Accessible Charts

Moritz Erlacher, Lisa Habich, Alexander Perko, Markus Stradner

706.057 Information Visualisation SS 2021
Graz University of Technology

10 May 2021

Abstract

This report summarizes the ARIA standard with all the rules, roles and properties, as well as different screen readers and tools to handle accessible charts. Furthermore, it gives a deeper insight into how charts are turned into accessible charts at all. And then, finally, a short overview of tools, which blind people can work with to get something out of it.

Contents

List of Figures

List of Listings

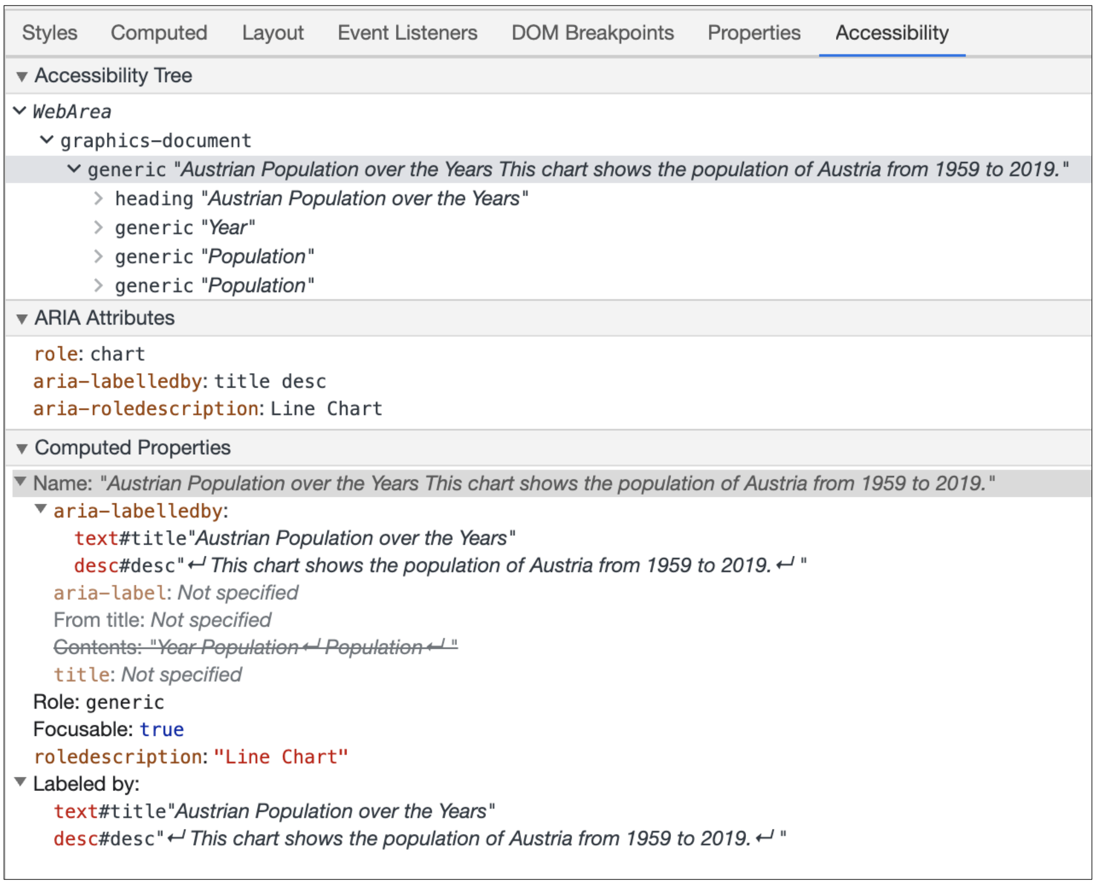
Chapter 1

Web Accessibility

Web accessibility, also sometimes known as “web a11y” (“11” represents the eleven characters between the starting “a” and the ending “y”), describes the conglomerate of efforts to make the web accessible for people with disabilities, like blindness and color-blindness. This includes web content ranging from web user interfaces through infographics to text. However, not only people with disabilities benefit from web a11y. The target group ranges from elderly people, through people using a slow internet connection to people simply using mobile devices. Access to information and communication technologies is defined as a human right by the United Nations and web a11y is even required by law in many situations [W3C 2021a]. To achieve a more accessible web, different standards emerged.

The web browser maintains two parallel internal data structures: the DOM Tree and the Accessibility Tree. In fact, the Accessibility Tree is a subset of the flattened DOM Tree. This subset is used to track user interface objects of the web browser and the objects of the current document. Accessible objects are created in the Accessibility Tree for every DOM element that can be handled by an assistive technology.

The Accessibility Tree can be inspected using a web browser’s development tools. In Google Chrome, for example, it can be viewed by right-clicking on a web page and then selecting Inspect. In this window, where all the inspection areas are open, you can select Accessibility, which can be found in the lower inspection section right next to Properties. The result is shown in Figure 1.1.



Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

▼ Accessibility Tree

```
⌄ WebArea
  ⌄ graphics-document
    ⌄ generic "Austrian Population over the Years This chart shows the population of Austria from 1959 to 2019."
      › heading "Austrian Population over the Years"
      › generic "Year"
      › generic "Population"
      › generic "Population"
```

▼ ARIA Attributes

```
role: chart
aria-labelledby: title desc
aria-roledescription: Line Chart
```

▼ Computed Properties

```
▼ Name: "Austrian Population over the Years This chart shows the population of Austria from 1959 to 2019."
  ▼ aria-labelledby:
      text#title"Austrian Population over the Years"
      desc#desc"↵ This chart shows the population of Austria from 1959 to 2019. ↵ "
    aria-label: Not specified
    From title: Not specified
    ~~Contents: "Year Population ↵ Population ↵ "~~
    title: Not specified
  Role: generic
  Focusable: true
  roledescription: "Line Chart"
▼ Labeled by:
    text#title"Austrian Population over the Years"
    desc#desc"↵ This chart shows the population of Austria from 1959 to 2019. ↵ "
```

Figure 1.1: Google Chrome showing the Accessibility Tree, ARIA Attributes, and Computed Properties. [Screenshot captured by Markus Stradner using Google Chrome.]

Chapter 2

WAI and ARIA

Accessible Rich Internet Applications (ARIA) [W3C 2021c] emerged from the Web Accessibility Initiative (WAI) [W3C 2021b] of the World Wide Web Consortium (W3C) [W3C 2021a]. ARIA consists of a set of rules, roles, and properties, and provides semantics, especially for dynamic content. As some parts of websites are not usable for some people who, for example, rely on screen readers, WAI-ARIA provides the developers with tools to make web content accessible for people with disabilities.

2.1 WAI

The Web Accessibility Initiative is part of the W3C (World Wide Web Consortium). It develops different standards to improve the accessibility of the World Wide Web (WWW) for people with disabilities. Such standards are for example the Web Content Accessibility Guidelines (WCAG) [W3C 2021b].

2.2 ARIA Rules

When working with the ARIA standard, a few rules [Damera 2021] have to be followed:

1. Always use native HTML, unless there is no other way to make elements accessible.
2. Do not change the semantics of native HTML. Use the `<button>` element instad of `<span role="button">`.
3. Make aria-controls keyboard accessible, with the help of `tabindex="0".`
4. Never use `role="presentation"` or `aria-hidden="true"` on focusable elements. It might be confusing if a plain face is focused.
5. Always use accessible names by using the `<label>` element or `aria-labels="Search"`.

All mentioned elements and attributes are part of the standard HTML or SVG markup.

2.3 ARIA Properties

ARIA defines different properties to annotate web content for accessibility. All ARIA properties are denoted by the prefix `aria-`. These properties are used to add further information to an element. There exist many different ARIA properties, in this survey only the most important are described in detail:

- `aria-label`: `aria-label` is used to label an element with a short name or a value. This can be used to give an accessible name to the element. Listing 2.1 shows an example how to use `aria-label` on a button for sending a mail.

```html
<button aria-label="Send" type="submit">Send</button>
```

Listing 2.1: An example of using `aria-label`.

```html
<div id="SendID">Send</div>
<button aria-labelledby="SendID" type="submit">Send</button>
```

Listing 2.2: An example of using `aria-labelledby`.

- `aria-labelledby`: `aria-labelledby` is similar to `aria-label` but gives an alternative way of labelling an element. `aria-labelledby` uses the ID of another element. For example, it can use a text element with a label and refer to it by the ID. Listing 2.2 shows the same example as above but with `aria-labelledby`. It is also possible to combine more labels. This can be helpful to reuse labels. One use case can be seen in Listing 2.3. In the example, the screen reader would read the first button as "Send Mail" and the second button as "Send Direct Message". Notice the "Send" label was reused in both buttons.

- `aria-describedby`: `aria-describedby` property is used similar to `aria-labelledby`. The difference is that with `aria-describedby` one can give the element a longer more detailed description. This is especially important when the `aria-label` or `aria-labelledby` property does not give a detailed enough description of the element. Listing 2.4 shows a use case of `aria-describedby`, where the screen reader would read the more detailed description of the button given by the `aria-describedby` property.

- `aria-valuemin/aria-valuemax`: `aria-valuemin` and `aria-valuemax` properties are used to give a description to range elements, like sliders. This is important to give information of the maximum and the minimum of the slider. `aria-valuenow` can be used as the default value of the slider. Listing 2.5 shows a use case of these properties. The screen reader would give the information of the minimum value of the slider, in this case, 0, the maximum value of the slider, in this case, 100, and the current value of the slider, in this case, 10.

- `aria-roledescription`: `aria-roledescription` can be used to give a natural language description for the role of an element. Sometimes the role of an element is not very meaningful and in this case one can change this with the `aria-roledescription` property. Listing 2.6 shows a use case of `aria-roledescription`. The screen reader would understand that the element is a button and with the `aria-roledescription` it tells the user that it is an attachment button (for example attaching a file to a mail).

- `aria-hidden`: The `aria-hidden` property is used to hide an element and all its children from the Accessibility Tree and therefore hide the elements from the screen reader. This is especially useful for purely decorative elements on a webpage. It can be also used for repeated text or offscreen/collapsed content on the screen like menus. Listing 2.7 shows a use case of the property, where the screen reader would not read the text.

