

# Accessible EPUB

Bachelor Thesis  
by

**Sachin Rajgopal**

Study Centre for the Visually Impaired Students (SZS)  
Department of Informatics

First Reviewer:  
Second Reviewer:  
Supervisor:

Prof. Dr. Rainer Stiefelhagen  
**TODO: Eintragen**  
Dr. Thorsten Schwarz

Project Period: 01/12/2017 – XX/XX/20XX



---

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den **TODO: date**



## **Zusammenfassung**

TODO: Zusammenfassung (Deutsch)



## **Abstract**

TODO: Zusammenfassung (Englisch)





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	EPUB . . . . .	2
1.2.1	Changes from EPUB 2 to EPUB 3 and EPUB 3.1 . . . . .	2
1.2.2	EPUB Structure . . . . .	3
1.2.2.1	Package document . . . . .	4
1.2.2.2	Navigation . . . . .	5
1.2.2.3	Content documents . . . . .	5
1.2.3	Goal of this thesis . . . . .	5
1.2.3.1	Document standard . . . . .	6
1.2.3.2	Editor . . . . .	7
<b>2</b>	<b>Related Work</b>	<b>9</b>
<b>3</b>	<b>EPUB Document Standard</b>	<b>13</b>
3.1	Requirements . . . . .	13
3.1.1	Switching between versions . . . . .	13
3.1.1.1	JavaScript . . . . .	13
3.1.1.2	CSS . . . . .	14
3.1.2	Text . . . . .	16
3.1.3	Images . . . . .	16
3.1.4	Mathematical formulas . . . . .	18
<b>4</b>	<b>Evaluation EPUB Standard</b>	<b>23</b>
4.1	Software readers . . . . .	23
4.2	Hardware readers . . . . .	23
4.3	Discussion: what do developers have to do? . . . . .	23
<b>5</b>	<b>AccessibleEPUB Editor</b>	<b>25</b>
5.1	Requirements . . . . .	25
5.2	Programming language . . . . .	25
5.3	Main window with editor and preview . . . . .	25
5.3.0.1	HTML Editor . . . . .	25
5.3.0.2	Preview browser . . . . .	27
<b>6</b>	<b>Conclusion and Future Work</b>	<b>29</b>
	<b>Bibliography</b>	<b>31</b>



# 1. Introduction

## 1.1 Motivation

In recent years, the topic of accessibility has become increasingly important, especially for accessible documents. Several states have passed regulatory laws that ensure equal treatment of all people and ensure that documents are accessible to all [15].

The currently dominant format for accessible electronic documents are Microsoft Word and PDF (Portable Document Format) documents, or more precisely PDF/UA (PDF/Universal Accessibility) documents. First of all, both formats have a predefined page size. While this is useful for printed documents, a computer screen can rarely display all contents of the document to the detriment of visually impaired people [13]. Therefore, an electronic document format without a fixed document size containing semantic and structural information and a fixed reading order would be better suited to meet the requirements of accessibility.

Furthermore, different "selectable" forms of presentation would be advantageous, especially for graphics or mathematical formulas. For example, formulas in the  $\text{\LaTeX}$  source code for blind users or high-contrast images for users with limited residual vision. This could be combined with EPUB 3 [13].

EPUB stands for **E**lectronic **P**ublication and is a format primarily used for books in an electronic format (E-book). The EPUB format was created by the International Digital Publishing Forum (IDPF) and the current version is 3.1 which is a minor update to EPUB 3 [10]. EPUB uses XML based formats like XHTML, and thus also uses the accessibility standards and guidelines already established in many nations like the Web Content Accessibility Guidelines (WCAG)[17]. This was done as reading systems can have different screen sizes and the EPUB content can therefore be reflowable. Font type and size can also be adapted to the individual needs of the users. Visually impaired people could therefore adjust the document to their preferences in font style, size and color. The EPUB 3 specification also contains guidelines for accessibility so these features are built in and not an afterthought [6].

The EPUB working group has also made important changes from EPUB 2 to EPUB 3 which improve the accessibility of documents. For example, mathematical equations can now be displayed in MathML and there is better navigation and more support for

Name	Änderungsdatum	Typ	Größe
META-INF	08.01.2018 17:51	Dateiordner	
OEBPS	08.01.2018 17:51	Dateiordner	
mimetype	08.01.2018 17:51	Datei	1 KB

Figure 1.1: Folder structure within an EPUB file

Cascading Style Sheets (CSS). However, not all of these changes are yet supported by EPUB readers and devices [9].

## 1.2 EPUB

EPUB stands for electronic publication and is a format primarily used for books in an electronic format (E-book). The EPUB format was created by the International Digital Publishing Forum (IDPF) and the current version is 3.1, which is a minor update to EPUB 3.[10] EPUB uses XML based formats like XHTML, and thus also uses the accessibility standards and guidelines already established in many nations like the Web Content Accessibility Guidelines (WCAG). [17] This was done as reading systems can have different screen sizes and the EPUB content can therefore be reflowable. Font type and size can also be changed. Visually impaired people could therefore adjust the document to their preferences. The EPUB 3 specification also contains guidelines for accessibility so these features are built in and not an afterthought.[6]

### 1.2.1 Changes from EPUB 2 to EPUB 3 and EPUB 3.1

The EPUB working group has also made some important changes from EPUB 2 to EPUB 3 to make it more accessible. Equations can now be displayed in MathML.[9] In EPUB 2 they had to either be displayed in normal XHTML or as images. It is difficult to write math in normal XHTML, as complex symbols like fractions, roots and plus-minus might not be displayed properly. Images can display the equations properly, but only if they are SVG files can they be scaled without quality loss. Furthermore, all images are not as accessible as MathML, which can be scaled properly and can be read by a screen reader.

In EPUB 2 the navigation was done with a NCX file, usually named `toc.ncx`. [6] In EPUB 3 this is now done instead with a XHTML file, and can be created with regular XHTML elements like `ol` and `ul`, for ordered and unordered lists, respectively. XHTML is much more widely used and is already an important format in EPUB 3, while NCX files have a distinct syntax, which has to be learned in addition to XHTML. Replacing NCX means that the creator has one less document format to worry about.

Scripting with JavaScript was strongly discouraged in EPUB 2, but in EPUB 3 it is optional.[6] This allows documents to be interactive, but it is recommended that information is not hidden if a device does not support scripting. This should be done with a process called progressive enhancement. In essence, content should first be designed in the most accessible way, and then layers of enhancement should be added, which might not be supported on all devices. For example, if the creator wants to add an animation with JavaScript to the document, it might be better to add several images showing the

application/epub+zip
----------------------

Figure 1.2: Contents of the `mimetype` file

stills of the animation at important junctions. Each still image would also have alternative text for users with screen readers. The creator could then add an animation for devices that support it, enhancing their experience.

EPUB 2 supports Cascading Style Sheets(CSS) 2, while EPUB 3 supports CSS 2.1 with added modules from CSS 3 which were defined as basic EPUB CSS Profile. The EPUB CSS Profile was removed in EPUB 3.1 and the IDPF defined more general CSS support requirements in its place.[11] Henceforth they also use the more general definition of CSS as mentioned by CSS Working Group, which results in creators not being required to learn specific CSS attributes only seen in EPUB.

One new feature in EPUB 3 is media overlays, which might be of considerable interest to people with accessibility requirements.[6] This feature is supposed help integrate Digital Accessible Information System (DAISY) digital talking books in EPUB 3. The first DAISY Standard originated in Sweden in 1994. DAISY is a digital standard to create an audio substitute for printed media.[5] When it was created the primary audio media were tapes and CDs, both with rather limited runtime. Discs with DAISY books, however, can hold 40 hours of audio, while the corresponding number for CDs is about 74 minutes.[2] DAISY books also allow the user to skip between chapters, pages or even sentences and create bookmarks. EPUB 3 has the same features, but media overlays now allow it to synchronize audio narration to text. The user can still use Text-To-Speech(TTS) rendering, but as it is computer generated, it might not pronounce all words properly. Prerecorded audio narration is of course better at this, and now the user can switch between reading and audio narration without navigating in an audio file.

