Wikxhibit: Using HTML and Wikidata to Author Applications that Link Data Across the Web

Wikxhibit: Using HTML and Wikidata to Author Applications that Link Data Across the Web

Tarfah Alrashed
MIT CSAIL
tarfah@mit.edu

Lea Verou
MIT CSAIL
leaverou@mit.edu

David R. Karger
MIT CSAIL
karger@mit.edu

ABSTRACT

Wikidata is a companion to Wikipedia that captures a substantial part of the information about most Wikipedia entities in machine-readable structured form. In addition to directly representing information from Wikipedia itself, Wikidata also cross-references how additional information about these entities can be accessed through APIs on hundreds of other websites.

This trove of valuable information has become a source of numerous domain-specific information presentations on the web, such as art galleries or directories of actors. Developers have created a number of such tools that present Wikidata data, sometimes combined with data accessed through Wikidata's cross-referenced web APIs. However, the creation of these presentations requires significant programming effort and is often impossible for non-programmers.

In this work, we empower users, even non-programmers, to create presentations of Wikidata and other sources of data on the web, using only HTML with no additional programming. We present Wikxhibit, a JavaScript library for creating HTML-based data presentations of data on Wikidata and the other data APIs it cross-references. Wikxhibit allows a user to author plain HTML that, with the addition of a few new attributes, is able to dynamically fetch and display any Wikidata data or its cross-referenced Web APIs. Wikxhibit's JavaScript library uses Wikidata as the bridge to connect all the cross-referenced web APIs, allowing users to aggregate data from multiple Web APIs at once, seamlessly connecting object to object, without even realizing that they are pulling data from multiple websites. We integrate Wikxhibit with Mavo, an HTML language extension for describing web applications declaratively, to empower plain-HTML authors to create presentations of Wikidata. Our evaluation shows that users, even non-programmers, can create presentations of Wikidata and other sources of web data using Wikxhibit in just 5 minutes.

CCS CONCEPTS

• **Information systems** → **Web interfaces**; **Mashups**; *Web services*; Mediators and data integration.

KEYWORDS

Wikidata, Web APIs, Mashup, Data Integration, Semantic Web

UIST '22, October 29-November 2, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10.
https://doi.org/10.1145/3526113.3545706

ACM Reference Format:
Tarfah Alrashed, Lea Verou, and David R. Karger. 2022. Wikxhibit: Using HTML and Wikidata to Author Applications that Link Data Across the Web. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3526113.3545706

1 INTRODUCTION

There has been a longstanding vision of somehow taking all the valuable data that is distributed through the web and aggregating it into some coherent structure that could be used to drive powerful web applications. But the diversity of formats, schemas, and protocols has proven a formidable barrier. Large scale standardization efforts have been put forward, such as the Semantic Web and Linked Open Data [3], the Open Graph Protocol [14], and Schema.org [12], but none of these has become dominant. Instead, individual programmers with specific application needs have crafted special purpose code that accesses and merges together data from specific website APIs [9].

In this work, we propose a new, alternative approach to crafting applications that merge data across multiple websites. At its center is Wikidata [32], a companion to Wikipedia that holds structured data about many Wikipedia entities in a structured form, and *also* provides many links to external information about these entities on other websites. We combine this "data directory" with an *API description language* that can be used to *declaratively specify* how to access data behind a particular web API, and a *data templating language* that can be used to author presentations of structured data using only HTML. Combined, these three components permit an HTML author, using no additional programming, to create rich presentations that aggregate and link data from multiple API-backed websites.

To construct an application that incorporates data from multiple distinct websites, a person needs to accomplish several things:

- *access* the data needed from each website
- *normalize the schema* so that the data coming from the different websites can be treated the same way
- *link entities* to combine data about the same entity found on distinct sites
- *present* the information in appealing, human-readable form.

Wikxhibit incorporates 4 components to tackle the tasks. Shapir [2] uses a declarative API description language to simplify access to distinct web APIs and uses Schema.org [12] to normalize the different site schemas. We use Wikidata to provide the crosslinking information needed to unify entities. And we use Mavo [30, 31] to generate rich interactive presentations of the resulting unified information. Figure 1 shows a Wikxhibit application—described entirely in HTML—that integrates and presents data from different websites.

```
<div mv-app mv-source="wikxhibit" mv-source-service="wikidata" mv-source-id="Q19848">
    <img property="image" />
    <h1 property="label"></h1>
    ...
    <div property="SpotifyArtist" >
        <div property="albums" mv-multiple>
            <img property="image" />
            <h5 property="name"></h5>
            <p property="numTracks"></p>
        </div>
        <div property="tracks" mv-multiple>
            <iframe src="[embedUrl]"></iframe>
        </div>
    </div>
    <div property="YouTubeChannel">
        <div property="videos" mv-multiple>
            <iframe src="[embedUrl]"></iframe>
        </div>
    </div>
    <div property="SongkickArtist">
        <div property="events" mv-multiple>
            <h5 property="name"></h5>
            <p property="location"></p>
            <p property="startDate"></p>
        </div>
    </div>
</div>
```

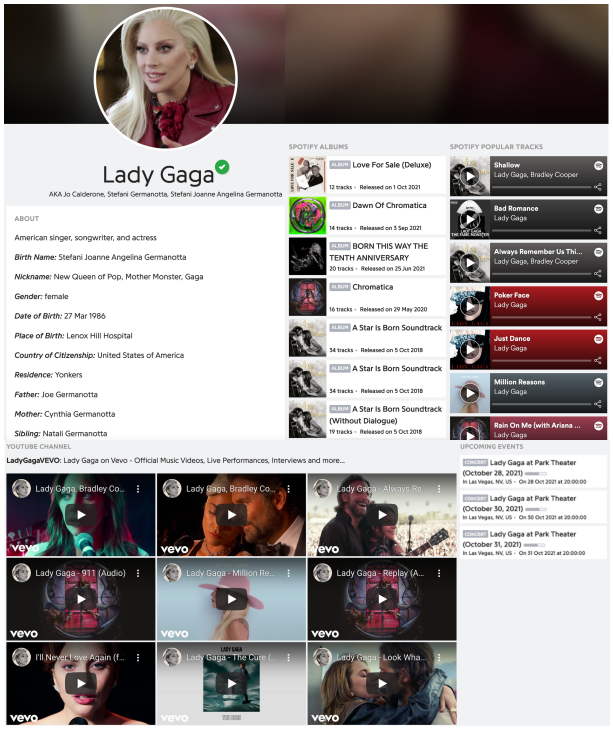


Figure 1: An artist page made with Wikxhibit that displays integrated data from different websites: general information about the artist from Wikidata, their albums and tracks from Spotify, their videos from YouTube, and their events from Songkick

Our Contribution

This paper demonstrates the viability of easily crafting applications that link data across multiple websites that describe the same entity, using only HTML with no additional programming. Our contribution comes as a sum of many parts. We depend on these parts, but Wikxhibit offers the architectural vision and additions needed to achieve it.

A key contribution is the idea of using Wikidata as a universal join table to cross-reference entities across different third-party APIs. However, Wikidata alone does not suffice. End-users can not effectively leverage Wikidata alone to interact with the data behind other sites that this join table refers to, let alone build applications that merge data from these different sites. Combining that universal join table with Mavo-Shapir gives us a whole that is greater than the sum of its parts.

Conversely, although Mavo-Shapir lets users build applications that combine data from multiple APIs, such as conference papers from IEEE and recorded conference talks from YouTube, it offers no mechanism to link data from these multiple APIs about a single entity. For example, to indicate that a particular paper from IEEE is the one being presented in the video. A user would have to write code (e.g., JavaScript) to link data from multiple sites that describe the same entity. This paper demonstrates the idea of using Wikidata to do so and solves the design problem of extending Mavo-Shapir to express (in HTML) and carry out the desired entity unification.

Making entity resolution available and showing how these components can work together to give end-users something powerful is part of our technical contribution. All prior mashup tools, including Shapir, expect users to develop their heuristics for entity resolution across different sites. These prior tools could have used Wikidata, but none of them did, possibly because of how challenging it would be to incorporate it manually.

In a survey of Wikidata users that we conducted, they expressed the desire to create presentations of Wikidata and the challenges they faced creating these presentations with enriching them with data from other websites. We conducted two user studies with 12 Wikidata users to test whether they could use Wikxhibit to author HTML-based applications that present data combining Wikidata and data from other sites. Not only were participants able to create applications that combine data from different websites in as little as 5 minutes, but they were enthusiastic about the ease with which they could do so.

