Accessible Charts With AChart Reader

Group 1

Lukas Bodner, Daniel Geiger, and Lorenz Leitner

706.057 Information Visualisation SS 2020
Graz University of Technology

29 June 2020

Abstract

Scalable Vector Graphics (SVG) charts cannot be read out by screen readers used by visually impaired users. They need additional internal markup, which describes all the important elements of a chart. Accessible Rich Internet Applications (ARIA) does exactly that. AChart Reader uses this ARIA markup to produce a textual summary of a chart and to read out the chart for visually impared users to easily unterstand a chart and its meaning, as well as enables users to explore the chart in an interactive way by navigating through the elements by means of key presses.

Contents

List of Figures

List of Listings

Chapter 1

Introduction

This chapter explains SVG charts and how they can be marked up to be more accessible.

1.1 SVG Charts

Scalable Vector Graphics (SVG) graphics are vector-based graphics. SVGs can be used for icons, simple images or charts. Due to the vector-based nature, they compress well and are small in file size. They can also easily scale up and down without any loss in image quality.

SVGs can be created by many different programs or they can be written manually and be opened in any web browser.

1.2 Accessible Charts

SVG charts are hard to understand for screen readers. They do not use the `alt` tag as in Hypertext Markup Language (HTML), which helps screen readers understand what is displayed in an image. Charts need to have additional code, which describes the image for a screen reader.

1.3 ARIA

Accessible Rich Internet Applications (ARIA) [W3C 2020] provides the `role` property, as well as many others, to describe the elements of an SVG chart. The role of an element signifies its use in the SVG chart. There are different roles for:

- Chart and data areas
- X and Y axes and labels
- Data set, series, points and values

There is also the ARIA property `aria-labelledby`, which can be used to reference other elements such as labels and data points.

1.4 Example Chart

In Figure 1.1 an example chart is shown. It shows the three days Friday to Sunday on the x-axis and on the y-axis the number of finished tasks during these days.

An excerpt of the corresponding SVG code of the example chart is shown in Listing 1.1. There are different roles for the chart, heading, chartarea, xaxis, yaxis, dataset and datapoints as well as the datavalues. For the datavalue “8 Tasks” there is also the tag `aria-labelledby` which references the x-axis label.

