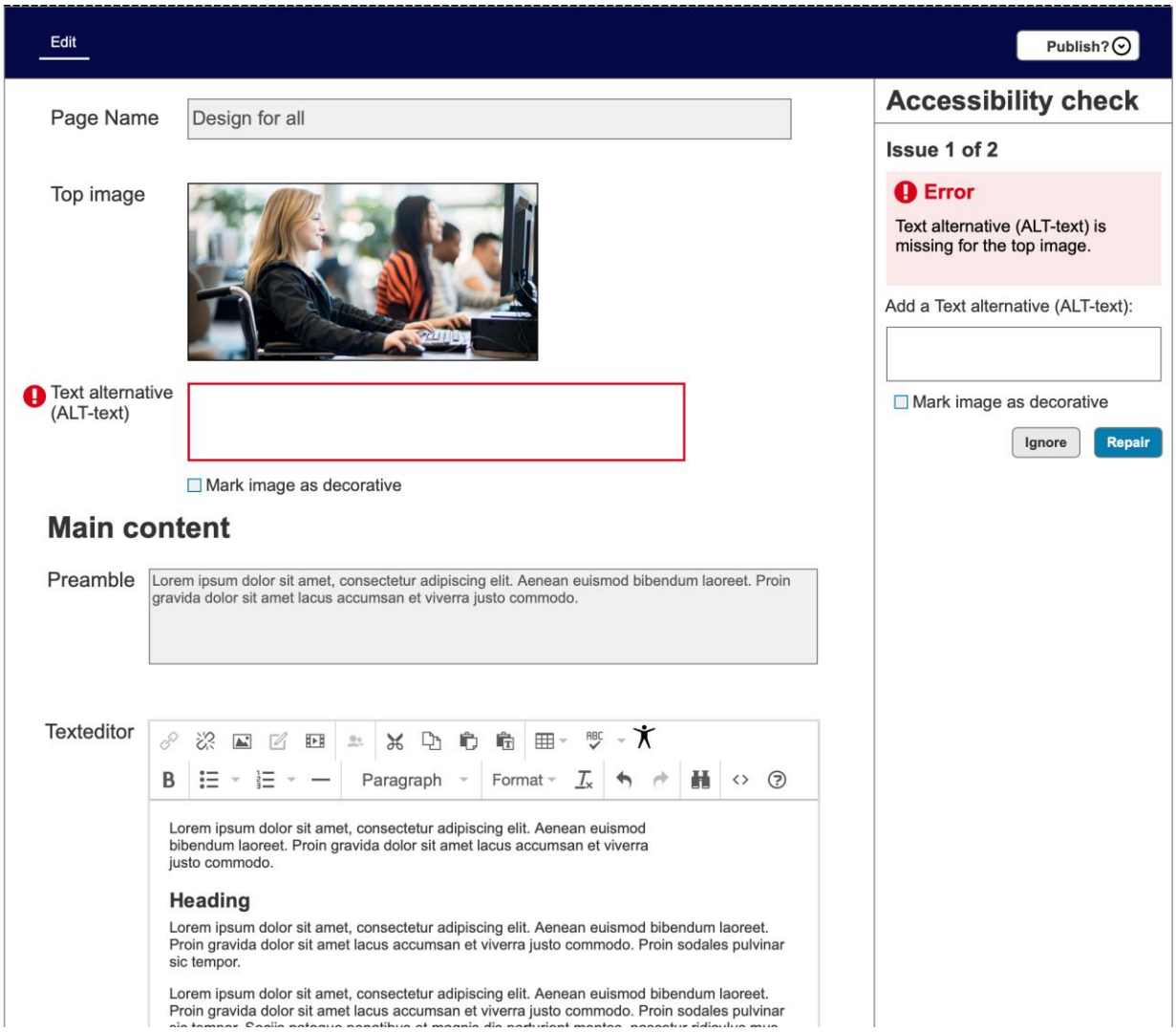

Figure 1: Web author view

Possible built-in success criteria conformance tests to be done

- Using <h1>-<h6> to identify headings.
- Providing heading elements at the beginning of each section of content.

- Sequential headings.
- Adjacent links.
- Using <ol>, <ul> and <dl> for lists or groups of links.
- Providing short text alternatives that provide a brief description of the non-text content or role="presentation" attribute.
- Using <caption> elements to associate data table captions with data tables.
- Using table markup to present tabular information, <th> and <td>.
- Using the scope attribute to associate header cells and data cells in data tables.
- Parsing, in content implemented, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique.

CivicActions maintains a list of open source tools for evaluating page-level accessibility:
https://accessibility.civicactions.com/guide/tools#page-level-evaluation

Tools like Sa11y and Editoria11y allow web authors to focus on what they can control. Most accessibility errors are introduced in the editor's body field, so it is possible to target the WYSIWYG to that field and generate accessibility errors that target that section of the page specifically.

Sa11y:
https://ryersondmp.github.io/sa11y/#install

Editoria11y:
https://itmaybejj.github.io/editoria11y/

Editoria11y needs to have JavaScript added to the HTML headers for the page:

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<link rel="stylesheet" media="screen" href="/css/editoria11y.css">
<script src="/js/editoria11y-prefs.js"></script>
<script src="/js/editoria11y-localization.js"></script>
<script src="/js/editoria11y.js"></script>
```

There is also a Drupal module:
https://www.drupal.org/project/editoria11y

Video documentation

This technical specification has been developed in the We4Author Cluster project to reflect recommendations for accessibility features that can be implemented in any authoring tool. The specifications are complemented by a video documentation, covering a live description of:

- web author challenges, and
- feature solutions.

Recommendations for implementation

To make sure the implementation of the features is not causing accessibility problems for web authors with disabilities:

- avoid drag-and-drop for choosing template, and/or always have a keyboard alternative;
- always provide one pointer alternative without specific gestures;
- always support keyboard navigation;
- consider keyboard shortcuts;
- do not rely on sensory characteristics as the sole indicator for understanding and operating content;
- do not indicate important information using colour alone;
- make sure there is enough contrast between text objects and its background colour.