# Olli: An Extensible Visualization Library for Screen Reader Accessibility

Matthew Blanco*
Northeastern University

Jonathan Zong†
Massachusetts Institute of Technology

Arvind Satyanarayan‡
Massachusetts Institute of Technology

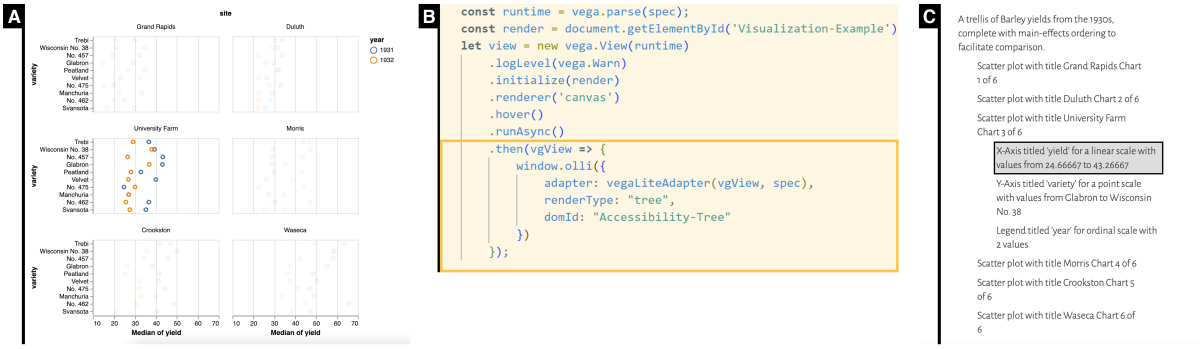


Figure 1: A) An example visualization. B) Example Olli code using a Vega-Lite adapter. C) An accessible tree view rendered by Olli.

## Abstract

Though recent research has explored the design of rich screen reader visualization experiences, accessible visualizations for blind and low vision users remain rare on the web. While some visualization toolkits offer accessible solutions, toolkit-specific implementations can present idiosyncratic user experiences that limit learnability. We present Olli, an open source library that converts visualizations into a keyboard-navigable structure accessible to screen readers. Using an extensible adapter design pattern, Olli is agnostic to the specific toolkit used to author the visualization. Olli renders a chart as an accessible tree view following the HTML Accessible Rich Internet Applications (ARIA) standard. Olli helps visualization developers easily create accessible visualizations across visualization toolkits.

**Index Terms:** Human-centered computing—Visualization—Accessibility systems and tools; Human-centered computing—Visualization—Visualization

## 1 Introduction

Most visualizations on the web are not accessible to blind and low vision screen reader users. A screen reader is assistive software that reads text and image content as speech. Lack of accessible, usable visualization experiences excludes screen reader users from accessing information and participating in conversations about data. Blind and low vision users have the same information seeking goals as sighted users when reading visualizations, seeking high level overviews followed by detailed descriptions [3]. However, these goals are not fulfilled by current best practices for accessibility, such as providing alt text or data tables [7].

While some visualization toolkits offer custom screen reader interactions for accessible reading experiences, these implementations have limited reusability across toolkits and lack shared user experience conventions. For instance, Highcharts [2] offers custom keyboard navigation within a visualization, but only for charts created with Highcharts. VoxLens [4] supports adding accessible interactions to a pre-defined set of three visualization toolkits, but can potentially conflict with other plug-ins' key bindings and ARIA attributes. As a result, it is currently difficult for visualization developers to ensure the *reusability* of solutions across visualizations made with different toolkits, and encourage *standardization* of screen reader user experiences.