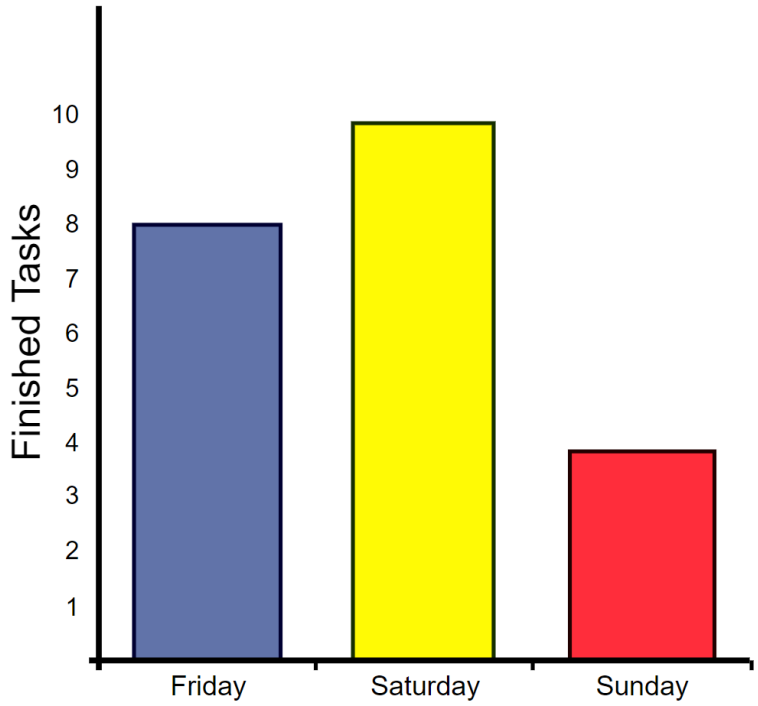

Figure 1.1: An example of a simple SVG Chart, that is marked up with ARIA. [Chart created by creators of AChart Reader.]

```
<g id="ChartRoot" font-family="Arial" role="chart" aria-charttype="bar"
    tabindex="0">
    <title role="heading" >Finished tasks per day</title>
    <g>
        <rect role="chartarea"  x="65" y="35" width="445" height="335"></rect>
    </g>
    <g id="xScale" role="xaxis" aria-axistype="category"  tabindex="0">
    <title role="heading" >Days</title>
    <text role="axislabel"  id="x-Friday" x="121" y="388">Friday</text>
...

    <g id="dataarea" role="dataset" >
    <g id="datapoint_0" role="datapoint"  tabindex="0" fill="rgb(97, 115, 169)">
    <title role="datavalue" aria-labelledby="x-Friday">8 Tasks</title>
    <path d="M83,147 H158 V370 H83 V147"></path>
...
```

Listing 1.1: Shows the code of the SVG of the Example Chart.

1.5 Inspiration - Describler

Describler [Schepers 2020] is a prototype to read out SVGs. An SVG file can be uploaded and when pressing *tab* or *arrow* keys, it can be navigated through the SVG file and the website will read out the ARIA markup.

Chapter 2

AChart Reader Original Version

The basis of this project is the AChart Reader web application developed by master student Christopher-Alexander Kopel. The master thesis includes the implementation of the AChart Reader application. The matter that Christopher is visually impaired did not hinder him to implement the application. In fact, this actually is the motivation behind the thesis.

The goal of this project is to (1) evaluate the original application (2) analyse it regarding functionality and usability (3) identify and fix implementation errors. In Chapter 2 the original functionality and behaviour of the application is described.

2.1 Main Functionality

The main task of AChart Reader is to imitate the behaviour of a screen reader. What a screen reader can do for native and web applications, AChart Reader can do for Scalable Vector Graphics (SVG).

2.1.1 Web Application

The AChart Reader application was initially implemented as a web application. While this was fine during the beginning of the project, the improvement of the user interface and usability was target through this project. The goal was to let the application look more like a native and standalone version rather than a website.

2.1.2 Graphic Panel

The graphic panel is responsible for the visual representation of the SVG file. It is the first component that will be generated after selecting or uploading a SVG file. Figure 2.1 show the graphic panel below the introduction text. While this panel is not important for the visually impaired user, it serves as a starting point for developers which test the correct markup of the SVG file.

2.1.3 Reader Panel

After the generation of the graphic panel all the required information is uploaded to the web server. The next step is to extract the textual information of the SVG file added by the ARIA markup. This information enables the generation of the textual summary. Figure 2.2 shows the reader panel. The reader panel has a tree-like structure which supports navigation using the tabulator and arrow keys.

2.1.4 Speech Synthesizer

Whenever a user navigates through the reader panel, AChart Reader will speak the textual content of the visited node out loud. This is done using a speech synthesizer. In this sense, AChart Reader acts as a screen reader. However, at the initial state the selection and workflow of the synthesizer was not bug-free.

2.1.5 Synchronization

AChart Reader supports two ways of interacting with the application. Firstly, it is possible to navigate through the chart using the navigation keys. Secondly, it is also possible to click on the SVG graph itself. During the navigation the currently highlighted node should be synchronized on both panels. So whenever a user clicks on the graphic panel it should sync the reader panel and vice versa. However, initial inspection of the tool revealed synchronization problems. Therefore, one task was to improve and fix the synchronization issues.

2.2 Additional Functionality

In addition to the screen reader functionality, AChart Reader does also provide further interesting features. These features help a visually impaired user to get a better understanding of the chart and its included data.

2.2.1 Statistics

During creation of the reader panel the application computes some statistical values. The list of values includes some general information, for instance the number of data points, or some statistical values, for example the sum and the average of all data points or the median. The additional information is valuable and enables another source of information for the user.

2.2.2 Sorting

Furthermore, the reader panel supports sorting of the data set. Initially the reader panel displays the data in its original order. After the initiation it is possible to sort the data points either in ascending or descending order. Note that the ordering does only affect the order of the enumerated data points inside the reader panel. The visual representation of the SVG file inside the graphic panel will remain the same. While this information may be obvious when looking at the chart, again, this feature aids a visually impaired user.