However, many of these changes are not supported by reading devices, and this will be topic of discussion in chapter 4, where reading devices are evaluated, both software and hardware.[9]

## 1.2.2 EPUB Structure

The first step to knowing what an EPUB is by knowing what an EPUB file is. It has the extension `.epub`, but is actually just a renamed ZIP file(extension `.zip`). After the extension of the EPUB file is changed, the new ZIP file can be decompressed and the files within the document can be accessed.[6]

Compressing the folder back to a ZIP file has to be done with care. The `mimetype` should be the first file to added to the ZIP container and not the other two folders. If this is not done properly, the EPUB file is not in the proper format and some readers can not read the file.

The folder structure of an EPUB file is shown in figure 1.1. The file named `mimetype` has a single line indicating that the ZIP file contains an EPUB. The META-INF folder contains just one file named `container.xml`. This file has to indicate the location of the package file(extension `.opf`). The name of the folder named OEBPS can actually be freely chosen, however, `container.xml` has to describe the path to the package file correctly. OEBPS stands for Open eBook Publication Structure, which was the format superseded by EPUB.[12]

```
<?xml version="1.0" encoding="UTF-8"?>
<container xmlns="urn:oasis:names:tc:opendocument:xmlns:container" version="1.0">
<rootfiles>
<rootfile full-path="EPUB/package.opf" media-type="application/oebps-package+xml"/>
</rootfiles>
</container>
```

Figure 1.3: Contents of container.xml

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http://www.idpf.org/2007/opf" xmlns:dcterms="http://purl.org/dc/terms/">
  <dc:title>BookTitle</dc:title>
  <dc:creator>Author</dc:creator>
  <dc:language>en</dc:language>
  <dc:publisher>Publisher</dc:publisher>
  <dc:identifier id="BookId">BookIdentificationNumber</dc:identifier>
  <meta property="dcterms:modified">2018-02-07T10:43:33Z</meta>
  <meta name="AccessibleEPUB" content="0.1.0" />
</metadata>
```

Figure 1.4: The various Dublin Core tags in the metadata

### 1.2.2.1 Package document

The OEBPS folder traditionally contains the package document which has the extension .opf. The package document contains five XML declarations:

- metadata (required)
- manifest (required)
- spine (required)
- guide (optional)
- bindings. (optional)

The `metadata` declaration information about the document in Dublin Core form. Only `dc:title`, `dc:language`, `dc:identifier` and one `meta` are required, but there are several optional ones. The data entries have to follow the specifications.

The `manifest` declarations contains all of the files which should be included in the EPUB file with an ID, the path to each file and an appropriate declaration of the media type. Properties also have to entered. If a XHTML file is a navigation document, `nav` has to be entered in the properties.

The `spine` declares the default reading order of the EPUB. The reader is free to go to whatever page they want, but the `spine` allows the document creator to set a logical order of the files in the spine. The only file types which do not require a fallback in the `spine` are XHTML and SVG files.[6]

The `guide` is deprecated and only needed for EPUB 2, but it is useful for backwards compatibility. `bindings` is used for fallback options for interactive media. Both of these are optional and will therefore not be discussed further in this bachelor thesis.

```

<manifest>
  <item id="ncx" href="toc.ncx" media-type="application/x-dtbnex+xml"
  "/>
  <item id="Content.xhtml" href="Text/Content.xhtml" media-type="
  application/xhtml+xml"/>
  <item id="style.css" href="Styles/style.css" media-type="text/css
  "/>
  <item id="navid" href="Text/nav.xhtml" media-type="application /
  xhtml+xml" properties="nav"/>
</manifest>

```

Figure 1.5: Files listed in the manifest

```

<spine toc="ncx">
  <itemref idref="Content.xhtml"/>
</spine>

```

Figure 1.6: The reading order declared in the spine

### 1.2.2.2 Navigation

The OEBPS folder also contains a file, normally named nav.xhtml, which is the table of contents. The table of contents can be used to navigate to any tagged section and is created by simply making an ordered or unordered list in HTML. A short example is shown in figure 1.7.

### 1.2.2.3 Content documents

One of the primary documents of an EPUB document are Extensible Hypertext Markup Language (XHTML) file. XHTML is used, because they are formats of the web. Websites have to be displayed on a variety of devices, from phones with screen dimensions of a few inches to 50 inch television screens with varying display resolutions. The content is therefore reflowable and suits Ebook readers as they are available in various sizes.

Another primary document of EPUB documents are Scalable Vector Graphics (SVG) files. They are based on XML and while image formats like JPEG and PNG are raster based with pixels, SVG files are vector based and are rendered with mathematical formulas. This means that they retain their appearance at any size and do not get blurry at larger sizes than they created at.

## 1.2.3 Goal of this thesis

The goals of this thesis will be described in the following sections.

1. Create an accessible document standard based on EPUB 3

```

<nav epub:type="toc" id="toc">
  <h1>Table of Contents</h1>
  <ol>
    <li>
      <a href=" ../ Text/Content.xhtml">Start</a>
    </li>
  </ol>
</nav>

```

Figure 1.7: An example table of contents

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 1.8: Quadratic Equation