We created Olli, an open-source JavaScript library that converts existing visualizations from multiple libraries into a keyboard-navigable structure with text descriptions at varying levels of detail. Olli's user experience is informed by prior work identifying design dimensions for richer screen reader experiences of visualizations for blind and low vision users [7]. Using an *adapter design pattern* [1], Olli can be extended to support any JavaScript visualization library by implementing a function that adapts existing chart instances into a standard `OlliVisSpec` interface. This interface details its structural, hierarchical, and visual components, that are then rendered as a tree view in HTML compatible with the Accessible Rich Internet Applications (ARIA) standard. A user can use the arrow keys to navigate the tree view, which follows existing ARIA standards. The adapter pattern allows developers to extend Olli to support new visualization toolkits, and provides a standard rendering format compatible with existing screen reader standards and conventions.

Olli makes screen reader accessibility easier for developers to incorporate into existing visualizations. With Olli's initial release, we include adapter implementations for Vega, Vega-Lite, and Observable Plot. Following the adapter pattern, other developers can contribute additional adapters for other commonly used visualization libraries. Because Olli offers a standard set of user experiences that can be repeated across a variety of charts, users can learn one set of interactions that they can take with them across different visualizations. Screen reader users benefit from Olli's interactions that offer access to data at varying levels of information granularity—from high level summaries to individual data points. Olli is available as open-source software at: *https://github.com/mitvis/olli*.

## 2 Olli's System Design

There are three main parts that make up Olli: the adapter interface for supporting external visualization toolkits, the abstract model for accessible visualization structures, and the renderer that outputs abstracted visualization structures as ARIA-tagged HTML.

*e-mail: blanco.m@northeastern.edu
†e-mail: jzong@mit.edu
‡e-mail: arvindsatya@mit.edu

### 2.1 Adapters

Because visualization toolkits are designed with different trade-offs in mind, their APIs can vary widely [6]. In order to support adding accessibility to the widest possible range of visualizations, Olli uses an *adapter design pattern* to convert visualization specifications from various toolkits into a common interface that can then be rendered as accessible HTML. Olli accomplishes this by wrapping an instance of another visualization toolkit within an adapter function, which returns an `OlliVisSpec` object that corresponds to the visualization. Olli then constructs a hierarchical data structure containing descriptions for elements of the visualization, which is then rendered as an accessible tree view. Because the accessible structure is constructed from the standard `OlliVisSpec` interface, this process is agnostic to the details of the toolkit with which the original visualization was implemented.

To extend Olli's coverage to support adding screen reader accessibility to a new toolkit, developers can simply implement an adapter function for that toolkit, without needing to re-implement the UX details of the accessible visualization. This lowers the barrier for visualization authors who lack specialized accessibility expertise to offer accessible visualization experiences.

#### 2.1.1 OlliVisSpec

An adapter takes in a visualization toolkit's output (e.g. an SVG, or a scenegraph instance) and its original specification, and returns that visualization as an object implementing the `OlliVisSpec` interface.

An `OlliVisSpec` object either describes a single visualization, or contains a list of objects that each describe a single view of a multi-view chart. Each object has information about a chart's visual elements, including its mark type and its *guides* (i.e. axes and legends). It also has a list of names of data fields participating in visual encodings, and other metadata such as the title.

Each view's `Guide` objects contain a title, the name of the field mapped to the axis/legend, and other metadata (e.g. the axis orientation or legend type). They also include information needed to divide axes and legends down into smaller sections (i.e., interval extents for continuous guides and categories for discrete guides), and a reference to the underlying data.

### 2.2 Accessible Structures

Like sighted users, research has found that screen reader users also follow an information seeking strategy of "overview first, zoom and filter, and details on demand" [3, 7]. Following prior work on design dimensions for rich screen reader visualization experiences [7], we represent accessible visualizations hierarchically, as a tree structure containing descriptions at varying levels of information granularity. The root node contains a high-level overview of the visualization. If the visualization has multiple views, the level below the root contains a node for each view. The next level contains nodes representing guides (axes and legends). Each guide node has children representing intervals and categories for continuous and discrete guides, respectively. Finally, the leaves of the tree contains the individual data points that correspond to those intervals and categories. The tree allows users to leverage a visualization's hierarchical structure to drill down into data, rather than being restricted to reading individual data points linearly or in a table.