2.3 Purposes

The implementation of AChart Reader serves several purposes. The main target groups are developers and visually impaired users.

2.3.1 Developers

As already described in the introduction, the SVG files must consist of valid ARIA markup. Therefore, a developer of accessible charts can use the AChart Reader application to verify the correct syntax of the SVG file. Further, the textual summary of the reader panel helps to debug incorrect ARIA markup.

2.3.2 Users

The main purpose of AChart Reader is to aid visually impaired users. AChart Reader acts a a screen reader for accessible charts. Apart from the audio feedback, AChart Reader does also provide valuable information about the data set, a set of statistical values.

2.4 Example of Original AChart Reader

Figure 2.1 and 2.2 combine the original user interface of the AChart Reader application. The initial application shows the following components one below the other:

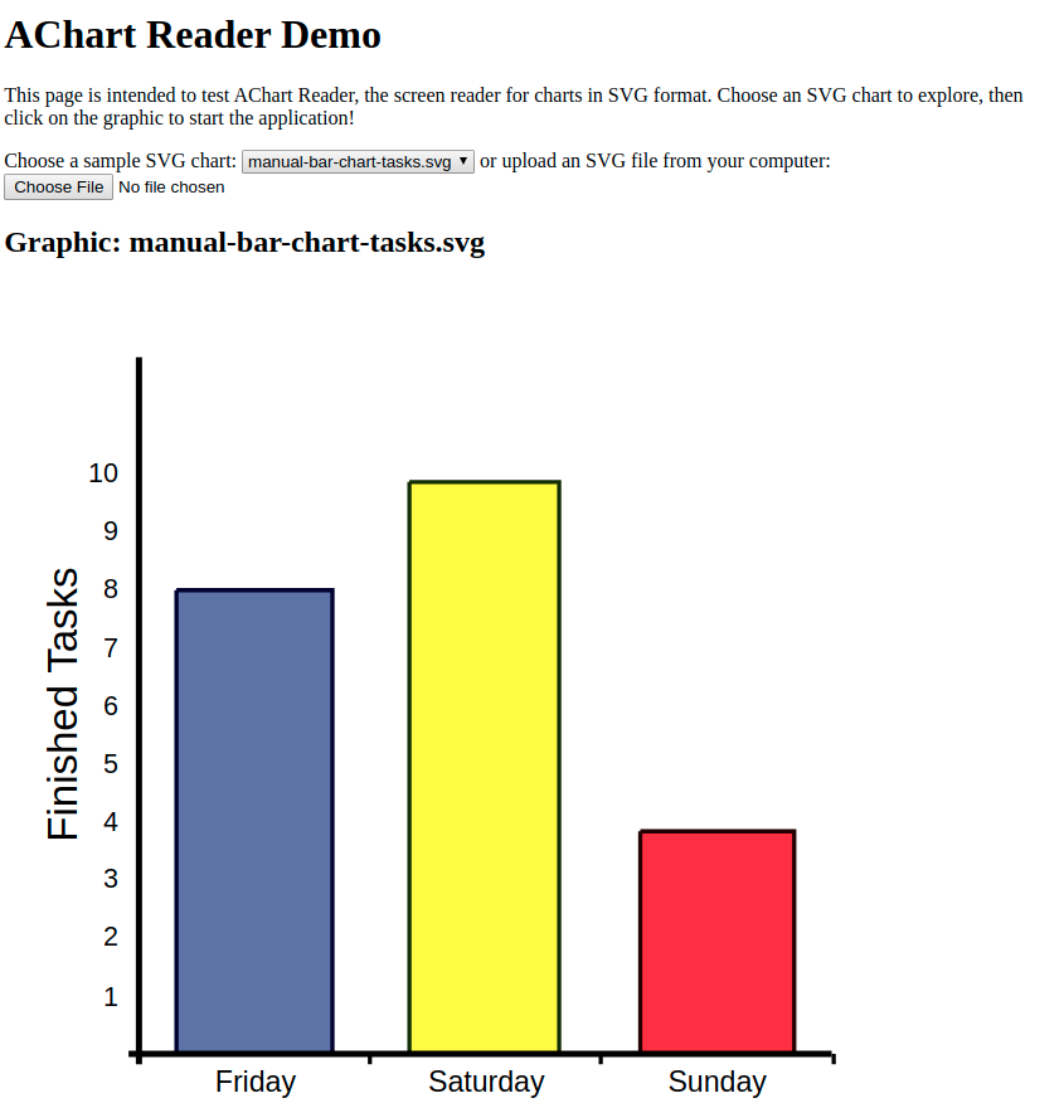

Figure 2.1: The top half of the user interface starts with an initial description of the application. It includes a list of sample SVG files and a button to upload custom SVG files. Below the introduction, the graphic panel shows the selected bar chart. [Screenshot taken by the authors of this report.]