Our contribution can be seen in two ways. From a Wikidata-centric perspective, we are making it easy to incorporate data from all over the web into the Wikidata entities, broadening the reach of Wikipedia. From a web-wide perspective, we show how Wikidata can be used as the interchange for linking data across arbitrary websites. While Wikxhibit supports both perspectives, our evaluation has focused on the Wikidata community, who have shown a keen

interest in generating presentations of Wikidata that incorporates information from other websites.

2 RELATED WORK

Our work builds on contributions from several research areas including the Semantic Web, mashup tools and data integration.

2.1 The Semantic Web

The Semantic Web provides a common framework that allows data to be shared and reused across different organizations. Fundamental to adopting the Semantic Web vision was the development of SPARQL, a protocol and query language for RDF data. The vision of the Semantic Web is that if all websites expose their data through a SPARQL endpoint, people will be able to access it uniformly.

However, despite SPARQL's widespread adoption within the Semantic Web community, it has not spread more widely. For web developers, the Semantic Web is unpopular because of its complexity [9]. SPARQL is associated with a high learning curve, which prevents its wide adoption by developers who typically access and query data using web APIs. When RDF data is exposed through SPARQL endpoints, many developers find it difficult to access. Thus, there were several efforts aiming to convert any given SPARQL endpoint into a simple REST API [9, 19, 26].

On the publisher side, there would be great value in having all the web data on the Semantic Web, but that value seems not to accrue to the companies publishing the data. Instead, a substantial amount of this data is made available through web APIs since that seems to be what developers want. Even some organizations that have adopted Semantic Web technologies to manage their data, such as data.gov and Wikidata [32], typically provide web APIs to serve their RDF data to developers.

If most websites exposed their data through SPARQL endpoints, we would be able to access and integrate data from these sites and use Mavo to allow end-users to build applications on top of this data. But given that is not the case, building Wikxhibit is necessary to connect data across different websites and create this uniform data model that HTML authors can access to create applications that present this data. Wikxhibit provides a less ambiguous but more practical approach than the Semantic Web by using Wikidata to connect all its cross-referenced APIs.

2.2 Mashup Tools

Kicked off by ChicagoCrime.org [15], a long line of research has explored ways to let end users create alternative presentations over data accessed through web APIs. The earliest mashups were hand-crafted by programmers. A later generation of tools, such as Marmite [37], Carpé Data [29] and D.mix [13] helped users author the applications but assumed the existence of pre-programmed web API connectors to access specific data sources or for programmers to create new ones.

Gneiss [5] gives users a spreadsheet-like interface to build application queries over any publicly accessible data APIs. However, it expects users to enter an API request URL with parameters manually; thus, it requires each user to learn the API and construct API query strings. Gneiss also expects the user to connect different APIs through their spreadsheet-like interface. Mavo [31] is a library that lets a user build an interactive application over any JSON data source simply by authoring and annotating an HTML document (we will use Mavo as part of our overall system). But like the other tools above, Mavo posits that programmers will provide the connectors to those JSON data sources.

Easing the burden further, Spinel [6] does help end-users connect data APIs to mobile applications without programming. Spinel has a form-based query-builder for describing these APIs. However, similar to previous mashup tools, Spinel does not connect these APIs and expects the user to traverse this connection. Shapir [2] is a system that empowers end-users to create interactive web applications that operate on data accessible through web APIs (we will use Shapir as Wikxhibit's web API access layer). Shapir allows relatively inexperienced programmers to describe and connect web APIs without writing code or assembling any query strings. Unlike previous mashup tools, Shapir standardizes the data model of web APIs that offer semantically similar data types. Thus, Shapir makes it possible for users to create an application over one web API and then use it, unchanged, with other APIs that offer semantically similar data. However, Shapir does not support the integration of heterogeneous types of data from different websites and expects the user to make this connection.

These prior tools focus more on building applications and mashups over easily accessible data. Wikxhibit would allow all these past tools to access this uniform data model of all web APIs and integrate them into their workflow. Wikxhibit's primary focus is to make linked data easily accessible. In addition, it allows relatively inexperienced programmers to access any web APIs without writing code or assembling any query strings. With Mavo, Wikxhibit also enables non-programmers to create HTML applications to query and access web APIs.

These prior tools focus on querying and retrieving individual API data; none supports easy access to heterogeneous types of data from different web APIs, which Wikxhibit supports.

Data flow language tools such as Yahoo Pipes [25], and Node-RED [23] allow users to edit and connect data through APIs. Node-RED allows users to configure individual HTTP requests and connect them using a web form. Node-RED does not offer an easy way for end-users to link data from Node-RED to their applications. Users need to make changes to their application and write code to make this connection. In addition, studies have found that the data flow representation is often difficult for end-users to understand [4].

2.3 Data Integration

Several efforts have been made to help users integrate data from multiple sources. Research on mashup tools has provided ways to let users extract and integrate web data from different websites without having to write conventional code [10]. Mashup tools like MashMaker [11] and Vegemite [18] allow users to scrape web data directly from web pages and allow users to create a mashup by browsing and combining different web pages. Potluck [16] is another data integration tool that allows users to combine, clean, and merge data coming from various sources. However, most of these mashup tools do not support the reuse and integration of the created Mashups. In addition to mashup tools, the Semantic Web vision includes representations like RDFS and OWL that can be

used to drive inference engines able to transform data between multiple schemas. RDFS aims to support transformation between any two schemas at any time. Shapir offers a less ambitious but more practical approach to describing and executing a translation from a specific website's API to Schema.org standard as it fetches the data. No transformations need to be applied during computation. However, the integration that Shapir offers is limited to the types of data provided by APIs; only web APIs with semantically similar types of data can be integrated using Shapir. Wikxhibit uses Shapir's *type unification* of web APIs and, together with Wikidata, extends this unification to support the integration of many heterogeneous types of data from different websites.

3 BACKGROUND

Our system builds heavily on three pieces of prior work that we describe here: Wikidata, Mavo, and Shapir.

3.1 Wikidata

Wikidata [32] is a knowledge base with tens of millions of data entities or items that anyone can access and edit. Item refers to a real-world object, concept, or event that is given an identifier in Wikidata together with information about it. Each item has an identifier (which starts with "Q" followed by one or more digits), label, description, aliases, and properties with their values. Each Wikidata property has its identifier (which starts with "P" followed by one or more digits). Properties have their pages on Wikidata and are connected to items, resulting in a linked data structure.

Figure 2(left) shows a Wikidata item's page for the artist Lady Gaga, which contains the item's identifier (Q19848); label (Lady Gaga); description (American singer, songwriter, and actress); and other properties and their values such as `instance of` and `image`. Wikidata items are connected; for example, Lady Gaga (Q19848) is an `instance of` human (Q5). Some of the Wikidata properties are *external identifiers*. These identifiers link Wikidata items to external databases, authority control files, or encyclopedia articles about the entity; for example, an ISBN for a book or the unique part of the URL of a movie or an actor in IMDb. Wikidata cross-references how additional information about items or entities can be accessed through APIs on hundreds of other websites using these external identifiers. In Figure 2(left), we highlight three external identifiers which identify the artist Lady Gaga on Spotify, YouTube, and Songkick. The values for these external identifiers are links to external web pages. For example, the value for the `Spotify artist id` property is a URL that links to the Spotify web page for Lady Gaga. Wikxhibit uses these links to connect to the web APIs for these sites and fetch data from them about these entities. For example, Wikxhibit calls the Spotify artist API to fetch data about Lady Gaga from Spotify using the value of her `Spotify artist id` provided by Wikidata (more details about this in section 5.2).

Wikidata allows users to query and retrieve its data, mainly through its SPARQL Query Service GUI [35] and API. Recently, Wikidata introduced the Wikidata Query Builder [34], which provides a visual interface for building a simple Wikidata query for users with little or no experience in SPARQL. Wikidata Query Service allows users to view their query result (depending on the query) as a table, an image grid, or a simple visualization.

Developers used the Wikidata SPARQL endpoint and created several applications that present Wikidata data [36]. However, these applications were built by skilled programmers. And according to a discussion we had with Wikidata engineers, there are no existing tools provided by Wikidata or other organizations that simplify this process for programmers and make it possible for non-programmers to create presentations of Wikidata and its cross-referenced APIs.

3.2 Mavo

Mavo [30, 31] is a bidirectional HTML templating language that extends the declarative syntax of HTML to describe Web applications that manage, store and transform data. Mavo aligns hierarchically structured data to a hierarchically structured web page. Authors link a page to a data source, then add a few attributes and expressions to their HTML elements to transform a static web page into a persistent, data-driven web application.