2.4 ARIA Roles

The ARIA properties described until now can be used for both HTML and SVG. The ARIA Graphics Module, which defines all the properties and roles to make SVGs more accessible. The properties defined in the ARIA Graphics Module [W3C 2018] are the same as those discussed in Section 2.3. For example,

```html
<div id="SendID">Send</div>
<div id="MailID">Mail</div>
<button aria-labelledby="SendID MailID"
  type="submit">Send Mail</button>
<div id="MessageID">Direct Message</div>
<button aria-labelledby="SendID MessageID"
  type="submit">Send Direct Message</button>
```

Listing 2.3: An example of combining `aria-labelledby` properties.

```html
<div id="SendID">Send</div>
<div id="MailID">Mail</div>
<div id="SendMailDescriptionID">Sending the text as mail</div>
<button aria-labelledby="SendID MailID" aria-describedby="SendMailDescriptionID"
  type="submit">Send Mail</button>
<div id="MessageID">Message</div>
<div id="SendMessageDescriptionID">Sending the text as a direct message</div>
<button aria-labelledby="SendID MessageID" aria-describedby="
    SendMessageDescriptionID"
  type="submit">Send Direct Message</button>
```

Listing 2.4: An example of using `aria-describedby`.

```html
<div class ="slidecontainer">
  <input type ="range" aria-valuemin="0" aria-valuemax="100" aria-valuenow="10">
</div>
```

Listing 2.5: An example of using `aria-valuemin`, `aria-valuemax`, and `aria-valuenow`.

```html
<div role="button" tabindex="0" aria-roledescription="attachment button">
  attach file
</div>
<button aria-roledescription="attachment button">attach file</button>
```

Listing 2.6: An example of using `aria-roledescription`.

```html
<p aria-hidden="true">
  This text would not be read by the screen reader.
</p>
```

Listing 2.7: An example of using `aria-hidden`.

```
<svg xmlns ="https://www.w3.org/2000/svg" viewBox="0 0 200 100" role="graphics-
    document">
```

Listing 2.8: An example of using `graphics-document`.

```
<g role="graphics-object" aria-label="x-axis">
  <line x1="0" y1="0" x2="900" y2="0" stroke="black" stroke-width="3"/>

  <line x1="114" x2="114" y1="-15" y2="15" stroke="black" stroke-width="2"/>
  <text x="114" y="20" transform="rotate(-45, 114, 20)" font-size="20"
    text-anchor="end">2014</text>
  ...
</g>
```

Listing 2.9: An example of using `graphics-object`.

the `aria-label` property can be added to define the value of a data point in an SVG chart or to define what is seen on the x and the y-axis. In this survey, only the three most important SVG-specific roles are described:

- `graphics-document`: Is used to define that the next part of the document conveys its meaning through visual appearance. In this case it is used to define that the SVG is a document which conveys its meaning through a chart and is accessible. This can be seen in Listing 2.8.
- `graphics-object`: Is a subsection of the `graphics-document` and is used for sub elements in the SVG. It is used inside the SVG `g` grouping element. Every `graphics-object` can contain more `graphics-object`. It can be used to define that the x or y-axis is a graphics object. `graphics-object` can be seen in Listing 2.9. In the example, an x-axis is defined. The role `graphics-object` is set for the x-axis. Inside line elements are defined which are used as the x-axis marks and text elements which are used as the label.
- `graphics-symbol`: Is again a subsection of the `graphics-object` role and can be used to define graphical symbols such as icons. It is part of a larger structure such as a chart. The usage is seen in Listing 2.10. In the example, a symbol is defined which is a simple arrow for the x-axis from the example before. In the x-axis declaration where the arrow is used the `graphics-symbol` role is added.

```
<symbol id="arrow" viewBox="0 0 50 50">
  <polygon points="25,0 50,50 0,50" fill="black" />
</symbol>

...

<g transform="translate(160, 800)" font-family="Verdana">
  <use href="#arrow" role="graphics-symbol" x="910" y="-5" width="10" height="10"
    transform="rotate(90 910,-5)"></use>
  <line x1="0" y1="0" x2="900" y2="0" stroke="black" stroke-width="3"/>

  <line x1="114" x2="114" y1="-15" y2="15" stroke="black" stroke-width="2"/>
  <text x="114" y="20" transform="rotate(-45, 114, 20)" font-size="20"
    text-anchor="end">2014</text>
  ...
</g>
```

Listing 2.10: An example of using `graphics-symbol`.

Chapter 3

Annotated SVG Charts

As discussed in Sections 2.3 and 2.4, SVGs can be annotated with ARIA roles and properties. In this section, a deeper look into the annotation of SVGs and how to accomplish such annotated SVGs is given. There are three different ways to annotate SVGs:

- *Manual annotation*: Simply annotate SVGs by hand. An already existing SVG file can be used and then all the necessary roles and properties get added. Or the SVG itself can be created from scratch and all the necessary roles and properties get added. This approach can be quite tedious when more complex SVG charts get created.

- *Semi-automatic annotation*: Already existing programs like AChart can be used. This is categorized as semi-automatic annotation, since it adds all the necessary annotations, but there is no deeper control of how the final chart will look without modifying the predefined "recipes" for each chart type. In AChart there are recipes for a pie chart, a bar chart, and a line chart (more information about AChart in Section 5.1.2). If a different chart is needed, the manual annotation process has to be used, or a new recipe has to be written in TypeScript.

- *Automatic annotation*: The final way to create annotated SVGs is to use a vector graphics editor which can add annotations. For example, Glimpse is such an editor (see Section 5.1.1).

Overall, it can be differentiated between *simply* annotated SVG charts and *richly* annotated SVG charts. Simply annotated SVG charts contain all the roles and properties seen until now and defined in ARIA. However, there are also proposals for properties and custom classes for annotating SVG charts which create more detailed annotations. These SVG charts are called richly annotated SVG charts. More about richly annotated SVG charts is given in Section 3.2.

3.1 Simply Annotated SVG Charts

First, a deeper look into simply annotated SVGs is given. A simply annotated SVG chart makes use of the standard WAI ARIA roles and properties. Figure 3.1 shows the structure of a simply annotated SVG chart. The dark blue parts contain the visible parts of the chart, like the heading, the bars, and the axes. The purple circles with the numbers are the tab indices. They define the order in which elements are read by the screen reader. In this case, first, the title is read out, then the more detailed description, then the x-axis label, and finally the values of the bars. Th red and grey parts of the chart can be focused by the screen reader. However, the grey parts are not visible on the chart, they are only provided for the screen reader to read out.

In Figure 3.2 such a simply annotated SVG is given. For non-blind people, the SVG looks like any other. Looking into the source code of the SVG, it is indeed a simply annotated SVG, as can be seen in Listing 3.1. It contains standard ARIA roles and properties, like `aria-label` and `aria-labelledby`.

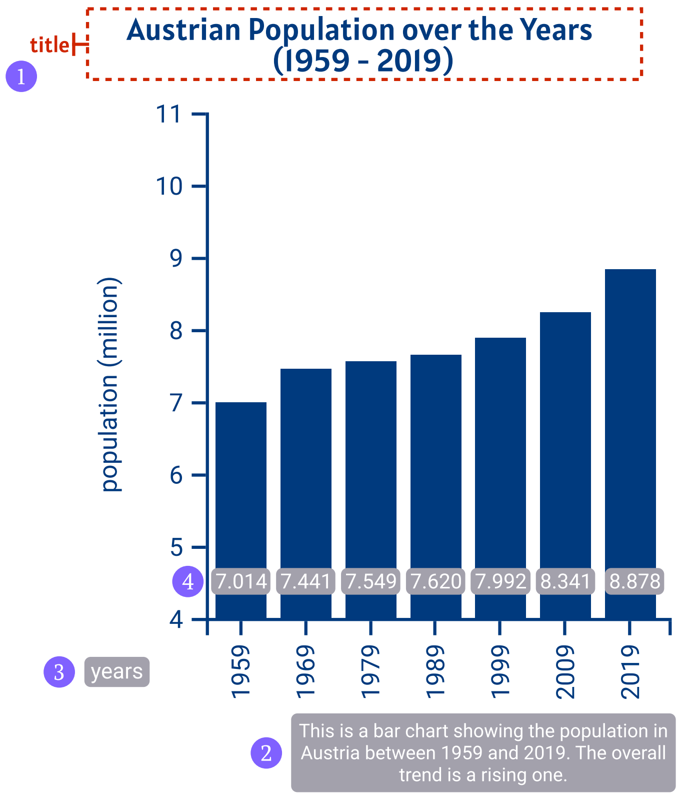


Figure 3.1: The structure of a simply annotated SVG chart. [Drawn by Markus Stradner, inspired by the illustrations of a barchart from Fizz Studio (`https://fizz.studio/wp-content/uploads/2018/04/barchart.png`).]