```
{x = \frac{ - b \pm \sqrt{ b^2 - 4ac }}{2a}}
```

Figure 1.9: Quadratic Equation in LaTeX code

2. Develop an editor which will then be able to create EPUB files of this new standard.

### 1.2.3.1 Document standard

The first goal of this thesis is to create an document standard based on EPUB 3. This document standard must also include support for STEM (Science, Technology, Engineering, Mathematics) subjects. Existing literature, both electronic and in paper form, is frequently not accessible due to the figures and mathematical formulas in them. If the figure does not have proper captions or alternative text, blind people will be unable to extract the knowledge sighted people will from it. Blind people read formulas with specialized mathematical braille notation, such as Nemeth code in the USA and Marburger mathematical code in Germany.[3] However, few books are printed in braille, braille printers are uncommon and expensive and the resulting books tend to be very heavy. When the textbooks are in electronic form, the mathematical equations have to written in LaTeX code. The quadratic equation is shown in figure 1.8, while the corresponding LaTeX code is shown in figure 1.9. Visually impaired people have different requirement for mathematics and graphics. They should scale and not get pixelated when the size is increased, as shown in figure This is best done with SVGs, which were mentioned in last chapter. Visually impaired people also need the text to be a sans-serif font, like Helvetica or Arial.[8]

Sighted people have their formulas appear as a serif font as shown in figure 1.8, visually impaired people need it to appear as sans-serif and blind people needs LaTeX code.[3] This is difficult to do in one document, so separate documents have to be created for

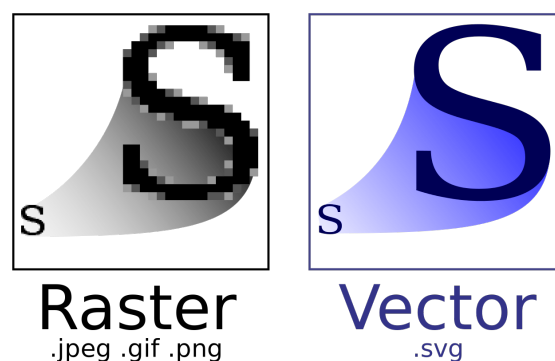


Figure 1.10: Source : [https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Bitmap\\_VSVG.svg/1280px-Bitmap\\_VSVG.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Bitmap_VSVG.svg/1280px-Bitmap_VSVG.svg.png)



the blind and the visually impaired, creating additional work for the document creator. The new standard must have all include versions for sighted, visually impaired and blind people in one document to simplify the creation process. Instead of using the currently dominant formats, PDFs and word documents, this will be done with EPUB 3, because they are more suited to electronic formats and allow dynamic content, thanks to CSS 3 and JavaScript. There will be a easy way to switch between three versions. For example, when the blind version is chosen, content such as formulas will change to LaTeX code.

#### **1.2.3.2 Editor**

The second goal of this thesis is to develop an editor to create documents of this standard easily. It should not require programming ability and allow insertion of formulas and figures which will be available in all three versions. The editor should not be difficult to learn and be What-You-See-Is-What-You-Get(WYSIWYG), much like a word processor, instead of XHTML which requires coding.



## 2. Related Work

A project which also uses HTML/XHTML to make documents accessible is discussed in a paper[16] by Voegler et al. HTML is chosen as a book format, because it is operating system independent, accessible to screen readers and the content can be adapted visually to a person's wishes. However, the work is not done by original document author, usually university staff, but by students who transcribe it to accessible versions. They have to request the content from authors and get permission. Coding with HTML creates problems like linking errors, invalid HTML-code, etc, so Voegler et al recommend using Markdown[7], which is easier to write than HTML. It can then be converted to a variety of formats using pandoc[14], which also converts LaTeX code to MathML. While reducing the number of mistakes compared to HTML, Markdown still requires programming and only people with sufficient knowledge of it will be able to use it effectively. Students doing the transcription will be more knowledgeable in coding than teachers in schools for the blind.

If EPUB is compared to HTML as book format, it is of course very similar as EPUB uses HTML/XHTML. HTML as a book format can be described as a simplified version of EPUB only without the packaging such as the package document, in essence only having the OEBPS folder. However, images have to be added manually to the folders, and CSS files have to be created. These are extra steps which might result in frequent mistakes. Furthermore, Pandoc is command line based and is therefore more difficult to use than programs with a graphical user interface(GUI). It also has to be run separately after coding in Markdown, instead of being integrated in the creation process. Nevertheless, this paper shows that HTML, if created properly, is suitable to the needs of blind and visually impaired students.

Another approach is offered by Leporini et al in their project Book4All[4]. It originally allowed PDFs to be exported to XHTML or DAISY, later EPUB 3 was added as an option. It reads the PDF with the PDF Viewer to gather information about fonts and recognize if the document is in a multi-column layout. Then Book4All has to take this information and extracts the text and semantic information. This includes keeping the format of the PDF, such as recognizing headings, images and tables being properly tagged. The user has the option of correcting tagging mistakes. It then has to be exported to its final format.

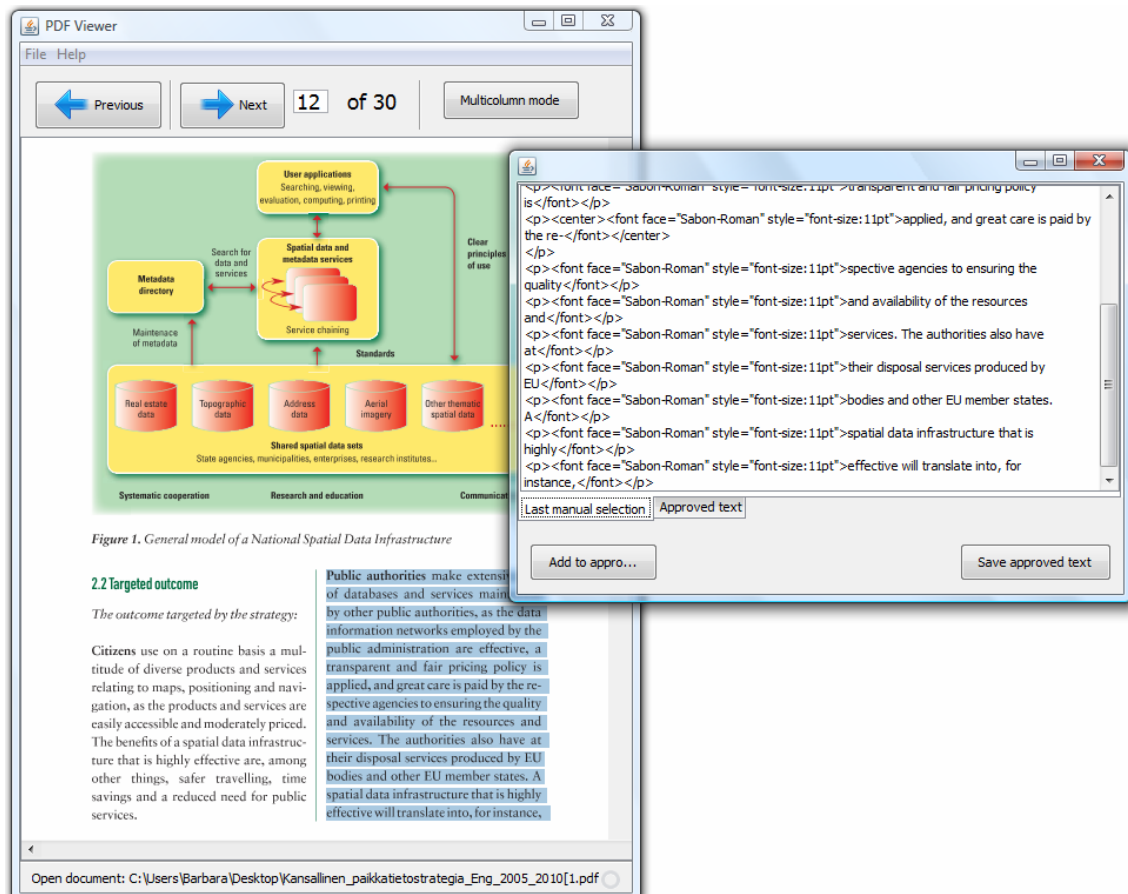


Figure 2.1: A view of the ad-hoc PDF Viewer extracting text from a multi-column layout document [4]

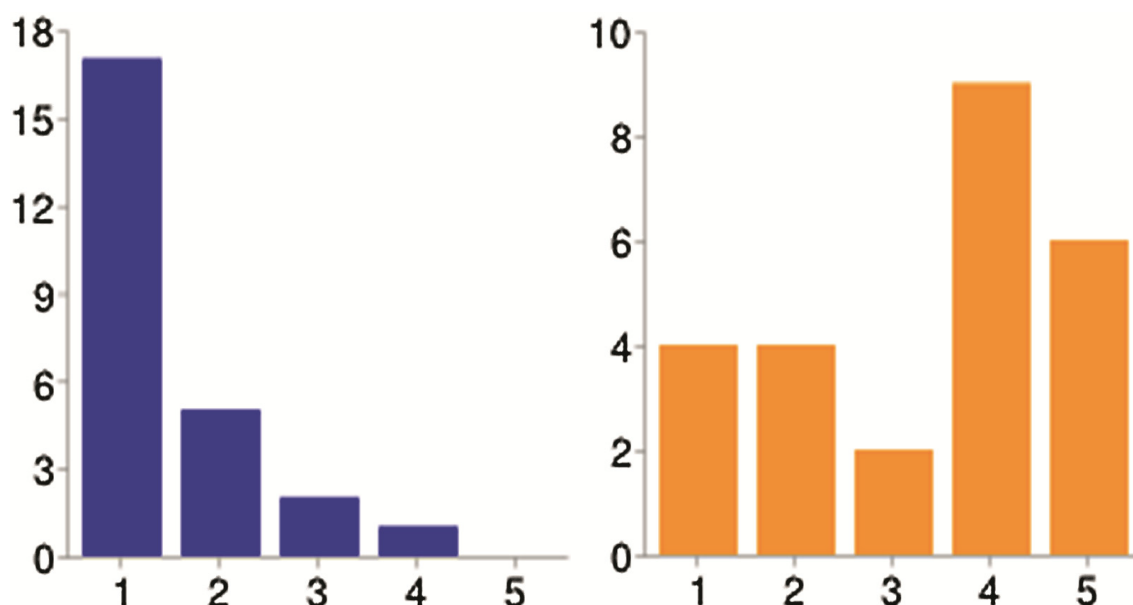


Figure 2.2: Difficulty degree to access the content for the EPUB (blue, left) and PDF (orange, right), respectively (1-not difficult to 5-very difficult)

[1]

Book4All attempts to semi-automate the conversion process as far as possible, but still offers the option of post processing the resulting markup as shown in figure 2.1. While this is optional, important parts like alternative text for images can only be added like this. The markup is in Intermediate Book Format (IBF), which is based on XML. Without basic knowledge of XML, post-processing becomes much more difficult.

Bartalesi and Leporini [1] also carried out an online survey in a follow-up work and asked 25 users to rate "enriched" EPUBs created in Book4All in comparison to the original PDF format regarding accessibility and usability. 50% of respondents preferred EPUB over other e-book formats, while 13% said EPUB was equivalent. The sample group also felt that it was easier to access content in EPUBs and use the table of contents than in PDFs. Furthermore, 80% of blind users were unable to read images in PDFs correctly with their screen reader, while the corresponding value for EPUBs was less than 50%. 64% of users found the EPUB's document structure easy to understand. Users also reported that it is much easier to access content in the EPUB than in a PDF, as shown in figure 2.2. The results show that EPUB is suitable format for blind people, if the EPUB contains proper tagging.



## 3. EPUB Document Standard

### 3.1 Requirements

One requirement of the new standard is that incorporates a way to switch between three versions, them being:

- Blind
- Visually impaired
- Normal/sighted

Switching between three versions should be simple and visible. It should not be hidden behind menus and settings and must be part of the document itself. A document of the standard must conform to EPUB 3 and not rely on external programs to switch between the versions.

The specifications of each version will be compared with each important component of a document.

#### 3.1.1 Switching between versions

Initially the intent was to find one new standard so that EPUB documents can be accessible. Unfortunately, creating a single standard was not possible due to not all EPUB readers supporting all EPUB 3 features.

##### 3.1.1.1 JavaScript

JavaScript is used to make websites interactive, so it was the logical option to dynamically change the versions. The coding was also quite simple and only involved replacing a single line in the CSS file with programming. This was easy to do in JavaScript, and testing the individual XHTML files in the EPUB showed that it was successful.

Shown in 3.1 is style.css which just imports the rules of a CSS file. The methods `selectVisible ()`, `selectImpaired ()` and `selectBlind ()` in script.js all function in the same way. To