#### 2.2.1 AccessibilityTreeNode

Olli uses an `OlliVisSpec` returned by an adapter to construct a tree of `AccessibilityTreeNode`s. Each `AccessibilityTreeNode` contains a reference to its parent node (null in the case of the root node), and a list of child nodes. It also contains a information about what part of the visualization it represents, and a textual description to be read by a screen reader. For example, a node representing an x-axis might have a text description reading, *"X-Axis for a quantitative scale with values from 14.43 to 34.7"*.

### 2.3 Rendering

Olli renders an accessible structure by traversing the tree and outputting HTML elements and necessary ARIA attributes. To implement an accessible HTML tree view, we adapted an example from the W3C's WAI-ARIA Authoring Examples documentation [5]. As the `AccessibilityTreeNode` is traversed, tree nodes that have children are rendered as a nested unordered list with a `group` role and `aria-expanded` attribute. Otherwise, a node is rendered as a list item with a `treeitem` ARIA role. The addition of the ARIA roles and extra attributes allow the screen reader to provide a more specific description of the node's position of the tree.

## 3 Usage Example

Consider the case of a visualization developer who wants to offer an accessible version of an existing Vega-Lite chart (Fig. 1A). They first pass their Vega-Lite chart instance and specification to Olli's `VegaLiteAdapter` function, which returns an object adapting the chart's specification to Olli's standard interface. They then pass this object to the `olli` function, along with a config object (Fig. 1B). At minimum, this config object includes a selector for a DOM element inside which to render the accessible visualization. The developer can optionally also specify alternate render formats, such as tree view or table. With these few steps, Olli can render an accessible version of the chart on a webpage (Fig. 1C).

## 4 Conclusion

Olli contributes a system for converting existing visualizations into keyboard-navigable structures accessible to screen readers, and a set of abstractions for extending support to additional visualizations using an adapter pattern. With the initial release of Olli, we provide adapter implementations for Vega, Vega-Lite, and Observable Plot. By releasing Olli as an open-source project, we hope that community members can contribute new adapters for visualization libraries, and refine the user experience of Olli's rendered output as accessible design guidelines evolve. By providing a reusable way to make visualizations accessible, and by providing a standardized screen reader UX for accessible charts, Olli can help make visualization accessibility more widespread on the web.

## Acknowledgments

We thank Josh Pollock and Daniel Hajas for feedback and support.

## References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., USA, 1995.

[2] Highcharts. Accessible line chart, 2021. https://www.highcharts.com/demo/accessible-line.

[3] A. Sharif, S. S. Chintalapati, J. O. Wobbrock, and K. Reinecke. Understanding Screen-Reader Users' Experiences with Online Data Visualizations. In *ACM Conference on Computers and Accessibility (SIGACCESS)*, ASSETS '21, pp. 1–16. New York, NY, USA, 2021. doi: 10.1145/3441852.3471202

[4] A. Sharif, O. H. Wang, A. T. Muongchan, K. Reinecke, and J. O. Wobbrock. VoxLens: Making Online Data Visualizations Accessible with an Interactive JavaScript Plug-In. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–19. ACM, New Orleans LA USA, Apr. 2022.

[5] w3c. Navigation Treeview Example, 2021. https://w3c.github.io/aria-practices/examples/treeview/treeview-navigation.html.

[6] K. Wongsuphasawat. Encodable: Configurable Grammar for Visualization Components. In *2020 IEEE Visualization Conference (VIS)*, pp. 131–135. IEEE Computer Society, Los Alamitos, CA, USA, Oct. 2020. doi: 10.1109/VIS47514.2020.00033

[7] J. Zong, C. Lee, A. Lundgard, J. Jang, D. Hajas, and A. Satyanarayan. Rich Screen Reader Experiences for Accessible Data Visualization. *Computer Graphics Forum*, 2022. doi: 10.1111/cgf.14519