- Application description
- File selection
- Graphic panel
- Reader panel

Most of the time the graphic and reader panel did not fit on the same browser page, due to the large resolution of certain SVG files. The future application will support a more responsive user interface, which uses the empty space of a wider screen better.

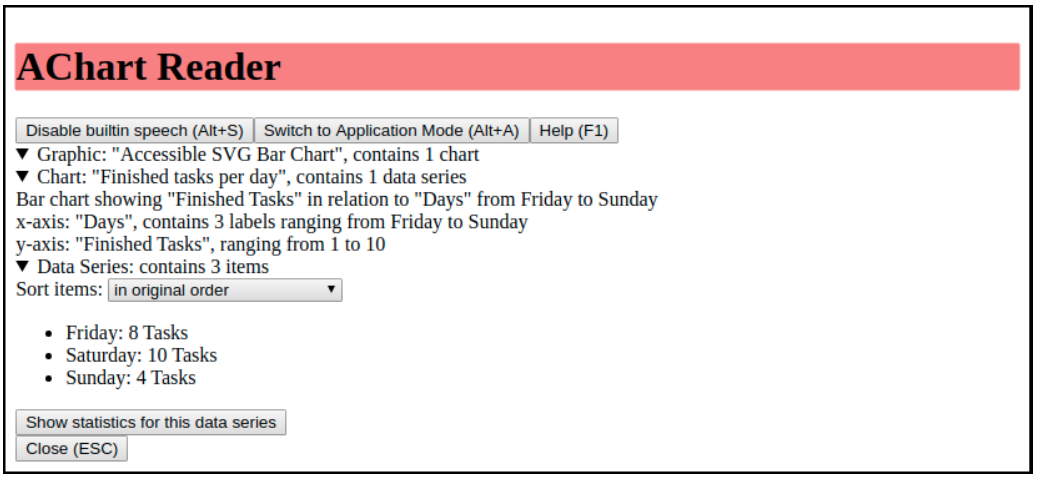

Figure 2.2: The bottom half of the user interface includes the reader panel. It includes the tree-like textual summary of the accessible SVG chart. [Screenshot taken by the authors of this report.]

Chapter 3

AChart Reader Changes

AChart Reader is an existing and ongoing project. During the Information Visualisation course, a task was given to make some improvements to the original version of AChart Reader, as it existed up to the point of the start of Information Visualisation. After the course is done, the changes are handed back to the original developer of AChart Reader, and if possible incorporated into the final product.

3.1 Main Changes

The following shows the changes that have the most impact on AChart Reader, regarding functionality or just visual aspects.

3.1.1 User Interface Overhaul

A main task was to improve the user interface of the original AChart Reader version. The original version very much looked like a static rigid web page. On top there was an intro text, the chart selection, below that the graphic, and below that, the reader panel.

It was rather obvious that the application was just a single "demo" web page, and therefore it was necessary to change the look and feel to be more reminiscent of an application rather than a website. Especially in the standalone versions of the program this would have stood out in an unpleasant way, as users do not like to be reminded that their binary desktop application is actually just a urlbar-less browser window running JavaScript and HTML.

3.1.1.1 Responsiveness

One of the first things that was changed was the fixed placing of top-to-bottom graphic and reader panel. Now, when the width is wide enough, the graphic panel will be on the left, and the reader panel will be on the right. If the width becomes narrower, the graphic panel will be on top, and the reader panel will be below that, accommodating for the narrower width.