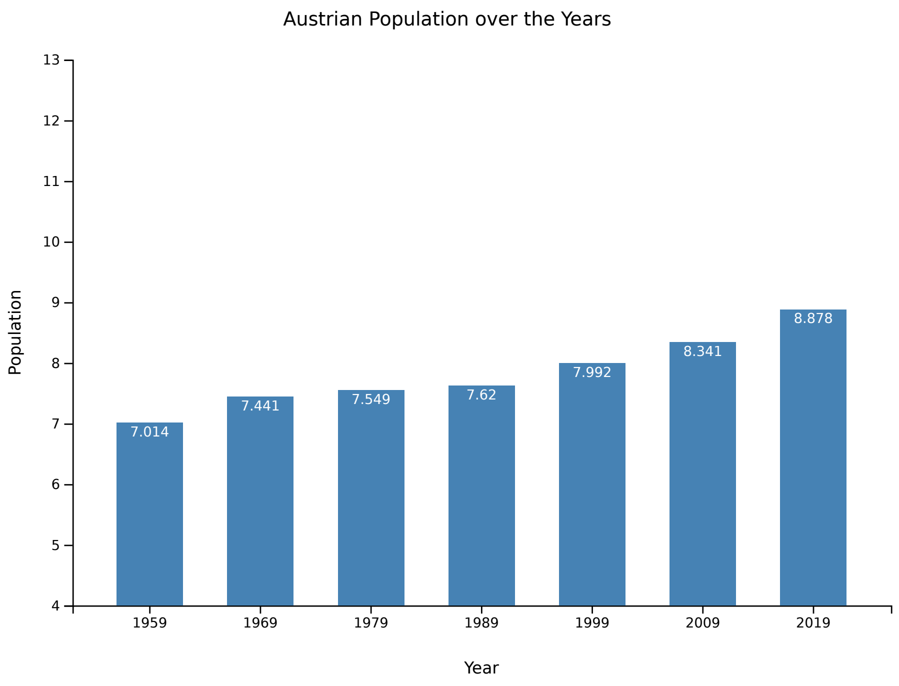


Figure 3.2: An example of a simply annotated SVG chart. It contains a bar chart showing the Austrian population from 1959 to 2019. On the y axis is the population in millions and on the x axis are the years. [Created by Alexander Perko with AChart.]

```
<svg viewBox="0 0 750 600" version="1.1" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink" role="graphics-document">
  <style type="text/css">
    .bar {fill: steelblue; }
  </style>
  <rect id="backdrop" width="750" height="600" fill="#fff"></rect>
  <g id="ChartRoot" tabindex="0" transform="translate(100,100)"
    aria-labelledby="title desc" aria-charttype="bar" aria-roledescription="Bar
        Chart">
    <desc id="desc">
      A bar chart showing the population in Austria between 1959 and 2019.
      The overall trend is rising.
    </desc>
    <rect width="600" height="400" fill="none"></rect>
    <text id="title" text-anchor="middle" font-size="14" x="275" y="-25">
      Austrian Population over the Years
    </text>
    <g id="xScale" aria-axistype="category" aria-roledescription="x-Axis"
      aria-labelledby="x-title" tabindex="0" transform="translate(0,400)"
      fill="none" font-size="10" font-family="sans-serif" text-anchor="middle">
      <text y="50" x="300" text-anchor="middle" fill="black"
        font-size="12" id="x-title">Years</text>
      <path class="domain" stroke="currentColor" d="M0.5,6V0.5H600.5V6"></path>
      <g class="tick" opacity="1" transform="translate(56.757,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" id="x1">1959</text>
      </g>
      ...
    </g>
    <g id="yScale" aria-roledescription="y-Axis" tabindex="0" aria-valuemin="4"
       aria-valuemax="13" aria-labelledby="y-title" fill="none" font-size="10"
       font-family="sans-serif" text-anchor="end">
      <text transform="rotate(-90)" y="-38" x="-200" text-anchor="middle"
        fill="black" id="y-title" font-size="12">Population</text>
      <path class="domain" stroke="currentColor" d="M-6,400.5H0.5V0.5H-6"></path>
      <g class="tick" opacity="1" transform="translate(0,400.5)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" id="y1">4</text>
      </g>
      ...
    </g>
    <g id="dataarea" tabindex="0">
      <title>
        Population
      </title>
      <g tabindex="0" transform="translate(32.432,266.044)"  aria-labelledby="x1">
        <rect class="bar" width="48.649" height="133.956"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10"
          fill="white" id="value1">7.014</text>
      </g>
      ...
    </g>
  </g>
</svg>
```

Listing 3.1: The source code of the simply annotated SVG chart. It contains standard ARIA roles and properties, like `aria-label` and `aria-labelledby`.

3.2 Richly Annotated SVG Charts

Richly annotated SVG charts contain more detailed descriptions and information about the chart. Numerous tools and libraries help with creating richly annotated SVG charts. However, each tool and library takes its own approach and standards still have to emerge:

- Describler [Schepers 2021a]: Describler used standard ARIA roles and properties for SVG charts, as well as introducing a custom taxonomy for additional rich annotations.
- AChart [Andrews and Kopel 2021a; Andrews and Kopel 2021b]: AChart builds upon Describler's roles and propeties for rich annotation of SVG charts.
- W3C (Bellamy-Royds) [Bellamy-Royds 2021]: Amelia Bellany-Royds proposed standard ARIA roles and properties for rich annotations. Until now, they have not been added to the standard ARIA roles and properties.
- Highcharts [HC 2020]: Highcharts also adds standard ARIA roles and properties and defines a custom taxonomy for rich annotations.
- Semiotic [Meeks 2021]: Semiotic handles annotations the same as Highcharts by adding the standard ARIA roles and properties and defining a custom taxonomy for rich annotations.
- amCharts [AMC 2021]: amCharts uses standard ARIA roles and properties as well as menu elements for rich annotations. Due to this, it is not as versatile and feature-rich as the other libraries.
- FusionCharts [FC 2021]: FusionCharts uses standard ARIA roles and properties and defines a custom taxonomy for rich annotations.

All of these tools and libraries define different ways of richly annotating SVG charts. The most promising concepts from the different proposals and taxonomies are summarised in Table 3.1. In particular:

- Data Point: A declaration of a data point. This is probably the most promising concept. It is used to define a data point in a chart as a data point. From these data point values, the screen reader or the chart interpreter can then calculate additional information like the mean of the data or a trend line.
- Collection of Data Points: Used to add different data points to a collection of data points. This is very useful if a chart shows more than one group of data points, since the screen reader can then differentiate between them and therefore calculate the additional information for each group of data separately.
- Legend: Used to define a legend for the chart. The goal is that the information is received from the data itself without creating all the labels by hand. This is very useful when the data changes.
- Legend Item: Part of the legend and used to define which items are in the legend.
- Axis: Used to define an axis. Again, without the need to define all the labels for the screen reader.
- Axis Label: Part of an axis and used to read out the label of the axis.

A richly annotated SVG chart contains all the standard roles and properties and additional non-standard properties, which are used to encode more of the semantics of the chart inside the SVG code. These additional properties are defined differently within the different tools and libraries (see Table 3.1). Figure 3.3 shows how such a richly annotated SVG is structured. The dark blue parts indicate the visible parts of the chart, like the heading, the bars, and the axes. The purple circles with numbers are the tab indices. They define the order in which elements are read by the screen reader. In this case, first, the title is read out, then the more detailed description, then the x-axis, the y-axis, and finally the values of the bars. The red and grey parts can be focused by the screen reader. However, the grey parts are not actually

Meaning	**Describler / AChart**	**W3C (Bellamy-Royds)**	**Highcharts**	**Semiotic**	**amCharts**	**FusionCharts**
Data Point	`datapoint`	`graphics-dataunit` / `aria-datavalues`	`highcharts-point`		`menuitem`	
Collection of Data Points	`dataset`	`graphics-dataline` / `aria-datavaluearray`	`highcharts-line-series`	`lines` / `pieces`	`menu`	`raphael-group-N-plot-group`
Legend	`legend`	`graphics-legend`	`highcharts-legend`			`raphael-group-N-legend`
Legend Item	`legenditem`		`highcharts-legend-item`			
Axis	`xaxis` / `yaxis`	`graphics-axis`	`highcharts-axis` / `highcharts-xaxis` / `highcharts-yaxis`	`axis`		`raphael-group-N-dataset-axis-name`
Axis Label	`axislabel`		`highcharts-axis-labels`	`axis-label`		`raphael-group-N-dataset-axis`

Table 3.1: Proposals for rich annotations in various systems. N denotes a variable number.

visible in the chart, they are only provided for the screen reader to read out. The orange parts indicate the rich annotations for the x and y-axis. The green part indicates supplementary information which is created by the screen reader from the encoded data points.

A richly annotated SVG chart does not look different for sighted people. The richly annotated SVG looks the same as the simply annotated SVG when displayed as a graphic, just like in Figure 3.2. Looking deeper into the SVG source code, shown in Listing 3.2, reveals the difference. It contains standard ARIA roles and properties, like `aria-label` and `aria-labelledby`, but also rich annotations like `datapoint`, `dataset` and `xaxis` (as proposed by Describler and AChart).

```
<svg viewBox="0 0 750 600" version="1.1" xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink" role="graphics-document">
  <style type="text/css">
    .bar {fill: steelblue; }
  </style>
  <rect id="backdrop" width="750" height="600" fill="#fff"></rect>
  <g id="ChartRoot" role="chart" tabindex="0" transform="translate(100,100)"
    aria-labelledby="title desc" aria-charttype="bar"
    aria-roledescription="Bar Chart">
    <desc id="desc">This is a bar chart showing the population in Austria
      between 1959 and 2019. The overall trend is a rising one.</desc>
    <rect role="chartarea" width="600" height="400" fill="none"></rect>
    <text id="title" role="heading" text-anchor="middle"
      font-size="14" x="275" y="-25">Austrian Population over the Years</text>
    <g id="xScale" role="xaxis" aria-axistype="category"
      aria-roledescription="x-Axis" aria-labelledby="x-title" tabindex="0"
      transform="translate(0,400)" fill="none"
      font-size="10" font-family="sans-serif" text-anchor="middle">
      <text y="50" x="300" text-anchor="middle" fill="black"
        font-size="12" role="heading" id="x-title">Years</text>
      <path class="domain" stroke="currentColor" d="M0.5,6V0.5H600.5V6"></path>
      <g class="tick" opacity="1" transform="translate(56.757,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em"
          role="axislabel" id="x1">1959</text>
      ...
    </g>
    <g id="yScale" role="yaxis" aria-roledescription="y-Axis" tabindex="0"
      aria-valuemin="4" aria-valuemax="13" aria-labelledby="y-title" fill="none"
      font-size="10" font-family="sans-serif" text-anchor="end">
      <text transform="rotate(-90)" y="-38" x="-200" text-anchor="middle"
        fill="black" role="heading" id="y-title" font-size="12">Population</text>
      <path class="domain" stroke="currentColor" d="M-6,400.5H0.5V0.5H-6"></path>
      <g class="tick" opacity="1" transform="translate(0,400.5)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em"
          role="axislabel" id="y1">4</text>
      </g>
      ...
    </g>
    <g id="dataarea" role="dataset" tabindex="0">
      <title>Population</title>
      <g tabindex="0" transform="translate(32.432,266.044)"
        role="datapoint" aria-labelledby="x1">
        <rect class="bar" width="48.649" height="133.956"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10"
          fill="white" role="datavalue" id="value1">7.014</text>
      </g>
      ...
    </g>
  </g>
</svg>
```

Listing 3.2: The source code of the richly annotated SVG chart. It contains standard ARIA roles and properties, like `aria-label`, `aria-labelledby` etc., and also some custom roles such as `datapoint`, `dataset` and `xaxis`. The rich annotations are used as proposed by Describler and AChart.

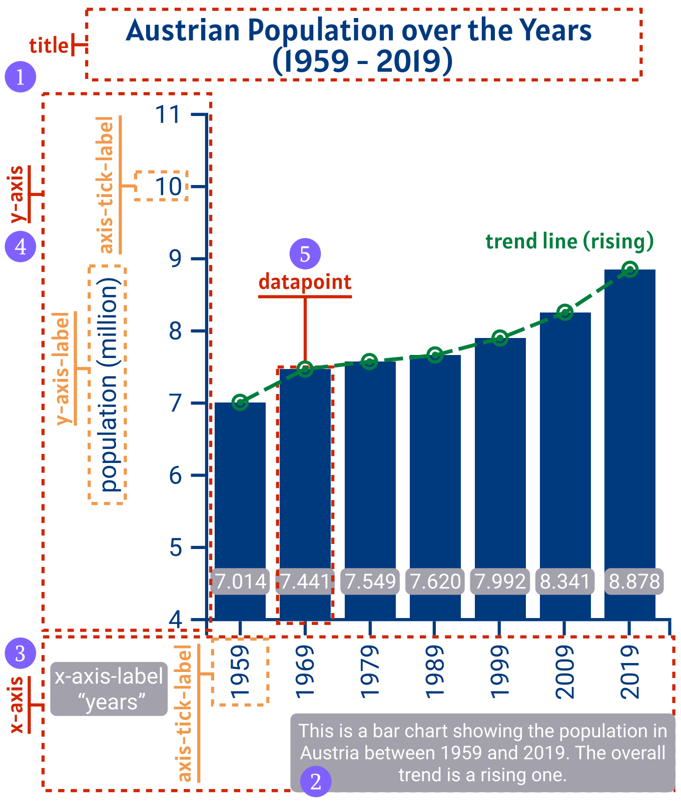


Figure 3.3: The elements of a richly annotated SVG chart encode more of its semantics. [Drawn by Markus Stradner, inspired by the illustration of a bar chart from Fizz Studio (`https://fizz.studio/wp-content/uploads/2018/04/barchart.png`).]