Mavo uses popular cloud services for storing and retrieving app data, such as Dropbox or GitHub. Authors specify which of these predefined services they intend to use by using `mv-storage` (read-write) or `mv-source` (read-only) HTML attributes. Support for new APIs can be added by authoring JavaScript plugins.

Loading data from arbitrary APIs in Mavo applications is technically already *possible*. Any URL that returns JSON is a valid `mv-source` for Mavo. However, this requires authors to assemble API-invocation URLs manually. In addition, Mavo does not integrate data from different sources.

3.3 Shapir

Shapir [2] is a system that simplifies the work for users to create interactive web applications that operate on standardized data accessible through arbitrary web APIs. It consists of three related components: WoOPI, ShapirJS, and ShapirUI. WoOPI is a standardized, machine-readable API ontology that can describe an API in terms of objects conforming to the canonical type definitions provided by Schema.org [12]. ShapirJS is a JavaScript library that uses a WoOPI description to present the API's data as typed objects in the local environment. And ShapirUI is a graphical tool that lets even non-programmers create the required WoOPI descriptions using standard data types. These three components are connected. A person uses ShapirUI to describe an API, and ShapirUI generates a corresponding WoOPI description of it. The ShapirJS JavaScript client library can read that WoOPI description to provide simple, local-environment access to the data behind the API. The WoOPI description only needs to be authored once; then anyone can use it. ShapirJS is also integrated with Mavo to empower users to create applications interacting with APIs' data by writing only HTML, with no JavaScript programming required.

Shapir unifies web APIs based on the type of data they offer. Using Shapir, a user can write an application over one web API and then use it unchanged with other APIs that offer semantically similar data types. However, Shapir does not support the integration of many heterogeneous types of data from different websites.

In this work, we recognize that an object in one web API often describes the same real-world entity as an object in a different API. Thus, we use ShapirJS as the web API access layer for Wikxhibit,

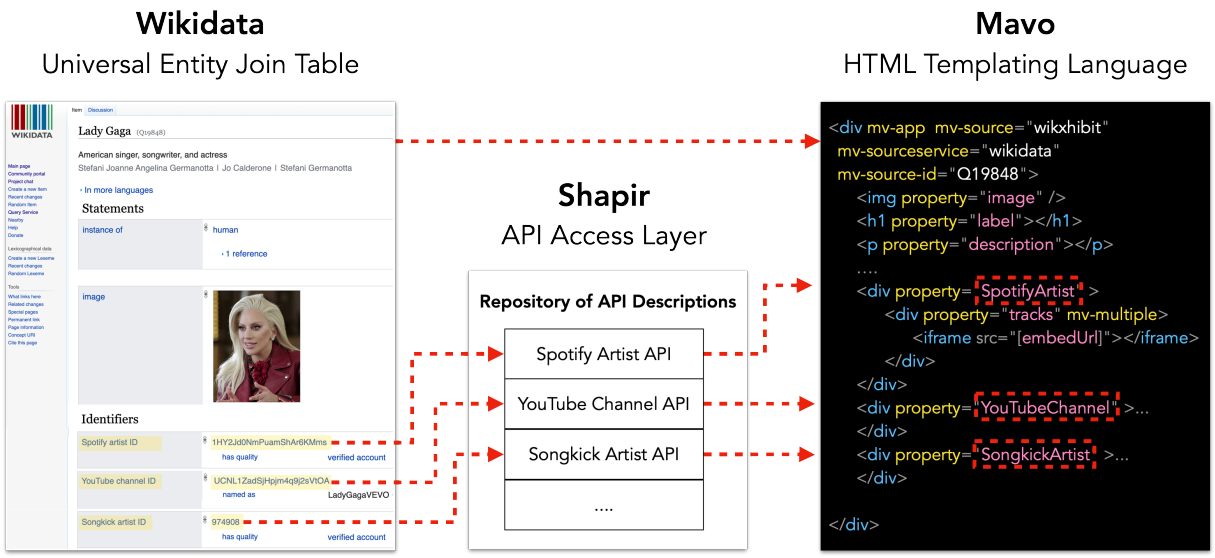


Figure 2: An illustration of the relationship between the different components that form Wikxhibit. We use Wikidata as a universal join table to cross-reference entities across different third-party APIs using its external identifiers (highlighted). We feed Shapir Wikidata's external IDs. We extend Mavo to support custom properties that hold the data returned by the APIs

allowing a user to create applications that link and interact with data drawn from multiple websites simultaneously.

Wikxhibit could be built on top of any web API access libraries (e.g., Swagger [27] or ScrAPIr [1]). But we choose to use ShapirJS because it offers additional advantages over the other API access libraries. Shapir presents the data behind a given web API as a uniform collection of readable and writeable standard-typed connected objects. Unlike accessing APIs directly or through any machine-generated library from a low-level API description, ShapirJS presents the object connections implicitly without requiring the user to invoke API endpoints to traverse the connections. The API's objects are connected through the properties of the object. For example, the artist object in Spotify is connected to its album and track objects. When users access the Spotify artist object, they can implicitly read the artist's albums and tracks.

4 EXPERIENCES WITH CREATING PRESENTATIONS OF WIKIDATA

We began by investigating people's experiences with Wikidata by surveying 25 Wikidata users; 75% of them use Wikidata daily, and the rest have used it several times. Participants use Wikidata for different purposes: 83% query Wikidata using the Wikidata Query Service, 33% query it using the Wikidata API, 75% edit Wikidata, and 16% build tools for/using Wikidata. Our participants come from the following backgrounds: Data Journalism, media art, biophysics, software engineering, tech support, and communications.

We asked *what* Wikidata presentations subjects wanted to create and *why*. We then asked *how* respondents tried to create these presentations and whether they succeeded.

90% of respondents expressed a desire to create presentations of Wikidata data, but only 44% had actually attempted to do so. Of those who attempted to create Wikidata presentations, only 20% were sometimes successful, while 22% were never successful. These results imply that while there is strong interest in creating presentations of Wikidata, there exist barriers to doing so, even among those who have the technical skills to complete the task. Table 1 lists some applications respondents are interested in creating.

All respondents who want to create presentations of Wikidata expressed the desire to integrate Wikidata's data with other data sources. Some of these presentations that respondents wanted to create but have not succeeded in are: aligning music entities from Wikidata with its information from MusicBrainz; unifying geo-entity in Wikidata with GeoNames and others; and creating a presentation that combines data from hi-knowledge.org with scholarly data from Wikidata. However, given that data integration from different websites is challenging, half of the respondents either never tried or tried and always failed.

Our respondents indicated three main obstacles to creating applications that integrate Wikidata with other data sources: (1) it requires advanced programming skills, which many of our respondents did not have; (2) it takes a lot of time and effort; and (3) it requires the unification of different data formats and schemas from multiple websites. Wikxhibit overcomes these obstacles.

5 WIKXHIBIT

Wikxhibit is a JavaScript library for creating HTML-based data presentations of Wikidata and its cross-referenced web APIs. Wikxhibit allows a user to author plain HTML that, with a few new attributes, can dynamically fetch and display any of the data that users can access through Wikidata and its cross-referenced APIs. Wikxhibit uses Wikidata as the bridge to connect all the cross-referenced APIs. It allows users to aggregate data from different websites at once, seamlessly connecting object to object, without even realizing that they are pulling data from multiple websites.

Presentations	Wikidata	Other
Display music entities from Wikidata along with information about them from MusicBrainz	✓	✓
Combine scholarly data from Wikidata with data from hi-knowledge.org	✓	✓
Display artworks with their provenances, collections, inventory numbers, etc., and integrate them with information in museum collections databases via the artwork IDs for the museums which are already in Wikidata	✓	✓
Integrate Wikidata data with data journalism curriculum and news projects	✓	✓
Unify geo-entity in Wikidata with GeoNames and others	✓	✓
List political parties of any country	✓	
Present information about traditional Ghanaian rulers	✓	
List superior courts in California	✓	

Table 1: Some of the applications that our survey respondents are interested in creating. Some of these applications integrate data from Wikidata and other data sources

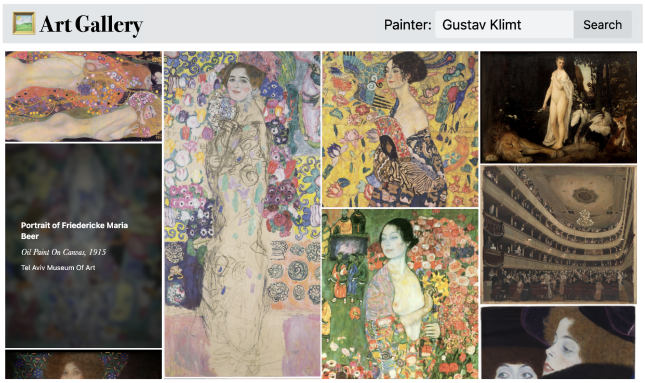
Wikxhibit integrates with Shapir, a system that standardizes access to web APIs by presenting the API's data as standardized typed objects in the user's local environment. This integration allows users to access data behind Wikidata's cross-referenced APIs. We integrate Wikxhibit with Mavo, an HTML language extension for describing web applications declaratively, to empower plain-HTML authors to create presentations of Wikidata. We implement Wikxhibit as a Mavo plugin that registers a new data source type. Mavo authors invoke it using `mv-source="wikxhibit"` on their Mavo root element (Figure 3, line 1). Wikxhibit provides Mavo users with a high-level syntax that does not require them to write SPARQL queries and make HTTP requests.