3.1.1.2 Highlighting Changes

Part of the user interface is the way elements that are currently focused and read out by the synthesizer voice are highlighted. It used to be that all highlighted elements used the same rather jarring pink background color as highlighting. Now, the currently selected elements in the text view get a dotted outline highlighting, as well as the SVG elements, however on those, a mask or fill highlighting can also be chosen, as the most suitable one depends on the SVG chart.

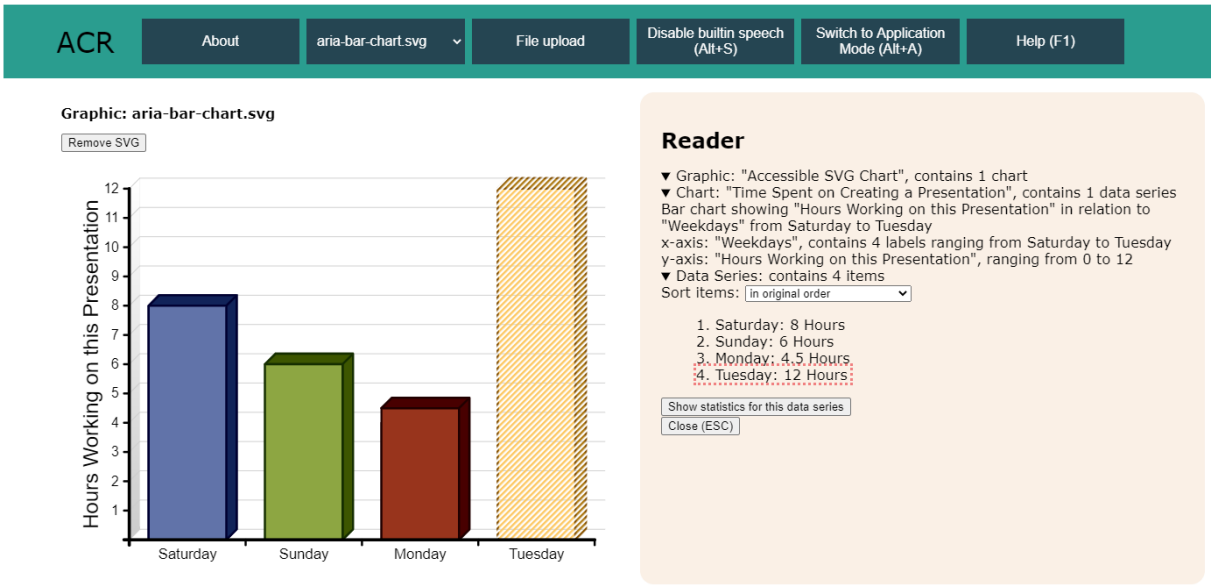

Figure 3.1: The new user interface shown with a larger width. The graphic panel is on the left and the reader panel is on the right. The header menu buttons are visible. [Screenshot taken by the authors of this report.]

3.1.1.3 Header Menu

Instead of the inline intro text and the SVG chart selection which was placed haphazardly on top of the graphic panel, there is now a header menu, with buttons for these functions. Also incorporated in the header menu are some of the option buttons of AChart Reader, which used to be in the reader panel itself, however, since they are global options and do not necessarily pertain to a specific open reader panel, the placement in the header menu seems more appropriate.

3.1.1.4 Font Changes

A rather small change but with a large impact was changing the font from the default serif browser font to a sans-serif version. This makes it far less obvious that it is actually just a web page that is run.

Figures 3.1 and 3.2 show the new user interface, with a larger and narrow width, respectively.

3.1.2 Synchronization

The synchronization of the graphic panel and the reader panel is a change with a focus on improved usability. As it used to be, clicking, selecting or focusing in any way by navigating to an element in the text in the reader panel was not reflected in the graphic panel in any way, at least not consistently. Navigating through the SVG elements in the graphic panel also did not trigger a readout of the text elements in the reader panel.

Now, whenever an SVG element like a data point is focused in the graphic panel, the corresponding text element in the reader panel is focused, highlighted and read out, and vice versa.