Chapter 4

Screen Readers

Screen readers are very important for blind people because the audio description from the computer screen, more detailed, menu elements, graphics and web-content, is the primary way, next to braille displays and keyboards, how to understand the interface and interact with the computer.

Available screen readers include NVDA (Windows), JAWS (Windows), VoiceOver (macOS), and Narrator (Windows). Figure 4.1 shows that NVDA is currently most used as a primary screen reader, followed by JAWS and VoiceOver.

4.1 NVDA

NVDA (Non-Visual Desktop Access) [NVA 2021] is an open-source and free-to-use screen reader for the Windows operating system. It supports more than 50 languages and is easy and intuitive to use. NVDA has also the ability to run from portable media, like a USB stick, without even installing it.

4.2 JAWS

JAWS stands for Job Access With Speech [FS 2021] and is a commercial screen reader with support for more than 30 languages on the Windows operating system. Text on an image file or in inaccessible PDF documents is also no problem for JAWS, due to its convenient OCR (Optical Character Recognition) feature.

4.3 VoiceOver

VoiceOver [Apple 2021] is a screen reader which comes preinstalled with macOS X, which makes it easy to set up as there is no third-party software required. It reads out elements of the User Interface and accessibility information and supports braille displays. VoiceOver starts before the sign-in and is also available on mobile devices (iOS).

In Figure 4.2, the window on the left shows the VoiceOver audio output for the accompanying SVG on the right. A glimpse of how VoiceOver works can be found at `https://youtu.be/EZtlH_2s60w`.

4.4 Narrator

Narrator [Assistiv 2021] is a screen reader which comes preinstalled with Windows 10, so there is no extra installation necessary. Since this app is developed by Microsoft, it supports all other Microsoft apps, like Edge, the Office suite, but also Google Chrome. However, Narrator does not support plugins or scripting.

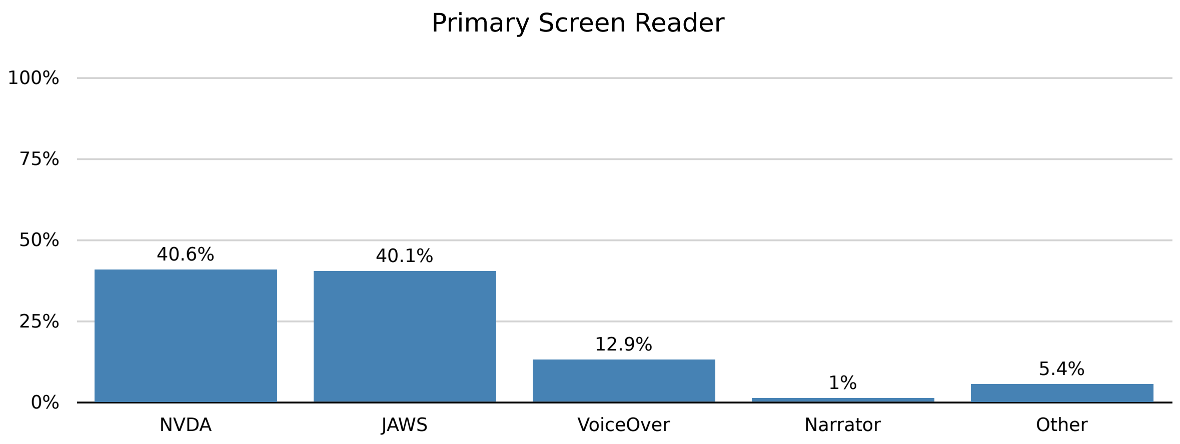
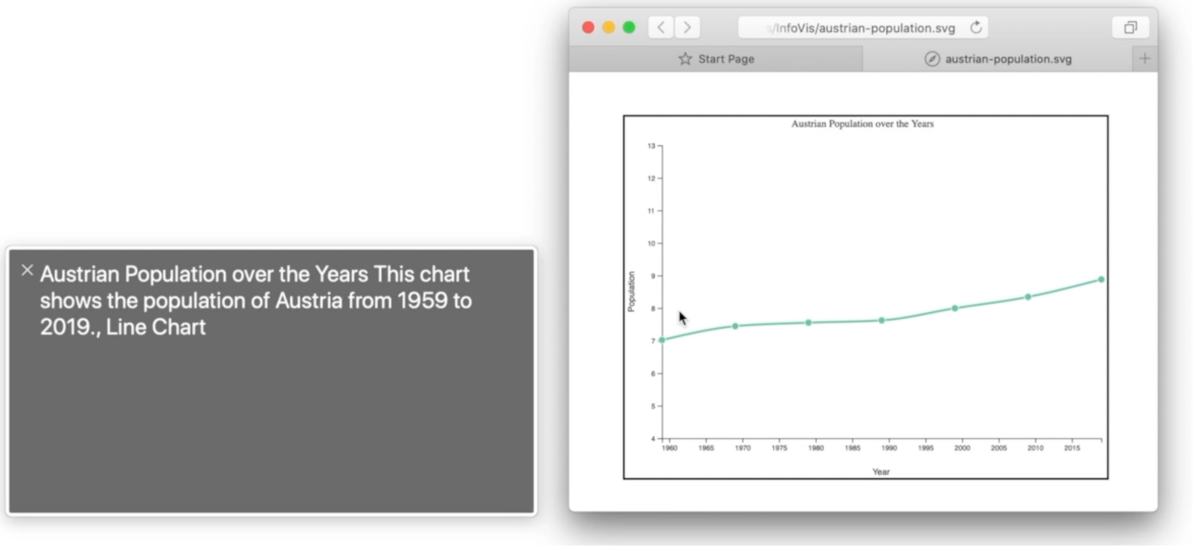


Figure 4.1: Primary screen readers used by respondents to WebAIM's Screen Reader User Survey #8. [Created by Lisa Habich, from the results of WebAIM's Screen Reader Survey (`https://webaim.org/projects/screenreadersurvey8/`).]


Austrian Population over the Years This chart shows the population of Austria from 1959 to 2019., Line Chart

/InfoVis/austrian-population.svg
Start Page
austrian-population.svg
Austrian Population over the Years
Population
Year

Figure 4.2: On the left hand side, the audio description given by VoiceOver to the accompanying SVG on the right hand side. [Screenshot taken by Lisa Habich.]

Chapter 5

Tools

Tools for dealing with accessible charts can be divided into three groups: chart generators, chart readers, and other tools. These will be looked at in turn.

5.1 Chart Generators

Chart generators allow users to create annotated SVG charts from an input dataset.

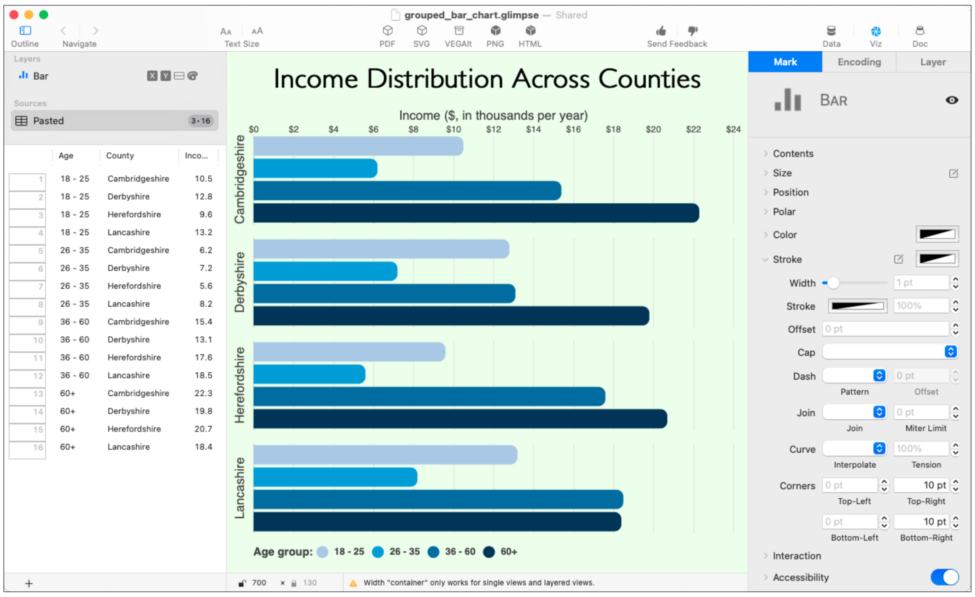
5.1.1 Glimpse

Glimpse is an editor for generating infographics, developed by Michał Kasprzyk et al. [Kasprzyk et al. 2021]. It is a native app for macOS and developed upon Vega-Lite. Currently (as of May 2021), it is in a closed beta stage. The developers state on their website, that they are planning to release it later in 2021. Glimpse seems especially promising with regards to its flexibility, as it is based on visual building blocks which can be manipulated directly. The customization process seems to be comparable to chart manipulation as seen in Microsoft Excel or Apple Numbers. Concerning accessibility, the software is promising, as charts can be exported to ARIA-annotated SVG. The annotation mostly consists of ARIA labels comprising contextual information as well as the values of data points within the chart. Once released, Glimpse could become a very useful tool for creating SVG charts and especially valuable because of its automatically generated ARIA-annotation. Figure 5.1 shows Glimpse being used to create a grouped bar chart. An excerpt of the output markup generated by Glimpse be seen in Figure 5.2.

5.1.2 AChart Creator

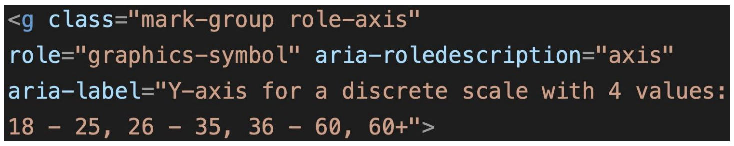
AChart Creator is a command-line tool for creating annotated SVG graphics from a CSV source. It is developed by Keith Andrews and Christopher Kopel [Andrews and Kopel 2021a], is freely available and open source. Tabular data stored in CSV format can be opened and transformed into simple SVG charts. There are three "recipes" available to choose from: bar chart, line chart, and pie chart. Optionally, a title, a description, and labels for the x-axis and y-axis can be added. Everything from source-file and output-file, over chart type to additional information has to be added as a parameter when invoking the program. SVG files created by the program contain rich information like, for instance, data points and their corresponding values. The program is easy to use, given basic knowledge in handling the command-line. AChart Creator is limited to the three pre-defined chart-types and cannot annotate already existing SVG charts. Making new chart types or modifying the existing chart types requires programming in TypeScript.