```
@import url("../Styles/visible.css");
```

Figure 3.1: Contents of style.css

```
<head>
<title></title>
<link rel="stylesheet" href="../Styles/style.css"/>
<script src="../Misc/script.js"></script>
</head>

<body epub:type="frontmatter" onload="storageCSS();">
<a id="a1" href="#a1" onclick="selectVisible();">Normal</a>
<a id="a2" href="#a2" onclick="selectImpaired();">Visually impaired</a>
<a id="a3" href="#a3" onclick="selectBlind();">Blind</a>
</body>
```

Figure 3.2: The links in the JavaScript version

remember the version chosen, local and session storage is used. First they check if local storage is available. If it is then the method attempts to get the item `accessibleEPUBcurrentCSS`. If it does not exist, then it is created. This item is then set with the same import line in figure 3.1, only with different CSS file name for each version. If local storage is not available the same process is repeated with session storage. Local storage is preferable to session storage as changes are kept even after the program has been exited. This is not the case with session storage so the user has to set version again at the start of the program. Afterwards the method `loadCSS()` is called which deletes the CSS rule of a XHTML file and adds the rule set by the earlier methods.

The content of the XHTML file responsible for the switching mechanism, `Version-Changer.xhtml`, is shown in figure 3.2. It is important that both `style.css` and `script.js` are referred to, or the switch mechanism would be unsuccessful. In the body of the XHTML document, the method `storageCSS()` is called whenever the XHTML file is loaded, so whenever the XHTML are switched back to it from another. This method checks if local and then session storage are available. If either one is available then it sets the item in the storage and calls `loadCSS()`.

There are some issues with JavaScript version changer. If the software or hardware reader does not support temporary storage, the EPUB is always displayed in its default appearance. Due to this, switching with JavaScript is not a reliable option. As mentioned in chapter 1, JavaScript does not have to be supported by all EPUB readers. Nevertheless, the JavaScript implementation supports multiple XHTML files, because temporary storage is supported. For the same reason table of contents are also supported, as they are normally displayed in a separate file, named `nav.xhtml`.

### 3.1.1.2 CSS

The second method does not use JavaScript and uses some advanced features of CSS, introduced in CSS 3. This mainly refers to CSS selectors. Selectors allow HTML documents to have limited interactive capabilities, such as changing the appearance of a clicked element.[18]<sup>1</sup>

Before showing how the selectors work, the format of the content file should be discussed. It is shown in figure 3.3. Unlike the JavaScript version, the links are in the same XHTML

<sup>1</sup>[https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp)



```

<body>
<a class="versionChanger" id="a1" href="#visible">Normal</a>
<a class="versionChanger" id="a2" href="#impaired">Visually impaired</a>
<a class="versionChanger" id="a3" href="#blind">Blind</a>

<div style="padding:none" id="impaired" class="impaired">
...
</div>
<div id="blind" class="blind">
...
</div>
<div id="visible" class="visible">
...
</div>
</body>

```

Figure 3.3: The links and divs in the CSS version