Wikxhibit creates this network of linked data objects and empowers end-users to author presentations in just HTML that fetch data from these web data objects and display their values. Users can create beautiful presentations over this virtual data model we created without understanding the technical details behind how it works.

This section describes the kind of applications end users can create using Wikxhibit.

5.1 Creating Presentations of Wikidata

Using Wikxhibit, users can author applications that read and present data from Wikidata in a few lines of HTML code. Figure 3 shows an art gallery application of paintings by the painter Gustav Klimt, and for each painting it displays its `image`, `title`, and `inception`. To present a list of paintings for a specific painter, a user can specify that they want to get paintings from Wikidata by specifying the type of entities they want to show using `mv-source-instance-of="painting"` (Figure 3, line 4) and in which language (line 3). Then they can use an input property (line 7) in order to collect the name of a painter that they would like to show using `mv-source-creator="[painter]"` (line 5). The user can then describe a template for each of the paintings they would like to display, and they do this by specifying which Wikidata properties they want to include, using the `property` attribute provided by Mavo, and how they want to show them (lines 9-11). To display multiple paintings, the user uses the `mv-multiple` attribute provided by Mavo (line 8), which will then make a copy of the painting template for each of the paintings they are displaying. The user can specify the number of paintings they would like to display by using the `mv-source-numberOfItems` property (line 6).



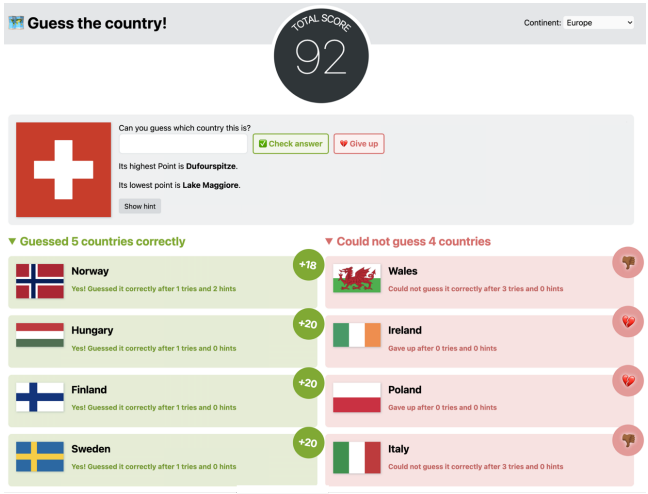

```
1  <div mv-app mv-source="wikxhibit"
2    mv-source-service="wikidata"
3    mv-source-language="en"
4    mv-source-instance-of="painting"
5    mv-source-creator="[painter]"
6    mv-source-numberOfItems="15">
7      <input property="painter" />
8      <div property="items" mv-multiple>
9          <img property="image" />
10         <h5 property="title"></h5>
11         <time property="inception"></time>
12     </div>
13 </div>
```

Figure 3: An art gallery of paintings by Gustav Klimt using Wikxhibit

In addition to presenting a list of data (e.g., paintings), users can create different types of applications that use data from Wikidata. Figure 4 shows a game application to guess the names of countries from their flags and other data about them, with the data being read from Wikidata.

5.1.1 Query Wikidata using human-readable properties and values. With the Wikidata Query Service, users need to query Wikidata using its SPARQL endpoint. Figure 5 shows a SPARQL query that fetches a list of paintings by Gustav Klimt.

With Wikxhibit, users do not need to learn SPARQL query language nor be familiar with the identifiers for the Wikidata properties and items. They can use the human-readable names or labels for the properties and entities, as provided by Wikidata. In the art gallery application in Figure 3, we use the Wikidata properties




```
<div mv-app mv-source="wikxhibit"
    mv-source-service="wikidata"
    mv-source-instance-of="country"
    mv-source-continent="[continent]">
    <!-- the rest of the HTML code can be found
         in the Wikxhibit site-->
</div>
```

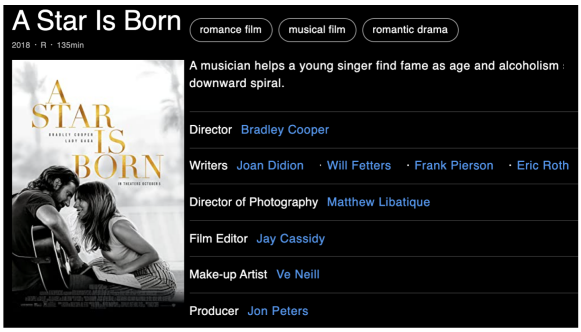
Figure 4: A game application to guess the names of countries from their flags and other data about them form Wikidata

`instance of` and `creator` to get a list of paintings by Gustav Klimt. The Wikxhibit library takes these property names and queries the Wikidata API to get the identifiers for these properties, P31 and P170, respectively. Then, the Wikxhibit library runs the SPARQL query in Figure 5 and returns a list of painting objects by Gustav Klimt with all of their properties and values.

```
SELECT *
WHERE{
    ?item wdt:P31 wd:Q3305213 .
    ?item wdt:P170 wd:Q34661 .
    ?item wdt:P1476 ?title .
    ?item wdt:P18 ?image .
    ?item wdt:P571 ?inception .
    SERVICE wikibase:label {
        bd:serviceParam
        wikibase:language "[AUTO_LANGUAGE],en".
    }
}
LIMIT 15
```

Figure 5: A SPARQL query to get a list of paintings by Gustav Klimt using the Wikidata Query Service

5.1.2 Access Wikidata linked entities. Wikidata consists of linked data. Wikxhibit allows users to traverse the connection between these linked data objects easily. Figure 6 shows a movie page that presents information about the movie and its cast members. In this application, we are displaying information about three linked entities: the `movie`, its `cast members`, and the schools where each cast member was `educated at`. To be able to traverse these linked entities and display information about them, the user will first need to specify the movie they would like to display using `mv-source-id="[movie]"` (line 3). Then, the user can display various information about the movie, like its `label`, `description`, and `image` (lines 4-6). To display information about the movie's cast members, the user can define a container element with `property="castMember"` (line 8). Within that container, they can display the properties of each cast member (lines 9-11), including the schools where each cast member was educated (line 12), which is a linked entity to the cast member. Wikxhibit makes it possible for users to traverse Wikidata's linked entities seamlessly through their properties.




```
<div mv-app  mv-source="wikxhibit"
  mv-source-service="wikidata"
  mv-source-id="[movie]">
    <h5 property="label"></h5>
    <p property="description"></p>
    <img property="image" />
    ...
    <div property="castMember" mv-multiple>
        <img property="image" />
        <h5 property="label"></h5>
        <p property="description"></p>
        <div property="educatedAt" mv-multiple>
            <h5 property="label"></h5>
        </div>
    </div>
</div>
```

Figure 6: A movie page that presents information about the movie and its cast members

5.1.3 Support Multiple Languages. Wikidata is an international and thus multilingual project. While English is the default language,

the project is intended to be used by users of every language. Wikidata users are familiar with Wikidata entities and properties in their language. Accordingly, Wikxhibit allows users to query Wikidata and get the data in their preferred language by specifying it using the `language` property. For example, a French user can get the paintings by Gustav Klimt using the properties `créateur` and `nature de l'élément`, and the returned results will be displayed in French.

5.1.4 Support Property and Item Identifiers. We choose to support human-readable properties and values because it is easier on users, specifically non-programmers or users who might not be very familiar with Wikidata identifiers. However, this approach has two limitations: Wikidata property names could change (but not often), and entity names are not necessarily unique. Nevertheless, their identifiers stay the same. For example, the Wikidata property `has parts` was changed by Wikidata editors to `has part or parts`, but its identifier remains the same (P527). Given that using identifiers is more robust, we also choose to support identifiers to query Wikidata using Wikxhibit.