An exemplary use of AChart Creator to create a line chart can be seen in Figure 5.3, alongside the input CSV file. The generated output chart can be seen in Figure 5.4. An excerpt of its SVG source code showing a data point group can be seen in Figure 5.5. To see AChart Creator in action, view the showcase video on YouTube (`https://youtu.be/tc9Z5zbMBLA`).



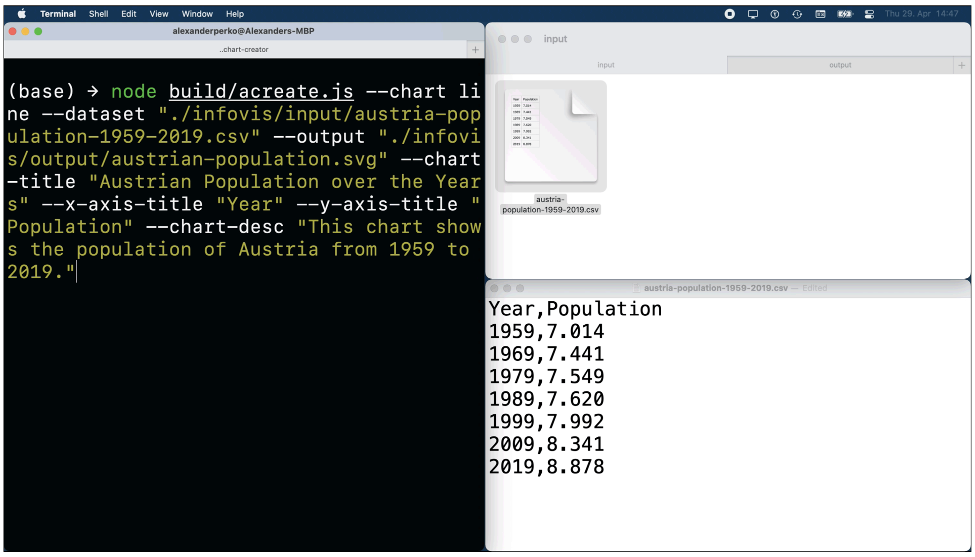
	Age	County	Inco...
1	18 - 25	Cambridgeshire	10.5
2	18 - 25	Derbyshire	12.8
3	18 - 25	Herefordshire	9.6
4	18 - 25	Lancashire	13.2
5	26 - 35	Cambridgeshire	6.2
6	26 - 35	Derbyshire	7.2
7	26 - 35	Herefordshire	5.6
8	26 - 35	Lancashire	8.2
9	36 - 60	Cambridgeshire	15.4
10	36 - 60	Derbyshire	13.1
11	36 - 60	Herefordshire	17.6
12	36 - 60	Lancashire	18.5
13	60+	Cambridgeshire	22.3
14	60+	Derbyshire	19.8
15	60+	Herefordshire	20.7
16	60+	Lancashire	18.4

Figure 5.1: Glimpse: Creating a grouped bar chart. [Screenshot taken from `glipmse.io`.]



```
<g class="mark-group role-axis"
role="graphics-symbol" aria-roledescription="axis"
aria-label="Y-axis for a discrete scale with 4 values:
18 – 25, 26 – 35, 36 – 60, 60+">
```

Figure 5.2: Glimpse: Part of the richly annotated SVG code for a grouped bar chart. [Screenshot captured by Alexander Perko viewing a Glimpse output file published on `glimpse.io`.]


Terminal
Shell
Edit
View
Window
Help
alexanderperko@Alexanders-MBP
..chart-creator
input
input
output
austria-population-1959-2019.csv
austria-population-1959-2019.csv — Edited

```
(base) → node build/acreate.js --chart li
ne --dataset "./infovis/input/austria-pop
ulation-1959-2019.csv" --output "./infovi
s/output/austrian-population.svg" --chart
-title "Austrian Population over the Year
s" --x-axis-title "Year" --y-axis-title "
Population" --chart-desc "This chart show
s the population of Austria from 1959 to
2019."
```

```
Year,Population
1959,7.014
1969,7.441
1979,7.549
1989,7.620
1999,7.992
2009,8.341
2019,8.878
```

Figure 5.3: AChart Creator: A command line to create a line and the corresponding input CSV file. [Screenshot captured by Alexander Perko.]

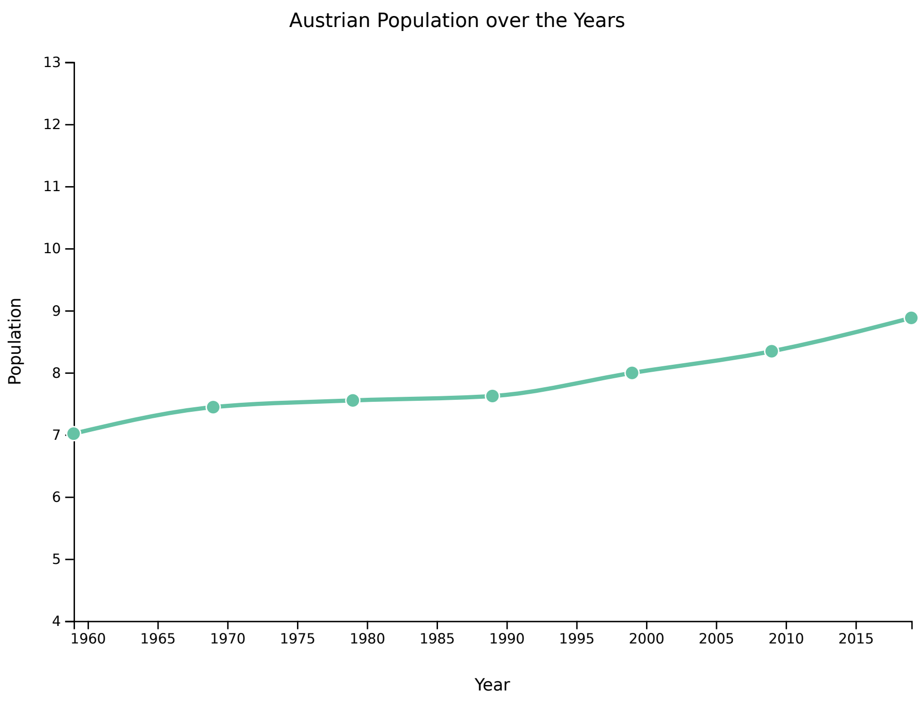
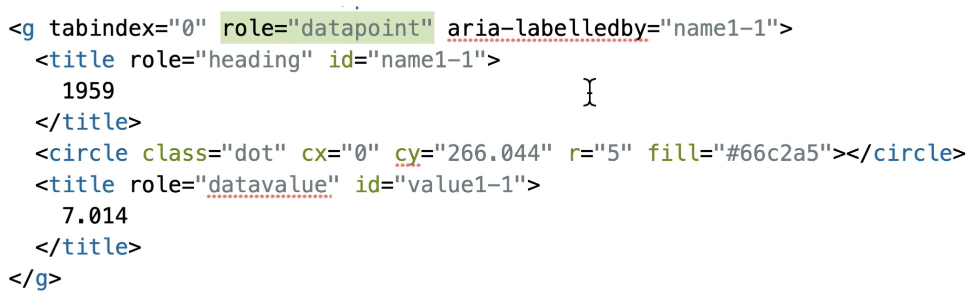


Figure 5.4: AChart Creator: The resulting richly annotated line chart. [Created as SVG by Alexander Perko with AChart Creator.]



```
<g tabindex="0" role="datapoint" aria-labelledby="name1-1">
  <title role="heading" id="name1-1">
    1959
  </title>
  <circle class="dot" cx="0" cy="266.044" r="5" fill="#66c2a5"></circle>
  <title role="datavalue" id="value1-1">
    7.014
  </title>
</g>
```

Figure 5.5: AChart Creator: Part of the richly annotated SVG code for a line chart. See A.2 for the full listing of the SVG output code. [Screenshot captured by Alexander Perko.]

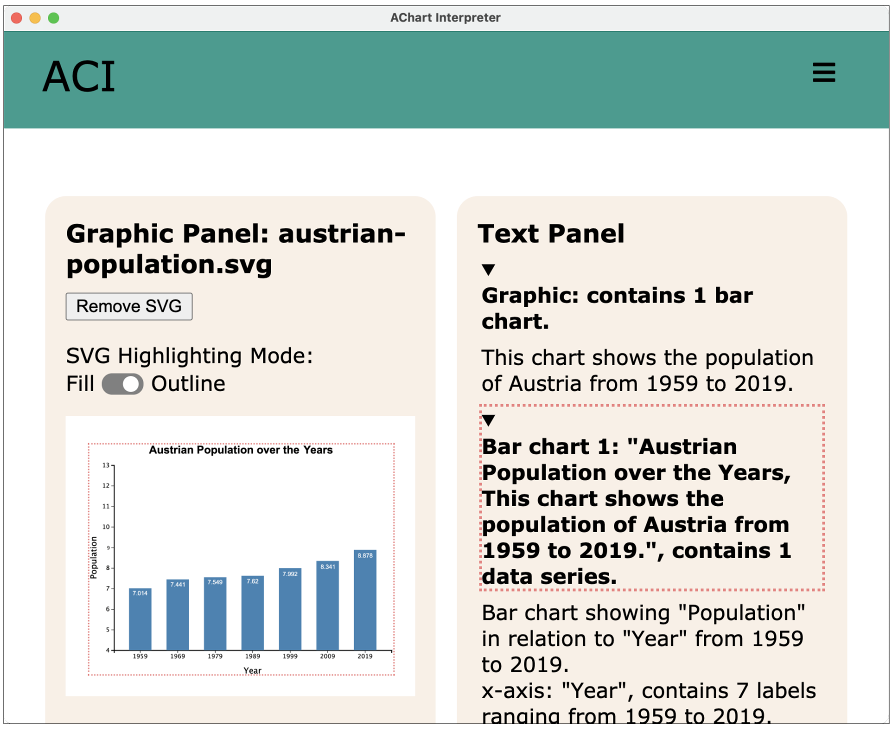
5.2 Chart Readers

Chart readers open and interpret annotated SVG charts in some way, for example by reading them out, or by summarising them.

5.2.1 AChart Interpreter

AChart Interpreter is a web application developed by Keith Andrews and Christopher Kopel [Andrews and Kopel 2021b] for reading out and navigating through annotated SVG files. It was inspired by Doug Schepers' Describler (see Section 5.2.2). The software is freely available and is open source. It can be downloaded from GitHub to run it locally either as a standalone Electron App or as a server on localhost. There is also a hosted version available on github.io (`https://tugraz-isds.github.io/achart-interpreter/`). As there are some difficulties when opening files from disk, Google Chrome (or Mozilla Firefox) is (are) recommended to access AChart Interpreter.

AChart Interpreter's main screen consists of two panels: a Graphic Panel on the left-hand side and a Text



Graphic Panel: austrian-population.svg

Remove SVG

SVG Highlighting Mode:
Fill Outline

Text Panel

▼
Graphic: contains 1 bar chart.