```

.visible {
display:inline;
}

.impaired:target ~ .visible {
display:none;
}

.blind:target ~ .visible {
display:none;
}

.impaired {
display:none;
}

.blind {
display:none;
}

.impaired:target{
display:inline;
}

.blind:target{
display:inline;
}

```

Figure 3.4: The CSS selectors responsible for the dynamic version switching

file and are shown at the beginning of the EPUB file. In the JavaScript version, the links are on a separate page.

The code shown in figure 3.4 is responsible for the switching mechanism. At first only the visible version can be seen and the other two are hidden. The versions appear with the `:target` selector, which affects the appearance of the element if it was a target of a link (`<a> element`). If another link was clicked, then the current version becomes hidden again. However, the visible version would not become hidden, and this was a major problem. CSS selectors have limited capabilities compared to JavaScript, because they were intended to complement and not replace each other. After a bit of experimentation with various CSS selectors, the `~` selector delivered the desired results. `~` allows every element on the right hand side to be selected if it preceded by a left hand side element. So in the case of `.impaired:target ~ .visible`, once the impaired link has been clicked, the visible sections becomes hidden.



Figure 3.5: Comparison of serif(left) and sans serif(right) fonts

Source: <https://cdncms.fonts.net/images/6bff0c2cdbbcca14/A.SerifSansPrint.jpg>

## **Lorem Ipsum**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vel placerat felis, ut scelerisque lectus. Integer tristique ipsum a rutrum molestie. Curabitur ultricies lectus auctor, ultrices arcu a, facilisis risus. Nullam in sem orci. Duis porta pulvinar lectus, vitae ultrices nunc lacinia at. Donec sed fermentum ligula. Praesent ligula est, dignissim ut facilisis tempus, pellentesque auctor dui. Integer rhoncus tristique arcu nec facilisis. Suspendisse id metus in leo dictum sollicitudin. Praesent consectetur nunc eget lacus euismod, non accumsan erat condimentum. Sed imperdiet pellentesque iaculis. Vivamus ullamcorper rutrum venenatis. Proin commodo leo vitae metus consectetur, consequat varius ligula condimentum. Pellentesque sed faucibus massa. Sed non nisi at ante congue vulputate.

Figure 3.6: Text of the visually impaired version

### **3.1.2 Text**

The blind version does not have to have any special requirements for text, and its appearance will be identical to the normal version. The text size could be made much smaller, but leaving it at a standard size will allow other people to also read the text, such as for proofreading of the blind version.

The visually impaired version needs to use sans-serif fonts, as seen in figure 3.5, instead of serif fonts, as it is easier to read and identify individual characters.[8] Furthermore, the font should be larger than the normal version.

The font of the normal version can be any font, but for this standard a serif font was chosen. Most importantly, the text should not go beyond the edge of the screen so that horizontal scrolling is not required. All normal text in the document will be between `<p>` and `</p>` tags. A comparison of the visually impaired and normal version can be seen in figures 3.6 and 3.7.

### **3.1.3 Images**

All images need to have alternative text to describe them and preferably have a title given to them. This will help blind users as the screen reader will read out both of them.

# Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vel placerat felis, ut scelerisque lectus. Integer tristique ipsum a rutrum molestie. Curabitur ultricies lectus auctor, ultrices arcu a, facilisis risus. Nullam in sem orci. Duis porta pulvinar lectus, vitae ultrices nunc lacinia at. Donec sed fermentum ligula. Praesent ligula est, dignissim ut facilisis tempus, pellentesque auctor dui. Integer rhoncus tristique arcu nec facilisis. Suspendisse id metus in leo dictum sollicitudin. Praesent consectetur nunc eget lacus euismod, non accumsan erat condimentum. Sed imperdiet pellentesque iaculis. Vivamus ullamcorper rutrum venenatis. Proin commodo leo vitae metus consectetur, consequat varius ligula condimentum. Pellentesque sed faucibus massa. Sed non nisi at ante congue vulputate.

Figure 3.7: Text of the normal version

Usually the alternative text will be set as a property and the screen reader will have to identify the image. However, there was an issue. In some programs the screen reader does not read out the alternative text, so a different way had to be found to display it. The image will be part of a figure element of HTML5, shown in figure 3.10. A figure can also contain a caption, which will be identified by the screen reader as such, seen in figure 3.9 Furthermore, the alternative text will be inserted as a paragraph with the `<p>` element. It will remain invisible in normal and visually impaired mode, but will appear as normal text in blind mode. The text should be surrounded by specific tags, like `<Image>` or `<Graph>` shown in figure 3.8. It also has the CSS class "transparent", which only appears in blind mode.

Images in visually impaired and normal mode will be displayed normally, but the image, caption and figure will all have a maximum length of 100% of the screen size.

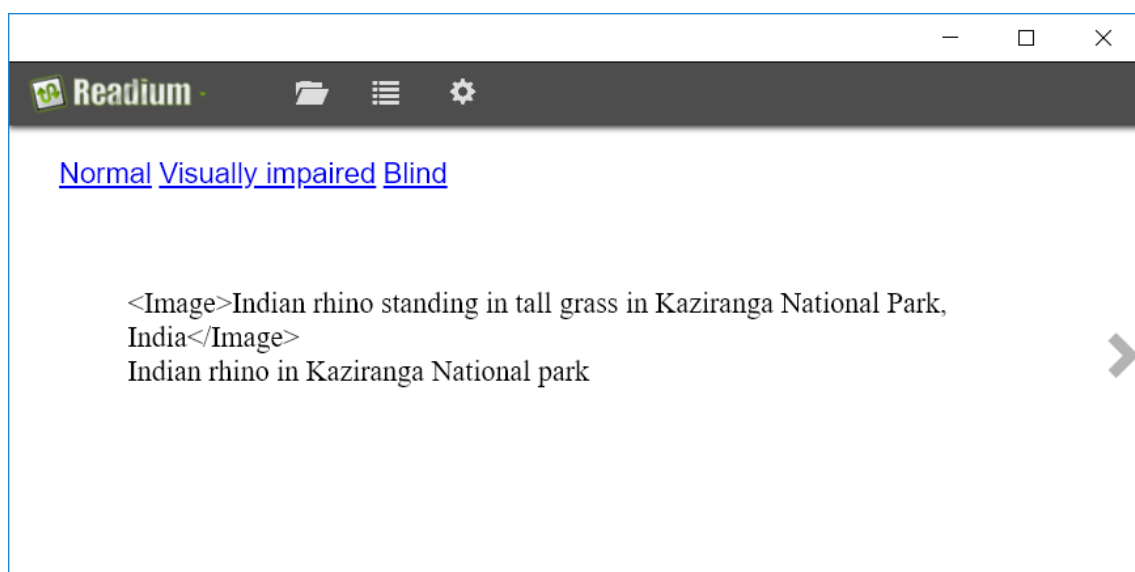


Figure 3.8: Image in 'Blind' mode

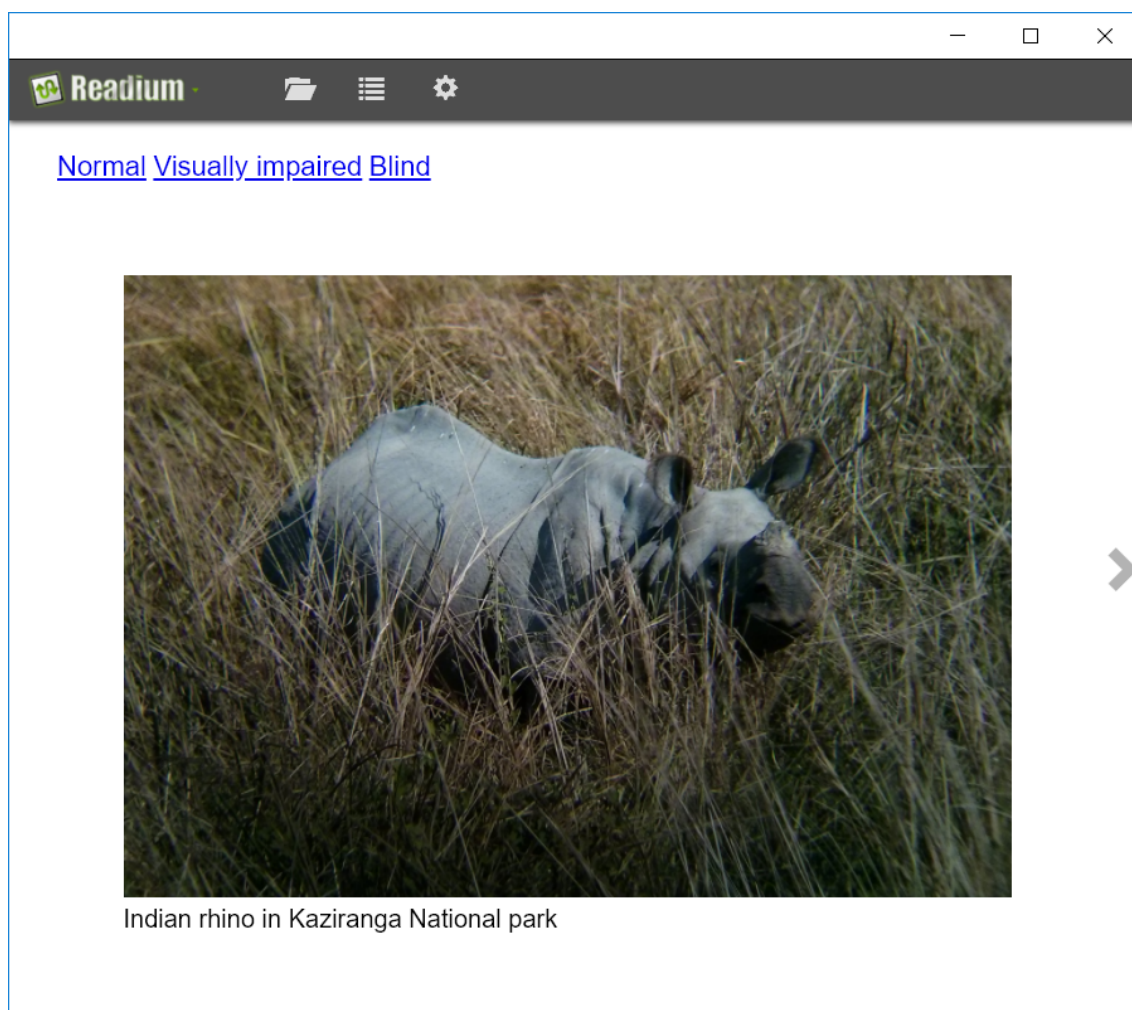


Figure 3.9: Image in 'Visual impairment' mode