3.1.3 Internal Synthesizer Voice Selection

Initially, the voice readout audio did not work on all configurations. It was ascertained that this was due to the fact that a synthesizer must be installed in some way in a configuration for the audio to be heard. For example, if Google Chrome is used, which ships with its own proprietary Google voices, it works fine. If any browser on Windows is used, which comes with a synthesizer installed on the operating system, it also works fine. However, if a configuration without either of these installed synthesizers is

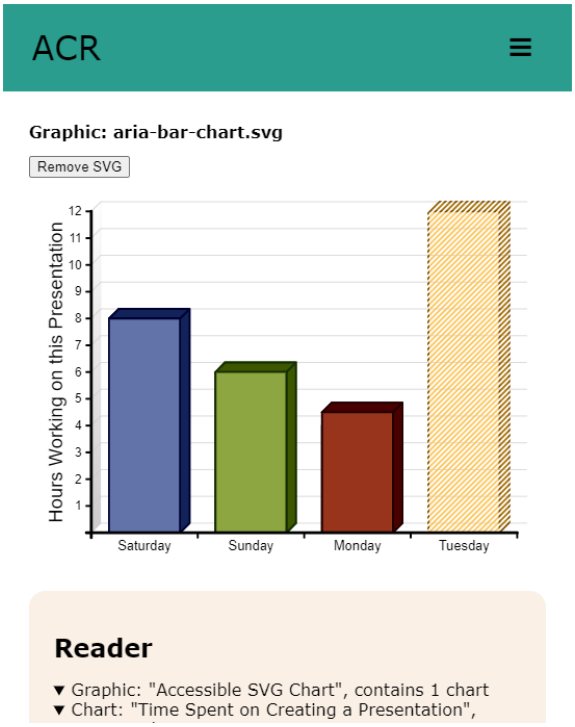

Figure 3.2: The new user interface shown with a smaller width. The graphic panel is on top and the reader panel is below it. The header menu buttons are not visible but can be faded in by clicking on the hamburger menu button. [Screenshot taken by the authors of this report.]

used, for example Firefox or Chromium on Linux, there will be no audible voice readout. In such a case, a synthesizer voice must be installed locally on the operating system. An example of such a local installation would include Speech Dispatcher [Free(B)Soft 2020] and eSpeak BG [eSpeak NG 2020].

What changed in AChart Reader itself was the selection of an appropriate voice out of all available voices. In specific, an English, ideally US or British English voice must be chosen and is now chosen, so that the English text elements that are read out do not sound garbled, as can be imagined if for example a German synthesizer voice tries to read out English text. If no English voice is found, the first available voice will be used, although it will sound less than ideal. For the future, the voice selection depends on a `language` variable which is currently hardcoded to English, however it could easily be set to something different, maybe by the user via a selection menu.

3.1.4 Standalone Version

A standalone desktop version of the browser-only-single-web-page version was created, for easier distribution and self-hosting without the need of a development setup or local web server, using Electron [OpenJS Foundation 2020]. Different version were created, as well as a development version, and scripts for creating these standalones for future releases.

3.1.4.1 Electron Development Version

Before any binary packages are created it is beneficial for a developer to run the “standalone” version in an Electron window to see whether everything is working correctly. This can be done using the locally installed Electron program, but still uses the development setup of AChart Reader and its files. Running `npm run electron` accomplishes this.

3.1.4.2 Binaries

If everything is working fine in the development version using Electron, binaries for different platforms can be created using `npm run create-packages-<platform>`, where `<platform>` is one of:

- `all` - All platforms
- `linux` - Linux
- `mac` - Mac
- `win` - Windows

These are already packaged binaries that can be run from any location on an independent operating system, without the need of having the development version and development dependencies of AChart Reader set up.

3.1.4.3 Installers

Using these binaries, installers for different platforms can be created as well, using `npm create-installer -<type>`, where `<type>` is one of these installer types:

- `deb` - Debian, Ubuntu
- `rpm` - Redhat, Fedora, CentOS
- `dmg` - Mac
- `win` - Windows

These scripts can be used by the developers that take on the project from this point on to create future releases for future versions of AChart Reader. Another automation script which removes all these binary files is `npm run clean`.

3.2 Other Changes

Some of the other more minor changes are described below, as well as additional information that pertains to some of these changes.

3.2.1 Automatic Sample Files

Originally, the drop-down menu of sample SVG charts was populated by a hardcoded list. Now, during build time, the list of files in the `samples` directory is used to generate that list automatically. Also fixed was the copying of that directory to the `build` directory during build time, which was missing.