5.2 Creating Presentations of Wikidata's Cross-referenced APIs

Wikxhibit connects Wikidata's linked data objects to objects on other websites, allowing users to create beautiful presentations over this virtual data model. Figure 1 shows an artist page that combines data from multiple websites: Wikidata, Spotify, YouTube, and Songkick. When the user reads data from Wikidata about a specific artist (line 1), they can access information about this artist from other websites through custom properties created by Wikxhibit to connect the artist object in Wikidata to objects on other sites (highlighted properties in lines 5, 15 and 20). The user can list the artist's `albums` (line 6) from Spotify and play their `tracks` (line 11), albums and tracks are objects connected to the Spotify artist object. In addition, to Spotify, the user can also play the artist's `videos` from YouTube and list their upcoming `events` from Songkick.

Wikxhibit provides a network of linked objects about a specific entity (the artist in this case) from various websites: Wikxhibit connects Wikidata artist object to Spotify artist object, YouTube channel object, and Songkick artist object. Moreover, each one of these objects is connected to other objects within the same site (e.g., YouTube `channel` has `videos`). Wikxhibit makes it easy for end-users to traverse these connections between objects from different websites without even knowing that they are moving from one website to the other.

Wikxhibit uses Wikidata, which cross-references how additional information about its entities can be accessed through other sites' APIs, as shown in Figure 2. Wikidata does not provide access to the data behind these web APIs. However, it provides external identifiers/URLs that *link* its entities to information about them on other web pages. Wikxhibit uses these external identifiers of other websites and queries these sites' APIs, then creates custom properties containing the data returned by these APIs. Wikxhibit integrates with Shapir to access these web APIs.

5.2.1 Integrate Wikxhibit with Shapir. We integrate Wikxhibit with ShapirJS, a JavaScript library that allows users to access data behind a web API as a uniform collection of readable and writeable standard-typed connected objects. Shapir acts as the web API access layer for Wikxhibit to fetch data about Wikidata's cross-referenced web APIs. Shapir has a repository of web APIs descriptions of multiple websites, which Wikxhibit uses to access these APIs and fetch data from them. Each API description describes the functions that fetch and update the different types of objects supported by the API. Wikxhibit uses the APIs' functions that fetch data about these APIs' objects and return that data in custom properties that it returns with Wikidata entities.

5.2.2 Create custom proprieties of external objects. This section describes how Wikxhibit connects Wikidata entities to data objects on other websites. When a user presents information about a Wikidata artist using Wikxhibit (Figure 1), Wikxhibit returns all the artist properties with their values from Wikidata. Wikxhibit then reads all the external identifiers for this artist, such as `Spotify artist ID`, `YouTube channel ID`, and `Songkick artist ID`. Moreover, for each identifier, Wikxhibit uses Shapir to access the appropriate site's API function or endpoint, which returns the object related to the artist from that site. For example, for the `Spotify artist ID`, Wikxhibit will access the Spotify API endpoint https://api.spotify.com/v1/artists/{Spotify-artist-ID} which returns an artist object from Spotify. Wikxhibit then creates a custom property `SpotifyArtist`, using the name of the external identifier, which contains all of the properties that Wikxhibit fetched from the Spotify artist API endpoint. Similarly, when Wikidata provides a `YouTube channel ID` and a `Songkick artist ID`. In that case, Wikxhibit will use these identifiers and query the YouTube and Songkick APIs and defines `YouTubeChannel` and `SongkickArtist` properties that contain that returned data from YouTube and Songkick APIs, respectively. These properties are highlighted in Figure 1.

We use Shapir because it implicitly presents APIs' object connections without requiring the user to invoke API endpoints to traverse the connections. For example, the user can access the albums and tracks for the artist from Spotify. Shapir maintains the ID of the artist internally so it can pass it to relevant API endpoints to read that artist's `album` and `track` objects. In addition, given that Shapir normalizes the data models of semantically similar data APIs, it makes it easier for users to integrate similar types of data from different sites like videos on YouTube and Vimeo.

5.2.3 Unify Entities Across Multiple Websites. Given that a Wikidata entity links to external sources of data, it provides this notion of a *uniform identifier* that identifies this entity on multiple websites. Using Wikxhibit, users can create the artist page in Figure 1 by referencing one of this artist's identifiers; it does not have to be the Wikidata one (in line 1). The user can use the `YouTube channel ID`, the `Spotty artist ID`, or any other identifier that represents this entity in Wikidata.

For example, if a user wants to refer to an entity on the web using an identifier, other than the Wikidata one, they can specify the identifier and the service that identifier belongs to via the `mv-source-service` and `mv-source-id` attributes. For example, we can create the same artist page in Figure 3, by replacing

`mv-source-service="wikidata"` with `mv-source-service="spotify"` and `mv-source-id="Q19848"` with the Spotify's artist ID `mv-source-id="1HY2Jd0NmPuamShAr6KMms"` or `mv-source-id="https://open.spotify.com/artist/1HY2Jd0NmPuamShAr6KMms"`.

We support this functionality because we would like to give access to this rich uniform data model, even to users who might not be familiar with Wikidata. So Wikxhibit allows users to refer to an entity on the web from any site and get a lot of information about them from different websites.

5.3 Demos and Code Generator

Wikxhibit's website provides demos that users can use to create their applications. Users can download the source code of these demos, and they can fork these demos using CodePen [7], an online HTML and CSS editor, and see the resulting page immediately. This really shows the power of a serverless system like Wikxhibit. Because there is no server, users can add their whole application on CodePen, so other people can fork it and edit it.

In addition to the demos, Wikxhibit provides a code generator that, based on the properties and values specified by the user, will generate a simple application that helps users get started with Wikxhibit. The objective of this code generator is to (1) help users find relevant properties and entities to query and display Wikidata's data and (2) help users know the properties that are returned by Wikidata's cross-referenced APIs.

6 IMPLEMENTATION

Wikxhibit has a frontend web interface, built using JavaScript, HTML, and CSS, and a backend component, built using JavaScript. We implement Wikxhibit as a JavaScript library that queries Wikidata using its SPARQL endpoint and other web APIs using the ShapirJS library. Using Wikidata and Shapir, Wikxhibit creates a virtual data model that links entities across different websites. Wikxhibit queries Wikidata, using its SPARQL endpoint, and returns the result as an entity or a list of entities, each with properties and values. Wikxhibit uses the entities' external identifiers to fetch data about those entities from other web APIs using the appropriate functions provided by ShapirJS. Wikxhibit integrates with Mavo through a plugin that adds Wikxhibit as a new Mavo backend. It takes care of translating HTML attributes to Wikxhibit function calls that allow HTML authors to query Wikidata and other web APIs and present their data.

7 EVALUATION

In our evaluation, we examined whether Wikxhibit could be learned and applied by users, including non-programmers, to create various applications that present data from Wikidata and other data sources in a short amount of time.

It has already been shown [31] that non-programmers with basic HTML knowledge can quickly create Mavo applications to manipulate data that is defined and stored locally. The improvement provided by Wikxhibit is to empower those same types of authors to create the same kinds of applications but to present data from Wikidata and other web APIs.

In order to understand both the usability and flexibility of Wikxhibit, we designed two user studies. For a first *structured* study, we authored two Wikidata presentations and then gave users a series of Wikxhibit authoring tasks that gradually evolved those presentations into complete ones. We then asked users to create a presentation of specific Wikidata data from scratch. This study focused on learnability and usability. For a second *freestyle* study, we asked participants to create presentations of their preferred Wikidata data. This study focused on whether Wikxhibit's capabilities were sufficient to create presentations envisioned by users.

7.1 Preparation

7.1.1 Procedure. We recruited 12 Wikidata users (mean age 45.75, SD 7.1; 84% male, 8% female, 8% other) by publishing a call to participate in the structured study on the Wikidata mailing list, forum, and social media. And for the freestyle study, we conducted a workshop at a Wikidata conference, where we asked Wikidata users to create their applications. Of the participants, 8 identified as beginner or intermediate in HTML, and 4 as advanced or expert. When they were asked about programming languages, 7 described themselves as beginners or worse in any programming language, while 5 considered themselves intermediate or better. Of these participants, 7 participated in the structured study, and 5 participated in the freestyle study.

Before the structure study, we interviewed each user, asking them how they are using Wikidata, what type of presentations they would like to create, and the challenges that prevented them from doing so; we then gave them a tutorial on Wikxhibit. Before the freestyle study and during the workshop, we presented Wikxhibit, then gave all users a tutorial on Wikxhibit.