```

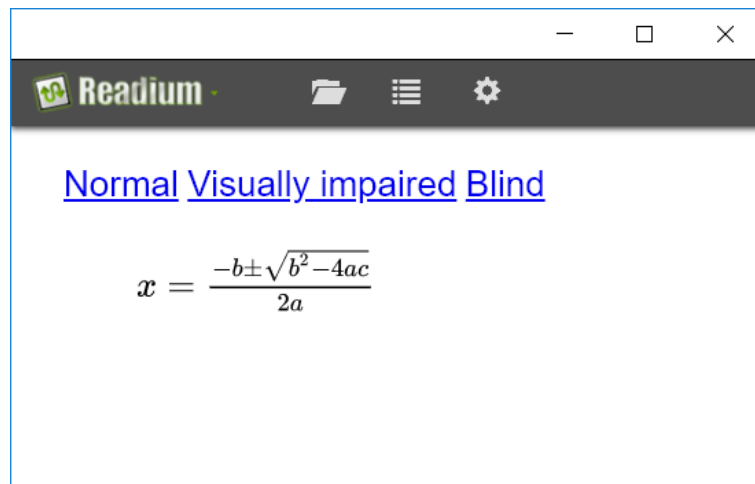
<figure>
  
  <p class="transparent">
    &lt;Image&gt;Indian rhino standing in tall grass in
      Kaziranga National Park, India&lt;/Image&gt;
  </p>
  <figcaption>
    Indian rhino in Kaziranga National park
  </figcaption>
</figure>

```

Figure 3.10: Code of figure 3.9 and 3.8

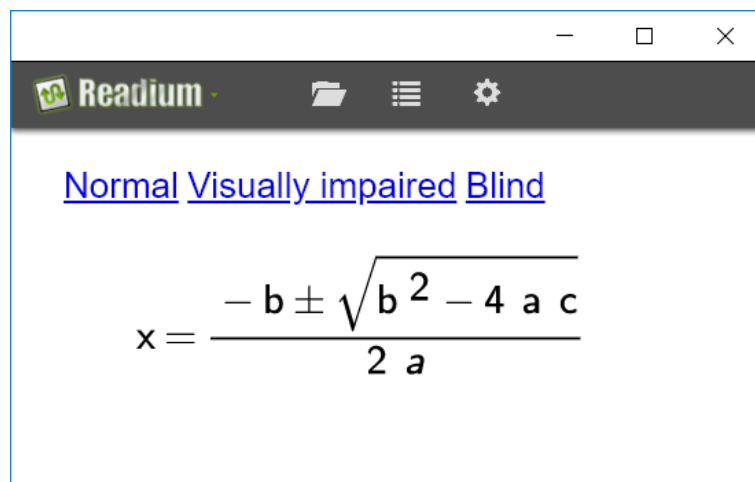
### 3.1.4 Mathematical formulas

Formulas should be displayed in MathML. MathML is quite clunky and time consuming to write in, so a LaTeX to MathML converter has to be used. For the sample documents, an online converting tool was used. The input formula was the quadratic equation, which appeared in 1.8.



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 3.11: Equation in 'Normal' mode



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 3.12: Equation in 'Visual impairment' mode

In the normal version the default display style is used, which uses a serif font and a combination of italicized and unitalicized letters, as shown in figure 3.11. Serif fonts are not suitable in the visually impaired version, so a sans serif had to be chosen. At first the font was changed in the CSS. This unfortunately resulted in the text not scaling properly, and therefore some characters were not properly spaced. The root symbol also wasn't displayed clearly. Instead the mstyle attribute of MathML had to be changed to sans serif which resulted in figure 3.12. The mstyle attribute can not be changed in CSS, it has to be inserted into every math element.

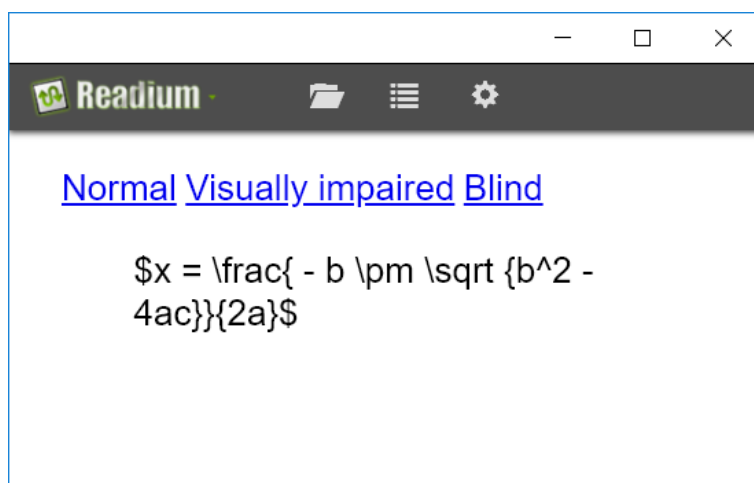


Figure 3.13: Equation in 'Blind' mode

In the blind version, the text has to appear as LaTeX code. Much like the alternative text for images, it will appear as a separate paragraph enclosed in `<p>` tags. To signify that it is code, it has to be surrounded by `$` signs, as shown in figure 3.13. The math element is in a `<div>` with CSS class "math", which is hidden in the blind version.

```

<figure>
  <div role="math" class="math">
    <math xmlns="http://www.w3.org/1998/Math/MathML" id="Formula" title
      ="Quadratic Formula" alttext="{x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}}">
      <mstyle>
        <semantics>

          <mrow>
            <mi>x</mi>
            <mo>=</mo>
            <mfrac>
              <mrow>
                <mrow>
                  <mo>-</mo>
                  <mi>b</mi>
                </mrow>
                <mo>\pm</mo>
                <msqrt>
                  <mrow>
                    <msup>
                      <mi>b</mi>
                      <mn>2</mn>
                    </msup>
                    <mo>-</mo>
                    <mrow>
                      <mn>4</mn>
                      <mi>a</mi>
                    </mrow>
                  </mrow>
                </msqrt>
              </mrow>
              <mrow>
                <mn>2</mn>
                <mo></mo>
                <mi>a</mi>
              </mrow>
            </mfrac>
          </mrow>

        </semantics>
      </mstyle>

    </math>
  </div>

  <p class="transparent">
    $x = \frac{ - b \pm \sqrt {b^2 - 4ac}}{2a}$
  </p>
</figure>

```

Figure 3.14: Code of figures 3.11 and 3.13