This chart shows the population of Austria from 1959 to 2019.

▼
Bar chart 1: "Austrian Population over the Years, This chart shows the population of Austria from 1959 to 2019.", contains 1 data series.

Bar chart showing "Population" in relation to "Year" from 1959 to 2019.
x-axis: "Year", contains 7 labels ranging from 1959 to 2019.

Figure 5.6: AChart Interpreter: Navigating through a richly annotated SVG chart. See A.3 for the full listing of the input file. [Screenshot captured by Alexander Perko.]

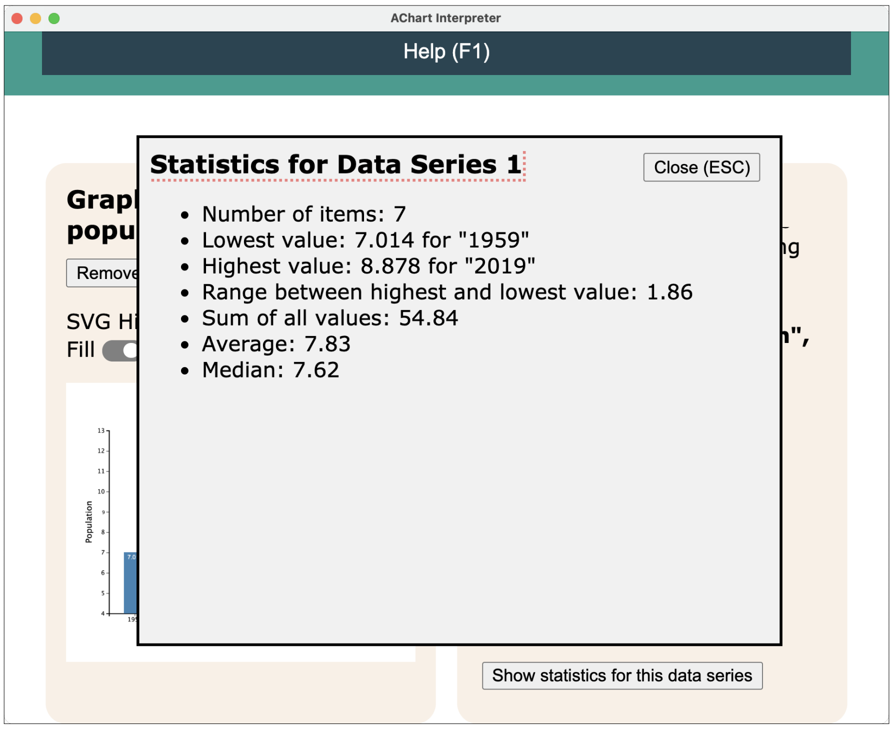
Panel on the right-hand side. The software is built from the ground up to be accessible, which is why it is fully navigable through the keyboard and reads out aloud every text element (including menu items) on screen. The audio description feature can be turned off completely or used in tandem with a screen reader, which-for there are different modes to optimize AChart Interpreter's keyboard shortcuts. Richly annotated SVG charts can be loaded from disk through the file chooser in the main menu. Moreover, there are multiple sample SVG files to choose from. The main menu may be collapsed behind a menu button depending on the screen size. Once opened, an annotated SVG chart is displayed in the synchronised split screen, as can be seen in Figure 5.6. As the user navigates through the elements of a chart, the accompanying annotations are read out aloud.

When the chart at hand was richly annotated with AChart Creator, for example, AChart Interpreter can access special properties like data points and their values and hence navigate through data series or even sort data points in a different order based on their value. In addition, it can jump between data points of different data series to directly compare them, if there are multiple data series in the chart. Another useful feature is the Statistics Window, which shows additional information about the data, such as the highest and lowest values, as well as the mean or the range, as can be seen in Figure 5.7. This aims to give the user a better understanding of the data and simulates the "first glance" at the chart experienced by sighted people. For better illustration of the usage of AChart Interpreter consider viewing the showcase video on YouTube (`https://youtu.be/NLKqTTnKLII`).

5.2.2 Describler

Describler [Schepers 2021a] is an experimental prototype screen reader for SVG by Doug Schepers and was the first of its kind. It is a tool where you can upload your ARIA-SVG and then the chart is introduced by it and you can navigate through the whole chart by pressing the tab or the right-arrow button to get all the accessible information about the chart, as can be seen in Figure 5.8.

If you visit describler.com, at first, you see a detailed introduction what Describler is and what it is supposed to do. Moreover, a description of what an SVG is and further information about how to use



Statistics for Data Series 1

- Number of items: 7
- Lowest value: 7.014 for "1959"
- Highest value: 8.878 for "2019"
- Range between highest and lowest value: 1.86
- Sum of all values: 54.84
- Average: 7.83
- Median: 7.62

Figure 5.7: AChart Interpreter: Statistics window. [Screenshot captured by Alexander Perko.]

it. Describler also offers to select between eight different example files, just to show how it works. Unfortunately, to load the image successfully, you have to load it twice. But when it is loaded you can navigate through it.

Since development for the tool has been stopped and it is still in an experimental state, other issues also arise. For example, once you tab through the chart and would like to compare one data point to another, it gets stuck. Also, the pronunciation of what has been read out depends on the operating system and not on the chart itself.

To mention positive things, it is easy to use the web app (written in plain JavaScript), on all common operating systems in every common browser. Furthermore, it is also possible to run it on your localhost, because the source code is open-source and can be found on Doug Schepers repository on GitHub [Schepers 2021b].

In summary, it would be a really nice tool to handle accessible charts, because it is easy to use and basically accessible to everyone, but the main problem is its very buggy behavior.

5.3 Other Tools

In addition to the classic screen readers and the tools mentioned above, which deal specifically with accessible charts, some other tools are also worth mentioning.

Web Accessibility [van der Schee 2021] and Bri1liant [Barth and McHugh 2021] are Visual Studio Code language extensions, supporting web developers to make their websites more accessible. Both suggest improvements to the HTML, CSS, and JavaScript code to improve the accessibility of websites. An example can be seen in Figure 5.9.

Finally, current web browsers also provide an accessibility audit as part of their development tools, which work like executing unit tests and serve as a short accessibility check of a website. Figure 5.10 and

Figure 5.11.

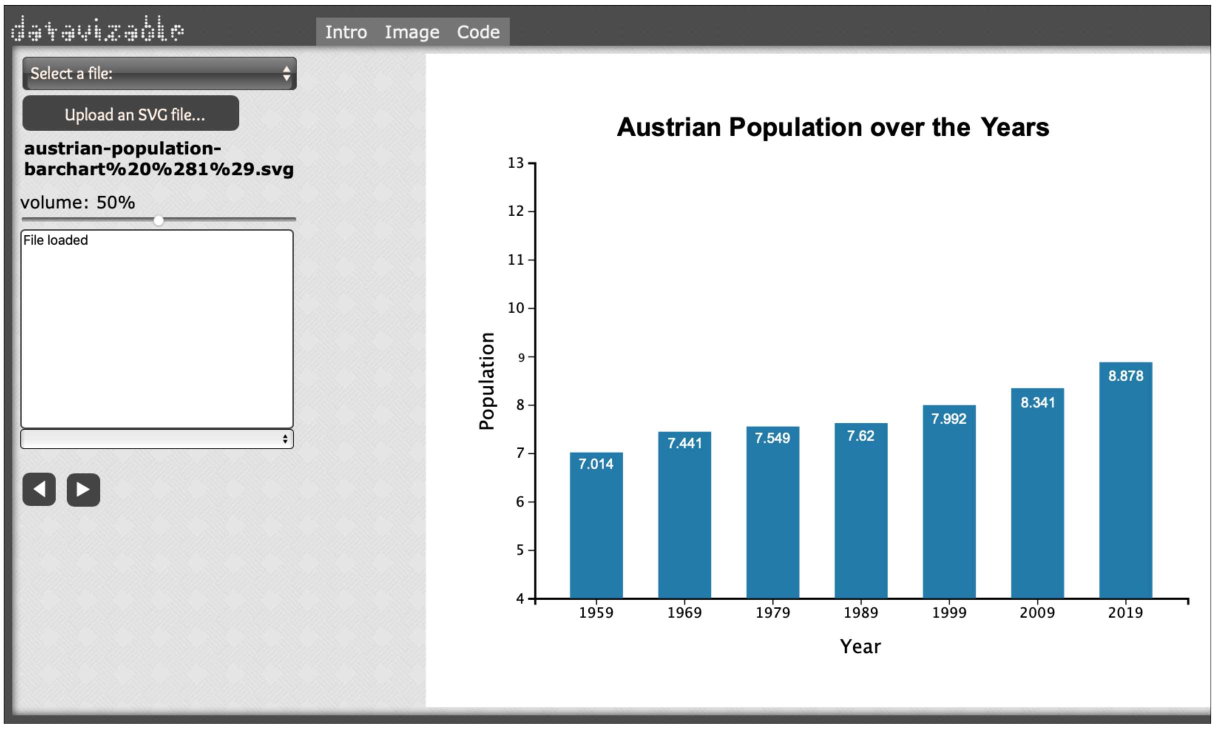
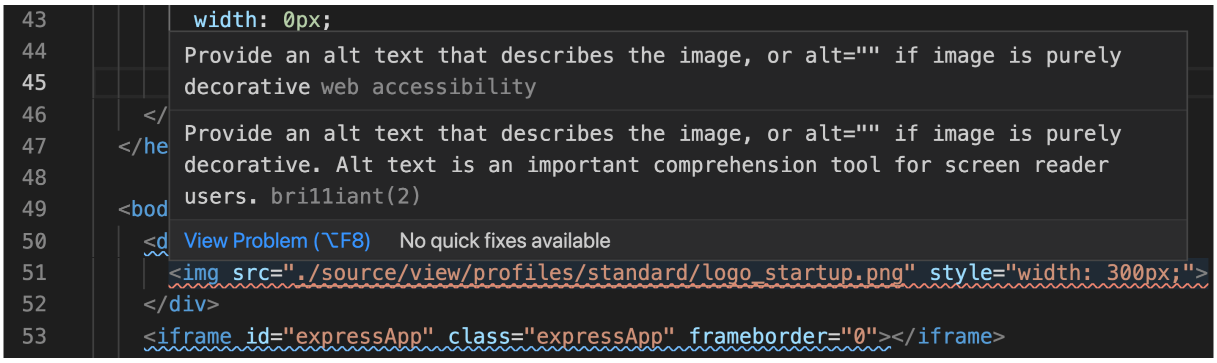
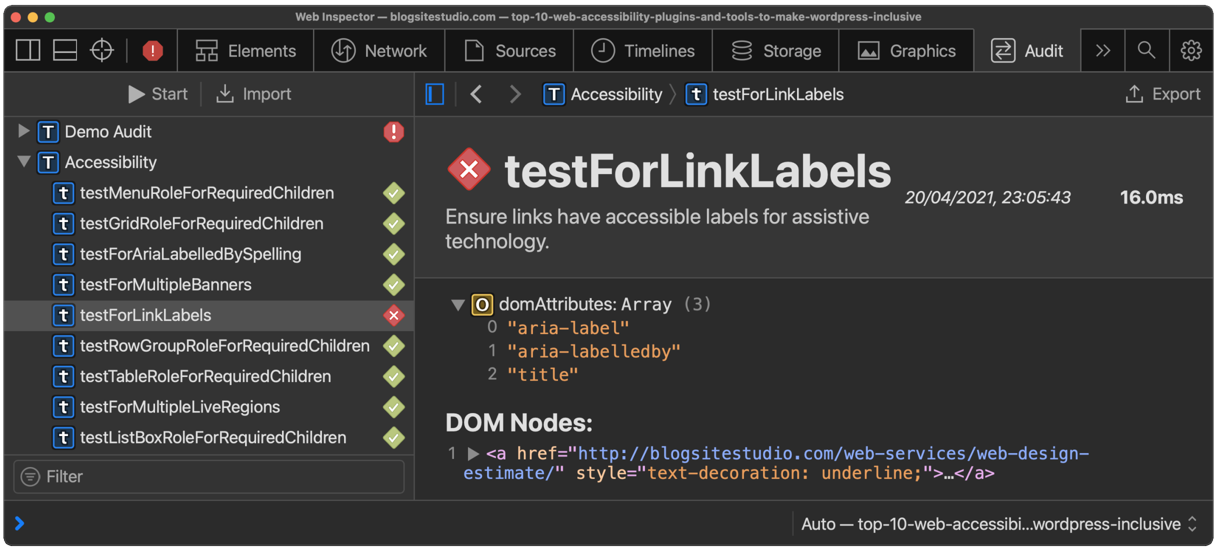