7.1.2 Participants Background. The participants we recruited for this study were all users of Wikidata; some have used Wikidata more than others. 3 of the participants were Wikidata editors: an editor of music data, an editor of German non-profit organizations data, and a communications manager in Wikidata. These Wikidata editors are beginners or worse in any programming language. In addition to editors, 3 other participants were professors: a digital humanity professor, a data journalism professor, and a software engineering and project management professor. Moreover, 3 participants are developers who have built tools using Wikidata, and their work is featured on the Wikidata tools page [36]. The rest of the participants are using Wikidata for their personal use.

7.2 The Structured Study

For the structured study, participants were assigned three Wikidata presentations or applications, two of which we created for the study. Our participants were given static HTML and CSS mockups of these applications and were asked to carry out a series of tasks by editing the HTML. These tasks tested their ability to use different aspects of Wikxhibit, as shown in Figure 7. The first application allows users to browse information about different tech companies: the company structure (its parent organization and subsidiaries), their products, etc. The other application presents information about US presidents: their family background, education, positions they have held, videos of their speeches, and posts about them on social media (e.g., Reddit). This application combines data from Wikidata and other websites (YouTube and Reddit). The third application was one

that we asked participants to build from scratch. This application displays information about botanical gardens in France.

We provided tasks to the user one at a time, letting them complete one before revealing the next. Participants were asked to speak aloud about their thoughts and confusion as they worked.

7.2.1 Study Tasks. In the case of the tech company application, users had 6 tasks to complete, while for the US president application, users had 5 tasks. The tasks increased in difficulty in order to challenge the users. Figure 7 shows the tasks we assigned the participants and the solution to these tasks. We only show the part of the HTML that participants have to write to perform the tasks. The tasks test different aspects of how Wikxhibit displays data from Wikidata and other websites: properties with primitive values (e.g., App#1/Task#1), properties with object values, those that link to other entities (e.g., App#1/Task#5), properties with multiple values (e.g., App#1/Task#4), and properties of objects from websites other than Wikidata (e.g., App#2/Task#3). In addition to displaying different types of properties from Wikidata and other sites, we tested how users will be able to query Wikidata using different sets of properties and values (e.g., App#3/Task#1).

7.2.2 Result. After going through the tutorial, 6 users went on to complete all the tasks for their three applications with no failures, and 1 user had no failures but had to leave before finishing the last application. The 6 users who completed all tasks successfully took, on average, 14 minutes (Tech Company, 6 tasks), 9 minutes (US President, 5 tasks), and 5 minutes (Botanical Gardens, build an app from scratch) to build the entire application. Some tasks were easier for participants to carry out than others. For instance, all participants were quickly able to display properties with primitive values with an average time of 1 minute or less. For properties that link to other entities, such as getting information about the company's co-founders (App#1/Task#4), participants were able to perform these types of tasks with an average time of 2-3 minutes. For the more challenging task that asked participants to query Wikidata (App#1/Task#6 and App#3/Task#1), participants spent 5 minutes on average performing these tasks.

7.2.3 Challenges and Suggested Improvements to Wikxhibit. During the study sessions, we observed some common challenges and limitations of Wikxhibit. This section presents some of the main challenges our participants faced during the study.

Finding relevant Wikidata properties. Users need to use entity and property names as defined by Wikidata. They need to look up these properties on the Wikidata website. The easiest way to find relevant properties is to look for an example entity page on Wikidata and check its properties. For example, in App#3, most participants looked for a French botanical garden page in Wikidata (e.g., `Jardin des plantes`(Q730948)) and checked the properties available for this garden. From there, they were able to find the `instance of` with value `botanical garden` to use it to query Wikidata. Not all participants did that, so it was challenging for them to know which property to use to get a list of botanical gardens from Wikidata. So, to help users find relevant information from Wikidata, we created a property and entity search widget with auto-complete.

Inconsistency with property names. In Wikxhibit, we use two different property syntaxes: One for querying Wikidata and another for displaying the values for these properties. To query Wikidata using Wikxhibit, we use hyphens to separate the words in property names (`instance-of`). However, if we want to display the value of property `instance of` on the page, we would use `instanceOf`. Using two different syntaxes to write properties is an obvious limitation of Wikxhibit. We do it this way because Shapir maps the properties from Wikidata's cross-referenced APIs to Schema.org. And Schema.org does not use hyphens in its property names. So, we wanted Wikidata properties to be consistent with Schema.org's properties. We can not use this syntax to query Wikidata, where we use hyphens instead because Mavo does not distinguish between capital and small letters. Wikxhibit needs a way to separate the words in property names to query Wikidata. It was confusing for participants to learn the two different syntaxes at first, but they quickly got used to it after the first application.

Showing multiple values. If a property returns a list of values, users have to use the `mv-multiple` attribute to show the list of values (e.g., App#1/Task#3). If the user does not add this attribute, Mavo will only show one of the values. Some participants forgot to add the `mv-multiple` attribute to these properties, and only when they saw that Wikxhibit/Mavo displayed one value did they remember to add the `mv-multiple` attribute to show all the values. One improvement to Mavo would be for it to detect that a property has a single or multiple values automatically, without the user explicitly indicating that.

Feedback. Wikidata entity and property names are case-sensitive. When users use these entities to query Wikidata in Wikxhibit, they need to use them as Wikidata defines them. For example, in App#3/Task#1, participants were asked to query Wikidata to get a list of botanical gardens from France, using the following properties and values: `instance-of="botanical garden"` and `country="France"`. Note that the value of `instance of` uses small letters, and the value of `country` starts with a capital letter. This is because these two values are defined that way in Wikidata. Currently, Wikxhibit does not return any error messages telling the user what could have gone wrong. A necessary improvement to Wikxhibit would be to show error messages that help guide users when they make mistakes, which we are working on adding to Wikxhibit.

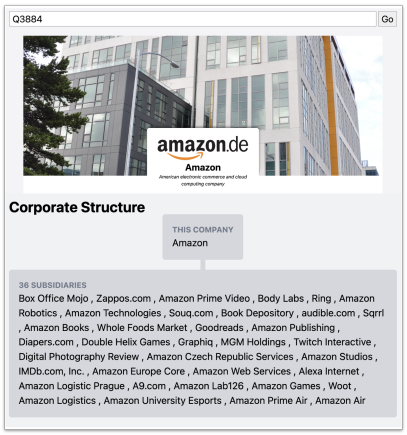
Documentation. Wikxhibit provides documentation with examples and demos. One participant suggested we add our Wikxhibit documentation on a wiki page "*The documentation could be moved on a wiki, so the community can help to expand and improve it*". This would really help the Wikidata community share their applications with the community and build on top of each other's work. We have already created a Wikxhibit page on Wikidata[1], and we plan on adding our detailed documentation to it.

7.3 The Freestyle Study

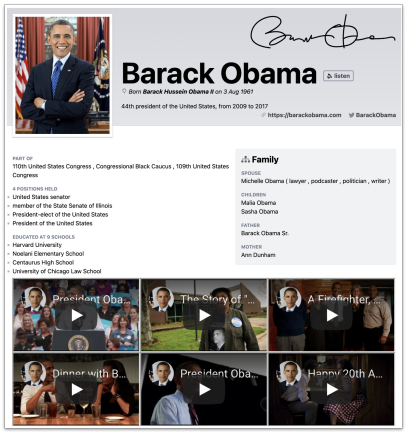
Our second freestyle user study was part of a workshop we conducted during a Wikidata conference. The conference's primary goal is to provide a space to bring together anyone interested in using Wikidata's data. More specifically, to bring together Wikidata