```

<mstyle scriptsize="1" lspace="20%" rspace="20%"
  mathvariant="sans-serif">

```

Figure 3.15: Addition to the code shown in 3.14 for the visually impaired version to make the formula appear as sans serif and not in italics





## **4. Evaluation EPUB Standard**

**4.1 Software readers**

**4.2 Hardware readers**

**4.3 Discussion: what do developers have to do?**



## 5. AccessibleEPUB Editor

The AccessibleEPUB editor will be presented with each Windows form covering a separate section.

### 5.1 Requirements

There are a variety of requirements which have to be fulfilled the editor. Most importantly, it has easy to use and not require any programming knowledge to use, except for LaTeX code for equations. If no equations are needed in the document, then there must be programming ability asked of the user.

### 5.2 Programming language

The first question was in which programming language should the editor be implemented in. C# and Java were picked early as the two main options, as both of them are object oriented and natively support forms. Furthermore, both support the ability to make the programs themselves accessible for blind users. C# has several accessibility properties, like AccessibilityDescription and AccessibilityName, which are passed to the screen reader. Java uses the Java Accessibility Bridge which makes it accessible to screen readers. Java is platform independent, and while C# programs can run on Mac OS and Linux, it relies heavily on Windows and its features. However, Java is not already installed on any operating system, while C# programs can run on Windows machine with only .NET as prerequisite. The target .NET version of the editor is contained in Windows 10. Therefore the editor was programmed in C#, as the users were predominantly Windows users and don't have to install prerequisites.

### 5.3 Main window with editor and preview

#### 5.3.0.1 HTML Editor

The main window required the most programming and therefore also had its fair share of problems. The first issue was regarding the editor on the left hand side in figure 5.1. At first a What-You-See-Is-What-You-Get(WYSIWYG) HTML editor was not intended,

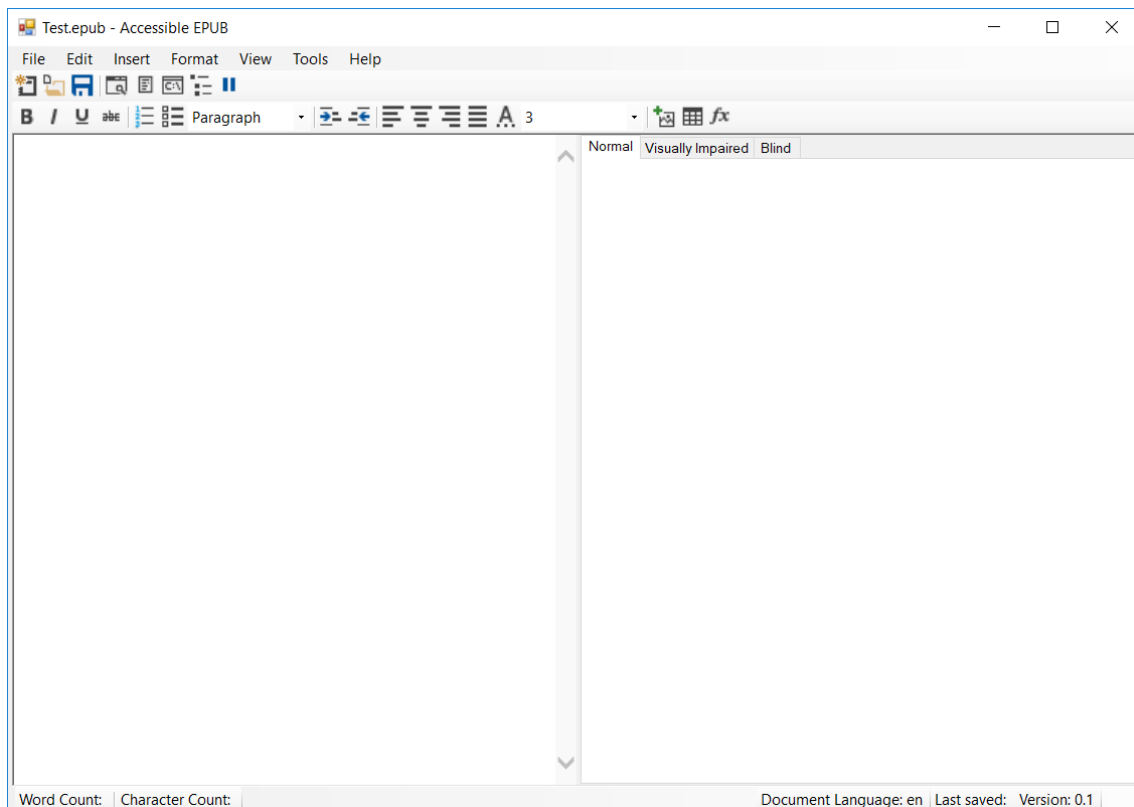


Figure 5.1: Main window with HTML editor and JavaScript switching preview

as it hides semantic information about the EPUB. The initial approach involved editing semantic information of XHTML elements, like images, and changing things like the `src`, `alt`, etc. This was still in the early, rough stages and the approach seemed too complicated, so the WYSIWYG editor was chosen instead.

The second issue was how to implement the WYSIWYG HTML editor. Programming it from scratch would not be possible in the time allocated to a bachelor thesis, as it would involve creating a browser engine. Fortunately, the inbuilt browser in C#, `WebBrowser`, allows editing with just a few lines of code. The `WebBrowser` control is based on Internet Explorer and displays web pages like it. It uses Internet Explorer version 6 as default, which does not display certain CSS properties such as `max-width` and cannot display SVGs. To fix this issue, some code has to be run when the form has loaded which determines the Internet Explorer version on the computer and loads the newest one to the HTML editor. After this SVGs and the CSS properties functioned as desired.

A major issue with the HTML editor is that content written in it is in HTML, while EPUB requires XHTML. While there aren't major differences, there are some small ones such as independent tags like `<br>` have to be self closing and written as `<br/>` in XHTML. If this is not done the EPUB reader will specify an error. Therefore a tool has to convert the HTML code to XHTML. There are several packages available, one of them being `HtmlAgilityPack`<sup>1</sup>. It can convert HTML and XHTML and also correct HTML parsing errors such as not closed tags. `HtmlAgilityPack` functioned well and as expected at first. The resulting code was in XHTML. However, there soon was an error. Many Greek signs,

<sup>1</sup><http://html-agility-pack.net/>

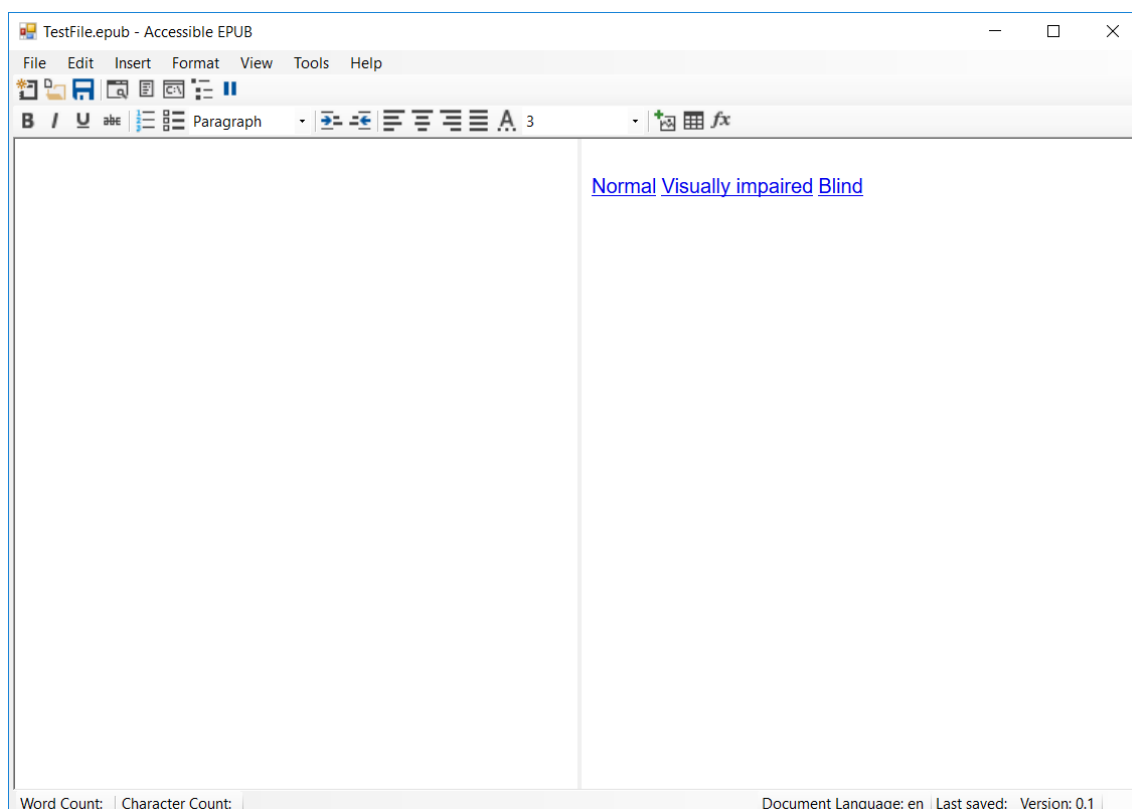


Figure 5.2: Main window with HTML editor and CSS switching preview

such as the sign " $\wedge$ ", were displayed as question mark. This meant HtmlAgilityPack could not be used.

Another available package was TidyManaged<sup>2</sup>, was also used, but it did not convert the code properly to XHTML. The third attempt involved using SgmlReader<sup>3</sup>, which converts SGML content, like HTML, to XML content, like XHTML. It successfully parsed the HTML and was able to convert mathematical signs properly. It did not create a XML declaration at the top of the document, but this was simpler to solve. A standard XML declaration was just added to the document and the new XHTML document was properly rendered by the browser.