Figure 5.8: Describler: An uploaded annotated SVG chart. See A.3 for the full listing of the input file. [Screenshot was captured by Markus Stradner using Describler.]



```
        width: 0px;
Provide an alt text that describes the image, or alt="" if image is purely
decorative web accessibility
    </
  </he
Provide an alt text that describes the image, or alt="" if image is purely
decorative. Alt text is an important comprehension tool for screen reader
users. bri11iant(2)
  <bod
    <d  View Problem (⌥F8)   No quick fixes available
      <img src="./source/view/profiles/standard/logo_startup.png" style="width: 300px;">
    </div>
    <iframe id="expressApp" class="expressApp" frameborder="0"></iframe>
```

Figure 5.9: Microsoft VS Code with multiple plugins. [Screenshot captured by Alexander Perko.]


Web Inspector — blogsitestudio.com — top-10-web-accessibility-plugins-and-tools-to-make-wordpress-inclusive
Elements
Network
Sources
Timelines
Storage
Graphics
Audit
Start
Import
Accessibility
testForLinkLabels
Export
Filter
Auto — top-10-web-accessibi...wordpress-inclusive

- Demo Audit
- Accessibility
  - testMenuRoleForRequiredChildren
  - testGridRoleForRequiredChildren
  - testForAriaLabelledBySpelling
  - testForMultipleBanners
  - testForLinkLabels
  - testRowGroupRoleForRequiredChildren
  - testTableRoleForRequiredChildren
  - testForMultipleLiveRegions
  - testListBoxRoleForRequiredChildren

testForLinkLabels 20/04/2021, 23:05:43 **16.0ms**

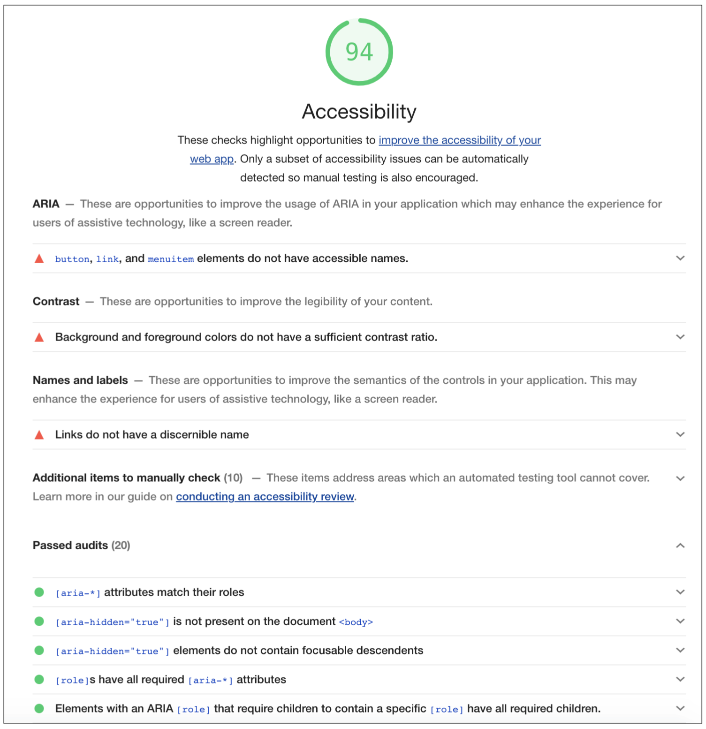
Ensure links have accessible labels for assistive technology.

```
▼ domAttributes: Array (3)
   0 "aria-label"
   1 "aria-labelledby"
   2 "title"
```

DOM Nodes:

```
1 ▶ <a href="http://blogsitestudio.com/web-services/web-design-
  estimate/" style="text-decoration: underline;">…</a>
```

Figure 5.10: An accessibility audit in Apple Safari. [Screenshot captured by Alexander Perko.]



94

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

ARIA — These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

▲ `button`, `link`, and `menuitem` elements do not have accessible names.

Contrast — These are opportunities to improve the legibility of your content.

▲ Background and foreground colors do not have a sufficient contrast ratio.

Names and labels — These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

▲ Links do not have a discernible name

Additional items to manually check (10) — These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

Passed audits (20)

● `[aria-*]` attributes match their roles
● `[aria-hidden="true"]` is not present on the document `<body>`
● `[aria-hidden="true"]` elements do not contain focusable descendents
● `[role]`s have all required `[aria-*]` attributes
● Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children.

Figure 5.11: An accessibility audit in Google Chrome. [Screenshot captured by Markus Stradner.]

Appendix A

Listings

A.1 AChart Creator Input CSV

Listing A.1 shows tabular data in CSV format about the population of Austria from 1959 to 2019 in 10-year increments, which was distilled from official Austrian statistics [SA 2021]. It is used as input for AChart Creator.

A.2 AChart Creator Line Chart

Listing A.2 shows the SVG source code including rich annotations generated by AChart Creator for a line chart with the dataset from Listing A.1.

A.3 AChart Creator Bar Chart

Listing A.3 shows the SVG source code including rich annotations generated by AChart Creator for a bar chart with the dataset from Listing A.1.

```
Year,Population
1959,7.014
1969,7.441
1979,7.549
1989,7.620
1999,7.992
2009,8.341
2019,8.878
```

Listing A.1: Tabular data in CSV format used as input for AChart Creator. The data shows the population of Austria from 1959 to 2019 in 10-year increments.

```
<svg viewBox="0 0 750 600" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:
    xlink="http://www.w3.org/1999/xlink" role="graphics-document">
  <style type="text/css">
    .line {
        fill: none;
        stroke-width: 3;
    }
    .overlay {
        fill: none;
        pointer-events: all;
    }

    /* Style the dots by assigning a fill and stroke */
    .dot {
        stroke: #fff;
    }

    .focus circle {
        fill: none;
        stroke: steelblue;
    }
  </style>
  <rect id="backdrop" width="750" height="600" fill="#fff"></rect>
  <g id="ChartRoot" role="chart" tabindex="0" transform="translate(100,100)" aria-
      labelledby="title desc" aria-charttype="line" aria-roledescription="Line Chart
      ">
    <desc id="desc">
      This chart shows the population of Austria from 1959 to 2019.
    </desc>
    <rect role="chartarea" width="600" height="400" fill="none"></rect>
    <text id="title" role="heading" text-anchor="middle" font-size="14" x="275" y="
        -25">
      Austrian Population over the Years
    </text>
    <g id="xScale" role="xaxis" aria-roledescription="x-Axis" aria-axistype="" aria-
        labelledby="x-title" tabindex="0" aria-valuemin="1959" aria-valuemax="2019"
        transform="translate(0,400)" fill="none" font-size="10" font-family="sans-
        serif" text-anchor="middle">
      <text y="50" x="300" text-anchor="middle" fill="black" font-size="12" role="
          heading" id="x-title">
        Year
      </text>
      <path class="domain" stroke="currentColor" d="M0.5,6V0.5H600.5V6"></path>
      <g class="tick" opacity="1" transform="translate(10.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x1">
          1960
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(60.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x2">
          1965
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(110.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x3">
```

Listing A.2: SVG source code of a line chart generated by AChart Creator with the dataset shown in Listing A.1.

```
          1970
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(160.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x4">
          1975
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(210.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x5">
          1980
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(260.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x6">
          1985
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(310.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x7">
          1990
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(360.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em">
          1995
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(410.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em">
          2000
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(460.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em">
          2005
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(510.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em">
          2010
        </text>
```

Listing A.2 (cont.)**:** SVG source code of a line chart generated by AChart Creator.

```
      </g>
      <g class="tick" opacity="1" transform="translate(560.5,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em">
          2015
        </text>
      </g>
    </g>
    <g id="yScale" role="yaxis" aria-roledescription="y-Axis" tabindex="0" aria-
        valuemin="4" aria-valuemax="13" aria-labelledby="y-title" fill="none" font-
        size="10" font-family="sans-serif" text-anchor="end">
      <text transform="rotate(-90)" y="-38" x="-200" text-anchor="middle" fill="
          black" font-size="12" role="heading" id="y-title">
        Population
      </text>
      <path class="domain" stroke="currentColor" d="M-6,400.5H0.5V0.5H-6"></path>
      <g class="tick" opacity="1" transform="translate(0,400.5)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y4">
          4
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,356.056)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y5">
          5
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,311.611)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y6">
          6
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,267.167)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y7">
          7
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,222.722)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y8">
          8
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,178.278)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y9">
          9
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,133.833)">
```