[1] https://www.wikidata.org/wiki/Wikidata:Wikxhibit

```
<!-- Task#6 show a list of 5 companies that are part of big tech in the United States of America-->
<div mv-app mv-source="wikxhibit" mv-source-service="wikidata"
  mv-source-part-of="Big Tech" mv-source-country="United States of America"
  mv-source-numberOfItems="5">
    <!-- Task#1 Show description-->
    <p property="description"></p>
    <!-- Task#2 Show inception date-->
    <time property="inception"></time>
    ...
    <!-- Task#3 Show the types of industry-->
    <div property="industry" mv-multiple>
        <p property="label"></p>
    </div>
    <!-- Task#4 Show the co-founders names and images-->
    <div property="foundedBy" mv-multiple>
        <img property="image" />
        <p property="label"></p>
    </div>
    <!-- Task#5 Show the total equity of the parent organization-->
    <div property="parentOrganization">
        <p property="totalEquity"></p>
    </div>
</div>
```




```
<div mv-app mv-source="wikxhibit" mv-source-service="wikidata" mv-source-id="[presidentId]">
    ...
    <!-- Task#1 Show the father's occupation-->
    <div property="father">
        <div property="occupation">
            <p property="label"></p>
        </div>
    </div>
    <!-- Task#2 Show the mother's religion-->
    <div property="mother">
        <div property="religionOrWorldView">
            <p property="label"></p>
        </div>
    </div>
    <!-- Task#3 Show videos from Obama's YouTube Channel -->
    <div property="YouTubeChannel">
        <div property="videos" mv-multiple>
            <iframe src="[embedUrl]"></iframe>
        </div>
    </div>
    <!-- Task#4 Show Obama's Reddit posts -->
    <div property="RedditUsername">
        <p property="text"></p>
    </div>
    <!-- Task#5 Replace Obama with Trump -->
</div>
```




```
<!-- Task#1 Show a list of 15 botanical gardens in France -->
<div mv-app mv-source="wikxhibit" mv-source-service="wikidata"
  mv-source-instance-of="botanical garden"
  mv-source-country="France"
  mv-source-numberOfItems="15">

  <!-- Task#2 For each garden, show its label, description, image,
       and location-->
    <div property="items" mv-multiple>
        <h1 property="label"></h1>
        <p property="description"></p>
        <img property="image" width="10%"/>
        <div property="locatedInTheAdministrativeTerritorialEntity">
            <h5 property="label"></h5>
        </div>
    </div>

</div>
```

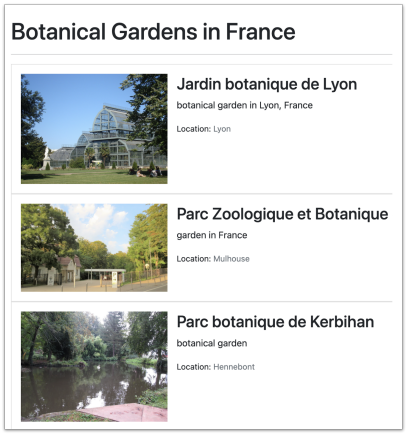


Figure 7: Wikxhibit user study tasks with their solutions. Participants were assigned 3 presentations (right) with tasks, in-lined as comments, and the code (left) is what participants added to perform the given tasks

users, data editors, and developers who want to build applications on top of Wikidata's data to understand better what they are building and their users' needs and wishes regarding Wikidata's data. Each group of users can hear the other's perspective on using Wikidata's data.

This conference was a perfect event to present Wikxhibit to the Wikidata community, get them to use it, and get their feedback. During the workshop, we presented Wikxhibit, then showed a demo of how to create an application that uses Wikxhibit and its cross-referenced APIs. Then, we asked participants to create their own applications within 30 minutes. Participants had access to the Wikxhibit documentation and demos and were allowed to ask us questions about Wikxhibit. Around 30 people attended the workshop, and 5 participants shared their applications with us during the workshop and gave us their feedback. We conducted this study to test whether Wikxhibit's capabilities were sufficient to create presentations envisioned by users.

7.3.1 Result. The Wikxhibit website provides several demos that users can fork and edit to create their applications. 4 of the 5 participants forked some of our demos, and 1 participant created their presentation from scratch. 3/5 participants created their Wikidata presentations in less than 15 minutes; the rest finished them within 30 minutes.

One participant created a page about the movie "The Big Lebowski". The page displays general information about this movie from Wikidata (the director, cast members, etc.). In addition to data from Wikidata, the page displayed movie ratings from IMDb, images from The Movie Database, movie clips from YouTube, and news articles about the movie from NY Times, integrating data from different websites seamlessly using only HTML. This participant used one of our demos and customized it to show their favorite movie.

Another participant created a presentation that showed a list of the superior courts of California. This participant also started their application using one of our demos. They used `mv-source-part-of= "Superior Courts of California"` to query Wikidata and display information about the superior courts of California.

The third participant created a page that shows the political parties based on the country specified by the user using an input field. The application uses `mv-source-instance-of="political party"` and `mv-source-country="Peru"` to query Wikidata and display the `label`, `description`, `official website`, and `political ideology`, etc. for each political party.

The fourth participant recreated an application they had already built by writing code. They wanted to test if they could create that same application using Wikxhibit more efficiently. The application shows information about comic strips that are part of xkcd [38], which is a webcomic of romance, sarcasm, math, and language created by Randall Munroe. The participant was pleasantly surprised that they could recreate this application using Wikxhibit in about 10 minutes. In contrast, they spent an hour building that same application in JavaScript.

The last participant used our art gallery demo to show paintings of their favorite painter. We added some of the applications created by our participants to the Wikxhibit website.

7.4 Participant Feedback

In a post study survey, we asked participants to rate how easy they found the Wikxhibit syntax with the tasks assigned. Participants answered all questions with a five-point Likert scale, from 1 (very easy to use) to 5 (very difficult to use). The average ratings were 2 and 3 for programmers and non-programmers respectively.

7.4.1 Pros and Cons of Using Wikxhibit. We asked participants what they liked and did not like about using Wikxhibit. Participants found Wikxhibit to be easy to use. Wikxhibit hides the complexity of querying Wikidata and other data sources and presents its data using simple HTML syntax. One participant said"*Querying Wikidata is a superpower, and to be able to do more than just query Wikidata and display the results on a beautiful webpage is simply wonderful*", and another said, "*compared to the complexity of the application built it was surprisingly simple.*" More specifically, they liked the immediate results that Wikxhibit presents, integrating data from Wikidata and other websites, where one said "*very intuitive and fast once you get the hang of the core principles, very rewarding seeing the results immediately*".

We noticed two main challenges. First, participants familiar with Wikidata entities/properties found it uncommon to use the human-readable names of the entities/properties "*As a Wikidata editor, it was a little exotic thinking in terms of labels rather than Q and P numbers*" The other challenge was in finding suitable properties to display. Given that Wikidata provides more than 9 thousand properties, users have to go to Wikidata and search for the properties they want to show for a given entity. One said "*something that would be great would be a property and entity search widget and/or autocomplete*". Based on these two challenges, we made the following improvements to Wikxhibit: To support both the simplicity of using the human-readable names of properties and the robustness of using their identifiers, we extend Wikxhibit to support using the identifiers in querying Wikidata. To help users find relevant properties, we created a property/entity search widget with auto-complete. We connect this widget with a code generator that creates a simple application based on the properties a user chooses to query Wikidata.

7.4.2 Wikxhibit in the Wild. We will be working closely with some of the participants to support their real-world use cases of Wikxhibit. Our participants indicated that Wikxhibit could be useful for various purposes: educational, enriching current Wikidata applications with data from other websites, exploring Wikidata linked data, supporting specific types of Wikidata entities, and being able to share quick demos of Wikidata presentations with friends and colleagues.

Educational purposes. Professor participants expressed interest in using Wikxhibit to help their students with their class projects. A Digital Humanity professor said, "*In my class, students build digital exhibitions with tools like Omeka or Wax. This takes time, due to installations, etc.*". Omeka [20] and Wax [21] are libraries for producing and publishing media-rich exhibitions with minimal computing principles. The exhibition in [28] is an example of one that the Digital Humanity students are expected to create. This exhibition allows users to browse an author from Wikidata and display a collection of their photographs and manuscripts. This

exhibition combines data from Wikidata with other sites like VAIF to get more information about the authors.

Creating such an exhibition requires an advanced technical background. But with Wikxhibit, it is possible to create these exhibitions only by authoring HTML and CSS. Students in the Digital Humanity class come from a Social Science/Humanities background and do not necessarily have the programming experience needed to create such exhibitions. But, those students learn how to use HTML and CSS as part of the class, making Wikxhibit a perfect tool for their projects. Wikidata provides the VAIF ID for authors; Wikxhibit uses this ID and gets the authors data using the VAIF API, making it easy for students to combine the VAIF data with Wikidata.

Enriching current Wikidata tools with data from other websites. Two participants who have built Wikidata tools expressed their desire to integrate data from other websites into their applications. One participant is building Scholia [22], a tool that handles scientific bibliographic information through Wikidata, and currently working on combining data from hi-knowledge.org with the data from Wikidata. This participant indicated that making this data integration easy will simplify the work of enriching Scholia's Wikidata with data from other sites.

Another participant built Conzept [8], a topic-exploration tool based on Wikipedia, Wikidata, Open Library, YouTube, and other data sources. The participant is interested in enriching their tool with more data sources, such as archive.org. Wikxhibit makes this possible by using Wikidata's `Internet Archive ID` to fetch relevant data from the Internet Archive API.