3.2.2 No Automatic Pre-Selection of Image on Image Load

A small change. When first opening an image, it was automatically pre-selected and highlighted for some reason. It is not changed so that the image is not highlighted any more automatically when first opening it.

3.2.3 Generalization of Sample Files and User-Uploaded Files

There were some strange discrepancies regarding the choice of samples files or uploading own files. Only one sample chart at a time could be visible, but multiple charts of user-origin could be uploaded and visible. The code for these functions was also duplicated heavily. Now, both of these functions behave the same way, and use the same code. Only one chart can be visible simultaneously, and if a new chart is chosen, the existing chart is replaced.

3.2.4 Numbered List Instead of Unordered List

The data points used to be represented in an unordered list of bullet points. In itself that is fine, however, when sorting the items via the sort functions, the way in which they have been sorted is not reflected in an unordered list. Due to this, a numbered list is now used, which reflects the sorting after the fact.

3.2.5 Removal of Modes

The on-demand additional statistics about a data series featured an entry of modes, that is the value which occurs most often, which is rather uninteresting in most data series. Therefore, this entry was removed from the statistics function and display.

3.2.6 Instant Element Readout on First Open

As it used to be, when first pressing anywhere on the chart, the reader panel opened and read out "AChart Reader", focusing the title element at the top of the reader panel. Now, the element which is clicked on first is read out when the reader panel opens, for example, if a data point is the first element that is clicked, the reader panel opens and reads out, highlights, and focuses that element.

3.2.7 Legacy Naming Convention Removal

What is now referred to as the graphic and reader panel was not always referred to in this way. It used to be that the reader panel was called the `main_application` in the code and its title in the web page was "AChart Reader". This stemmed from some legacy reasons for the initial plan of the application to be an add-on that can be dropped into any web page with an SVG chart. However, in its current state of a more self-sufficient application, the separation into graphic and reader panel, and the removal of the "AChart Reader" title in the reader panel seems appropriate, as well as reflection in the code of these changes.

3.2.8 Placeholder Text when No Chart is Chosen

When the application is opened now it seems rather empty, as the intro text is hidden in the "About" button in the header menu. To assure the user that everything is worked as intended, as small placeholder text is placed beneath the header menu that prompts the user to select a chart to start the process. This placeholder text will vanish when a chart is visible, and reappear when a chart is removed, that is no chart is visible again.

3.2.9 Readme Reflection of Changes

Of course, all the changes regarding the usage and commands to build and run the application are reflected in the new version of the Readme file, which is mainly used by developers so that they can contribute to AChart Reader.

3.3 Showcase Video

A showcase video [Bodner et al. 2020] was produced that shows a live usage of AChart Reader, explaining how to use it and showing a deeper look into some of the functionality.

Chapter 4

Conclusion

AChart Reader can be used by any developer or person to test their ARIA marked-up SVG charts, but it can also be used by visually impaired users to understand already marked-up charts. AChart Reader could be further improved and also be implemented into a browser extension, which enables plugin capability to use AChart Reader on any web page that has an ARIA marked-up SVG chart embedded, to increase the reach of users. The function for detecting SVG content on a web page already exists in the code base. Future work to have some of the `role` properties and similar attributes standardized is needed. AChart Reader's ultimate goal is to make the web more accessible for all people.

Bibliography

Bodner, Lukas, Daniel Geiger, and Lorenz Leitner [2020]. *Showcase of AChart Reader*. 17 Jun 2020. https://youtu.be/7unJ2aU9ghc (cited on page 13).

eSpeak NG [2020]. *eSpeak NG*. 14 Jun 2020. https://github.com/espeak-ng/espeak-ng (cited on page 11).

Free(B)Soft [2020]. *Speech Dispatcher*. 10 Jun 2020. https://freebsoft.org/speechd (cited on page 11).

OpenJS Foundation [2020]. *Electron*. 24 Jun 2020. https://electronjs.org (cited on page 11).

Schepers, Doug [2020]. *Describler*. 24 Jun 2020. http://describler.com (cited on page 3).

W3C [2020]. *WAI-ARIA Graphics Module*. 27 Jun 2020. https://w3c.github.io/graphics-aria (cited on page 1).