Listing A.2 (cont.): SVG source code of a line chart generated by AChart Creator.

```
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y10">
          10
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,89.389)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y11">
          11
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,44.944)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y12">
          12
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,0.5)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y13">
          13
        </text>
      </g>
    </g>
    <g id="dataarea1" role="dataset" aria-roledescription="Data Series" tabindex="0"
         aria-labelledby="dataset-title1">
      <title role="heading" id="dataset-title1">
        Population
      </title>
      <path class="line" d="M0,266.044C33.333,258.156,66.667,250.267,100,247.067C133
          .333,243.867,166.667,243.593,200,242.267C233
          .333,240.941,266.667,241.215,300,239.111C333
          .333,237.007,366.667,227.919,400,222.578C433
          .333,217.237,466.667,213.63,500,207.067C533
          .333,200.504,566.667,191.852,600,183.2" stroke="#66c2a5"></path>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-1">
        <title role="heading" id="name1-1">
          1959
        </title>
        <circle class="dot" cx="0" cy="266.044" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-1">
          7.014
        </title>
      </g>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-2">
        <title role="heading" id="name1-2">
          1969
        </title>
        <circle class="dot" cx="100" cy="247.067" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-2">
          7.441
        </title>
      </g>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-3">
        <title role="heading" id="name1-3">
          1979
```

Listing A.2 (cont.)**:** SVG source code of a line chart generated by AChart Creator.

```
          1979
        </title>
        <circle class="dot" cx="200" cy="242.267" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-3">
          7.549
        </title>
      </g>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-4">
        <title role="heading" id="name1-4">
          1989
        </title>
        <circle class="dot" cx="300" cy="239.111" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-4">
          7.62
        </title>
      </g>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-5">
        <title role="heading" id="name1-5">
          1999
        </title>
        <circle class="dot" cx="400" cy="222.578" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-5">
          7.992
        </title>
      </g>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-6">
        <title role="heading" id="name1-6">
          2009
        </title>
        <circle class="dot" cx="500" cy="207.067" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-6">
          8.341
        </title>
      </g>
      <g tabindex="0" role="datapoint" aria-labelledby="name1-7">
        <title role="heading" id="name1-7">
          2019
        </title>
        <circle class="dot" cx="600" cy="183.2" r="5" fill="#66c2a5"></circle>
        <title role="datavalue" id="value1-7">
          8.878
        </title>
      </g>
    </g>
  </g>
</svg>
```

Listing A.2 (cont.)**:** SVG source code of a line chart generated by AChart Creator.

```
<svg viewBox="0 0 750 600" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" role="graphics-document">
  <style type="text/css">
    .bar {fill: steelblue; }
  </style>
  <rect id="backdrop" width="750" height="600" fill="#fff"></rect>
  <g id="ChartRoot" role="chart" tabindex="0" transform="translate(100,100)" aria-labelledby="title desc" aria-charttype="bar" aria-roledescription="Bar Chart">
    <desc id="desc">
      This chart shows the population of Austria from 1959 to 2019.
    </desc>
    <rect role="chartarea" width="600" height="400" fill="none"></rect>
    <text id="title" role="heading" text-anchor="middle" font-size="14" x="275" y="-25">
      Austrian Population over the Years
    </text>
    <g id="xScale" role="xaxis" aria-axistype="category" aria-roledescription="x-Axis" aria-labelledby="x-title" tabindex="0" transform="translate(0,400)" fill="none" font-size="10" font-family="sans-serif" text-anchor="middle">
      <text y="50" x="300" text-anchor="middle" fill="black" font-size="12" role="heading" id="x-title">
        Year
      </text>
      <path class="domain" stroke="currentColor" d="M0.5,6V0.5H600.5V6"></path>
      <g class="tick" opacity="1" transform="translate(56.757,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x1">
          1959
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(137.838,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x2">
          1969
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(218.919,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x3">
          1979
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(300,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x4">
          1989
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(381.081,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x5">
          1999
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(462.162,0)">
        <line stroke="currentColor" y2="6"></line>
```

Listing A.3: SVG source code of a bar chart generated by AChart Creator with the dataset shown in Listing A.1.

```
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x6">
          2009
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(543.243,0)">
        <line stroke="currentColor" y2="6"></line>
        <text fill="currentColor" y="9" dy="0.71em" role="axislabel" id="x7">
          2019
        </text>
      </g>
    </g>
    <g id="yScale" role="yaxis" aria-roledescription="y-Axis" tabindex="0" aria-
        valuemin="4" aria-valuemax="13" aria-labelledby="y-title" fill="none" font-
        size="10" font-family="sans-serif" text-anchor="end">
      <text transform="rotate(-90)" y="-38" x="-200" text-anchor="middle" fill="
          black" role="heading" id="y-title" font-size="12">
        Population
      </text>
      <path class="domain" stroke="currentColor" d="M-6,400.5H0.5V0.5H-6"></path>
      <g class="tick" opacity="1" transform="translate(0,400.5)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y1">
          4
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,356.056)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y2">
          5
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,311.611)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y3">
          6
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,267.167)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y4">
          7
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,222.722)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y5">
          8
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,178.278)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y6">
          9
```

Listing A.3 (cont.)**:** SVG source code of a bar chart generated by AChart Creator.

```
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,133.833)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y7">
          10
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,89.389)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y8">
          11
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,44.944)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y9">
          12
        </text>
      </g>
      <g class="tick" opacity="1" transform="translate(0,0.5)">
        <line stroke="currentColor" x2="-6"></line>
        <text fill="currentColor" x="-9" dy="0.32em" role="axislabel" id="y10">
          13
        </text>
      </g>
    </g>
    <g id="dataarea" role="dataset" tabindex="0">
      <title>
        Population
      </title>
      <g tabindex="0" transform="translate(32.432,266.044)" role="datapoint" aria-
         labelledby="x1">
        <rect class="bar" width="48.649" height="133.956"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white"
           role="datavalue" id="value1">
          7.014
        </text>
      </g>
      <g tabindex="0" transform="translate(113.514,247.067)" role="datapoint" aria-
         labelledby="x2">
        <rect class="bar" width="48.649" height="152.933"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white"
           role="datavalue" id="value2">
          7.441
        </text>
      </g>
      <g tabindex="0" transform="translate(194.595,242.267)" role="datapoint" aria-
         labelledby="x3">
        <rect class="bar" width="48.649" height="157.733"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white"
           role="datavalue" id="value3">
          7.549
        </text>
      </g>
      <g tabindex="0" transform="translate(275.676,239.111)" role="datapoint" aria-
         labelledby="x4">
```

Listing A.3 (cont.)**:** SVG source code of a bar chart generated by AChart Creator.

```
        <rect class="bar" width="48.649" height="160.889"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white" role="datavalue" id="value4">
          7.62
        </text>
      </g>
      <g tabindex="0" transform="translate(356.757,222.578)" role="datapoint" aria-labelledby="x5">
        <rect class="bar" width="48.649" height="177.422"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white" role="datavalue" id="value5">
          7.992
        </text>
      </g>
      <g tabindex="0" transform="translate(437.838,207.067)" role="datapoint" aria-labelledby="x6">
        <rect class="bar" width="48.649" height="192.933"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white" role="datavalue" id="value6">
          8.341
        </text>
      </g>
      <g tabindex="0" transform="translate(518.919,183.2)" role="datapoint" aria-labelledby="x7">
        <rect class="bar" width="48.649" height="216.8"></rect>
        <text x="24.325" y="10" text-anchor="middle" font-size="10" fill="white" role="datavalue" id="value7">
          8.878
        </text>
      </g>
    </g>
  </g>
</svg>
```

Listing A.3 (cont.): SVG source code of a bar chart generated by AChart Creator.

Bibliography

AMC [2021]. *Accessibility in amCharts 4*. amCharts. `https://amcharts.com/accessibility/accessible-charts` (cited on page 12).

Andrews, Keith and Christopher Alexander Kopel [2021a]. *AChart Creator*. 09 May 2021. `https://github.com/tugraz-isds/achart-creator` (cited on pages 12, 19).

Andrews, Keith and Christopher Alexander Kopel [2021b]. *AChart Interpreter*. 09 May 2021. `https://github.com/tugraz-isds/achart-creator` (cited on pages 12, 21).

Apple [2021]. *VoiceOver User Guide*. Apple, 09 May 2021. `https://support.apple.com/de-at/guide/voiceover/welcome/mac` (cited on page 17).

Assistiv [2021]. *Narrator Screen Reader*. Assistiv Labs, 09 May 2021. `https://assistivlabs.com/assistive-tech/screen-readers/narrator` (cited on page 17).

Barth, Cooper F. and Thomas B. McHugh [2021]. *Bri1liant*. 09 May 2021. `https://github.com/InclusiveTechNU/bri1liant/` (cited on page 23).

Bellamy-Royds, Amelia [2021]. *SVG Accessibility/ARIA roles for charts*. `https://w3.org/wiki/SVG_Accessibility/ARIA_roles_for_charts` (cited on page 12).

Damera, Suman [2021]. *Top 5 Rules of ARIA*. 09 May 2021. `https://deque.com/blog/top-5-rules-of-aria/` (cited on page 3).

FC [2021]. *Accessibility Extension for FusionCharts Beta*. FusionCharts. `https://semiotic.nteract.io/guides/accessibility` (cited on page 12).

FS [2021]. *JAWS*. Freedom Scientific, 09 May 2021. `https://freedomscientific.com/products/software/jaws/` (cited on page 17).

HC [2020]. *Accessibility Module Feature Overview*. Highcharts, 11 May 2020. `https://highcharts.com/` (cited on page 12).

Kasprzyk, Michał, Marc Prud'hommeaux, and Severino Ribecca [2021]. *Glimpse I/O*. 09 May 2021. `https://glimpse.io/` (cited on page 19).

Meeks, Elijah [2021]. *Semiotic – Accessibility*. `https://semiotic.nteract.io/guides/accessibility` (cited on page 12).

NVA [2021]. *About NVDA*. NV Access, 09 May 2021. `https://nvaccess.org/about-nvda/` (cited on page 17).

SA [2021]. *Total population (Annual Average)*. Statistik Austria, 09 May 2021. `https://statistik.at/web_en/statistics/PeopleSociety/population/population_stock_and_population_change/total_population_annual_average/index.html` (cited on page 26).

Schepers, Doug [2021a]. *Describler*. 09 May 2021. `http://describler.com/` (cited on pages 12, 22).

Schepers, Doug [2021b]. *Describler Source Code*. 09 May 2021. `https://github.com/shepazu/describler/` (cited on page 23).

Van der Schee, Max [2021]. *Web Accessibility*. 09 May 2021. `https://github.com/mvdschee/web-accessibility` (cited on page 23).

W3C [2018]. *WAI-ARIA Graphics Module*. World Wide Web Consortium, 02 Oct 2018. `https://w3.org/TR/graphics-aria-1.0/` (cited on page 4).

W3C [2021a]. *Introduction to Web Accessibility*. World Wide Web Consortium, 09 May 2021. `https://w3.org/WAI/fundamentals/accessibility-intro/` (cited on pages 1, 3).

W3C [2021b]. *Making the Web Accessible*. World Wide Web Consortium, 09 May 2021. `https://w3.org/WAI/` (cited on page 3).

W3C [2021c]. *WAI-ARIA Overview*. World Wide Web Consortium, 09 May 2021. `https://w3.org/WAI/standards-guidelines/aria/` (cited on page 3).