### 5.3.0.2 Preview browser

On the right hand side of the main window is the preview browser. It should display XHTML page in each of three versions(normal, visually impaired, blind). Since the HTML editor uses the WebBrowser control, it would have been easiest to use it on the right side too. However, Internet Explorer is unable to display MathML. Instead of showing the quadratic equation, as shown in figure 1.8, it showed figure 5.3. As a result, another browser had to be found. Only two browsers are able properly show MathML, Mozilla Firefox and Safari by Apple. The Firefox engine, Gecko, was chosen as it is open source and had a C# browser package. After inserting it in the form MathML was displayed properly.

<sup>2</sup><https://github.com/markbeaton/TidyManaged>

<sup>3</sup><https://github.com/lovettechris/SgmlReader>

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Figure 5.3: MathML depiction of the quadratic equation in Internet Explorer

As seen in figures 5.1 and 5.2, there is a small difference between the user interface of the JavaScript and CSS versions. The CSS version only uses one browser tab as preview, while the JavaScript version uses three. This is mentioned in chapter 3 and is due to the JavaScript version having a separate file named VersionChanger.xhtml, which handles switching versions. In the CSS version the whole content is in one XHTML file. Consequently, the CSS version is very easy to show in the preview. Only the three links shown in figure 3.3 have to be added to HTML editor content and file is done.

This is much more complicated in the JavaScript version, since the default display style will always be shown. The CSS of the blind and visually impaired version had to be changed. The initial attempt to solve this problem was done by changing the CSS of each browser in a tab. Unfortunately, even after changing the CSS of a gecko browser instance, it still displayed the default one.

pandoc 

## **6. Conclusion and Future Work**





# Bibliography

- [1] Valentina Bartalesi and Barbara Leporini. "An Enriched ePub eBook for Screen Reader Users". In: *Universal Access in Human-Computer Interaction. Access to Today's Technologies*. Ed. by Margherita Antona and Constantine Stephanidis. Cham: Springer International Publishing, 2015, pp. 375–386. ISBN: 978-3-319-20678-3.
- [2] Deutsche Zentralbücherei für Blinde (German Central Library for the Blind). *Was ist DAISY? (What is DAISY?)* [https://www.dzb.de/index.php?site\\_id=7.8](https://www.dzb.de/index.php?site_id=7.8), visited on 09.03.2018.
- [3] Verband für Blinden-und Sehbehindertenpädagogik e. V. (VBS) (Association for teaching for blind and visually impaired students). *Augenbit Wiki*. <http://www.augenbit.de/wiki/index.php?title=Hauptseite>, visited on 09.03.2018.
- [4] Antonello Calabrò, Elia Contini, and Barbara Leporini. "Book4All: A Tool to Make an e-Book More Accessible to Students with Vision/Visual-Impairments". In: *HCI and Usability for e-Inclusion*. Ed. by Andreas Holzinger and Klaus Miesenberger. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 236–248. ISBN: 978-3-642-10308-7.
- [5] DAISY Consortium. *Baseline for Accessible EPUB3*. [www.daisy.org/baseline](http://www.daisy.org/baseline), visited on 29.01.2018.
- [6] Matt Garrish and Markus Gylling. *EPUB 3 Best Practices*. O'Reilly, 2013. ISBN: 1449329144, 9781449329143.
- [7] John Gruber. *Markdown*. <https://daringfireball.net/projects/markdown/>, visited on 08.03.2018.
- [8] University of Hagen. *PDF- und Word Dokumente barrierefrei umsetzen (Making accessible PDFs and word documents)*. [https://www.fernuni-hagen.de/barrierefrei/pdf\\_word.shtml](https://www.fernuni-hagen.de/barrierefrei/pdf_word.shtml), visited on 09.03.2018.
- [9] IDPF. *EPUB 3 Changes from EPUB 2.0.1*. <http://www.idpf.org/epub/30/spec/epub30-changes.html>, visited on 29.01.2018.
- [10] IDPF. *EPUB 3 Specification*. <http://www.idpf.org/epub/301/spec/epub-overview.html>, visited on 29.01.2018.
- [11] IDPF. *EPUB 3.1 Changes from EPUB 3.0.1*. <http://www.idpf.org/epub/31/spec/epub-changes.html>, visited on 09.03.2018.
- [12] IDPF. *Open Publication Structure (OPS) 2.0.1 v1.0.1 Recommended Specification*. [http://www.idpf.org/epub/20/spec/OPS\\_2.0.1\\_draft.htm](http://www.idpf.org/epub/20/spec/OPS_2.0.1_draft.htm), visited on 09.03.2018.
- [13] IDPF. *Understanding EPUB 3*. <http://epubzone.org/epub-3-overview/understanding-epub-3>, visited on 29.01.2018.

- [14] John MacFarlane. *Pandoc document converter*. <http://pandoc.org/>, visited on 08.03.2018.
- [15] WebAIM (Web Accessibility in Mind). *World Laws: Introduction to Laws Throughout the World*. <https://webaim.org/articles/laws/world/>, visited on 30.01.2018.
- [16] Jens Voegler, Jens Bornschein, and Gerhard Weber. "Markdown – A Simple Syntax for Transcription of Accessible Study Materials". In: *Computers Helping People with Special Needs*. Ed. by Klaus Miesenberger et al. Cham: Springer International Publishing, 2014, pp. 545–548. ISBN: 978-3-319-08596-8.
- [17] W3C. *Web Content Accessibility Guidelines (WCAG) 2.0*. <https://www.w3.org/TR/WCAG20/>, visited on 29.01.2018.
- [18] W3Schools. *CSS Selectors Reference*. [https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp), visited on 15.03.2018.