Exploring Wikidata linked data. Another usage of Wikxhibit that some participants expressed is to visualize the relationship between Wikidata entities, where one participant said, "*this framework can help me easily understand the relationship between different Wikidata items by visualizing it and presenting it in a page*". There are more than 99 million connected entities or items in Wikidata. Wikxhibit provides an easy way to visualize these connections and make sense of them.

Supporting specific types of Wikidata entities. Our participants, who manage and edit specific types of Wikidata entities, such as music and non-governmental organizations (NGOs), each have a particular use case in mind with how they want to use Wikxhibit.

The music editor wants to create an application that will show a timeline of different types of music from Wikidata and information about them from MusicBrainz. The NGOs Wikidata editor had to hire a developer to create a website that displays all the NGOs in Germany [17] from Wikidata. This editor created the HTML and CSS, but the developer wrote the code to fetch and present the data from Wikidata onto the page. The editor said, "*I wish I had Wikxhibit at the time to use it to build this website myself*". The editor is now working on using Wikxhibit to create a page for each NGO to present information about it from Wikidata. The editor will only need to create one page that would work with any NGO entity in Wikidata.

8 DISCUSSION

Accessing and integrating data from multiple websites has become essential to modern applications. But it is also a significant obstacle. Past efforts, such as the Semantic Web, hold out the vision of representing all that data in a common simple model to make it easier to build such cross-web applications. But there has not been enough uptake, because the websites that hold much of the data do not see enough value in joining the effort.

Instead, we propose an ecosystem that produces a common model of web data without demanding the assistance of the sites that hold the data. We make it possible for end users to create applications that link and integrate data from multiple websites using HTML and with no additional programming. Wikxhibit combines 4 components. Shapir uses a declarative API description language to simplify access to distinct web APIs and uses Schema.org to normalize the different site schemas. To this we add the main contribution of this work: using Wikidata to provide the crosslinking information needed to unify entities. We then use Mavo to generate rich interactive presentations of the resulting unified information.

This work is a capstone project; it relies on the prior innovations of Mavo, Shapir, and Wikidata. Our contribution is to show how these prior innovations can be integrated to create a whole greater than the sum of the parts.

8.1 Wikidata versus Schema.org

Schema.org provides a vocabulary for structured data markup on the Web, which can be found on more than 30% of web pages. Schema.org is structurally much simpler than Wikidata to ease adoption. A challenge for applications using Schema.org is aggregating data about a given entity from different websites. Schema.org defines standardized properties and types, but the identifiers for the individual entities were not standardized. And because Schema.org does not provide entity identifiers, it is encouraging the use of Wikidata as a common entity base for the target of the `schema:sameAs` relation [33]. The `schema:sameAs` is allows web publishers to define a URL of a reference Web page that unambiguously indicates the item's identity (e.g., the URL of the item's Wikipedia entity).

In a slightly ironic twist, the schema used by Wikidata to represent its own information about entities is completely different from the Schema.org schema that Shapir uses to represent the information it fetches about those entities from *all other* websites. Thus, while Shapir allows schematically uniform access to, e.g., all the album information on Spotify and Apple Music, the information that Wikidata holds about albums is schematical distinct. Thus, the most natural approach for a Wikxhibit user is either to use *only* Wikidata information, or to *ignore* the Wikidata information (aside from its entity unification) and use only the schematically uniform information fetched from other websites. The obvious way to overcome this limitation is to use Shapir to write a Schema.org wrapper for Wikidata. But this is a significant challenge, because Wikidata is essentially intended to represent *all conceivable entities*, which requires a much broader API than those typically presented by vertical websites.

There are small efforts that tried to map Schema.org to the Wikidata model. One is YAGO 4 [24], a knowledge base that is based on Wikidata entities. The top-level classes and properties come from Schema.org, and the lower-level classes are a selection of Wikidata classes. But unfortunately, YAGO only covers 198 properties out of 9,840 properties provided by Wikidata. Wikidata also provides a

SPARQL query that maps its properties to Schema.org properties, but similar to YAGO, it is only mapping 156 of Wikidata properties to Schema.org.

One approach we consider promising for would be to use Shapir to map Wikidata to Schema.org *piecewise*. For example, if a user is interested in getting a list of movies from Wikidata, they can use Shapir to map the movie properties from Wikidata to Schema.org/Movie properties. Then, the user will be able to create applications that integrate and present movies from Wikidata and IMDb, for example.

8.2 Data Unification Challenges

Wikxhibit builds on and strengthens the benefits of Shapir. Shapir performs *type unification*, mapping the data coming from distinct websites to common Schema.org types. This allows users to substitute different sources of data without changing their application and even lets them work with mixed collections of data from multiple sources all under the same schema. However, Shapir does no *entity unification*. An application built with Shapir may present a collection of albums combined from Apple Music and Spotify data. But, the collection will contain duplicate entities, where Apple and Spotify both provide information that will be treated as *distinct* entities by Shapir. Wikxhibit provides the additional information needed to merge information about the same entity coming from different sources.

But this stronger merger creates a new problem: *resolving inconsistencies*. Although Shapir ensures that the Apple and Spotify Album descriptions share the same schema, we cannot guarantee that the two APIs provide consistent data. They may disagree about an album name or track ordering, for example. Coming up with automatic resolutions of these inconsistencies is an important problem for future research. At present, Wikxhibit skirts the problem by keeping the information from different APIs distinct, even when under the same schema. A user working with an album entity in Wikxhibit can use the `SpotifyAlbum` property to access the data from Spotify (represented in the standard Album schema) and the `AppleMusicAlbum` property to access the data from Apple (represented under the same schema). If they wish, they can encode special purpose rules for merging that data in *their* application, but we do not attempt a general solution.

8.3 Technical and Design Challenges

Our core technical challenge is designing how to incorporate entity unification into a modular set of components that let non-programmers create rich web applications. Wikxhibit democratizes entity unification. There is tons of literature on entity unification. However, all this literature is directed toward skilled programmers in the database community. We are not aware of any tools that allow end-user programmers to access the power of entity unification, as we provide in Wikxhibit.

We grant that given the proposal to use Wikidata to build applications over resolved entities from multiple sites, actually implementing the connection is not very challenging. We believe our contribution centers on recognizing the new capabilities that would emerge from combining previous components, by determining how to expose the entity resolution capability to users via our extension of Mavo and on our presentation and user evaluation of an architectural vision of how these distinct components (API connection, UI description, and entity resolution) can be combined to support powerful end-user programming.

8.4 Web Components

Wikidata is a valuable resource that has emerged from Wikipedia, but it remains quite separate in some ways. Wikipedia manages massive amounts of human-readable text for human consumption, while Wikidata focuses on managing machine-readable data but seems to give little attention to its general use by people. Wikidata offers relatively basic tools that support querying Wikidata (using the complex SPARQL language) and viewing and editing results in almost raw form. For a few special cases, such as Artists, developers have taken on the work of creating custom presentations. But most Wikidata is not accessible in a human-friendly way.

Wikxhibit demonstrates that it is quite easy to "skin" Wikidata into elegant human-readable presentations. We believe this could significantly increase the usage and value of the data held by Wikidata. Many of the applications built by our study subjects were essentially "result views" for a particular type of Wikidata object. Mavo could be used to define *web components* offering suitable presentations of each type of object in Wikipedia (with built in editing functionality), and users browsing Wikidata could see the information they are looking at wrapped in presentations chosen according to the data type.

8.5 Extension to Read/Write

At present, Wikxhibit provides only *read* access through Wikidata and its cross-referenced web APIs. Mavo applications that fetch data from websites can still manipulate and store that data in Mavo's usual storage locations. Thus, for example, an author could use Wikxhibit to connect to Wikidata to query and present paintings while managing those paintings that they store locally. Enabling Mavo to push edits back to websites requires changes to Mavo's implementation of its core storage model, which we hope to pursue in the future.

9 CONCLUSION

This paper presents Wikxhibit, a JavaScript library for creating HTML-based data presentations of Wikidata and other data APIs it cross-references. Wikxhibit allows a user to author plain HTML that, with a few new attributes, can dynamically fetch and display any of the data that can be accessed through Wikidata or its cross-referenced APIs. Wikxhibit uses Wikidata as the bridge to connect all the cross-referenced APIs, allowing users to aggregate data from different sites simultaneously. We integrate Wikxhibit with Mavo, an HTML language extension for describing web applications declaratively, to empower plain-HTML authors to create presentations of Wikidata. Our evaluation showed that even non-programmers could create data-based applications in just 5 minutes.

ACKNOWLEDGMENTS

We gratefully thank Lydia Pintscher and Denny Vrandečić from Wikidata for their valuable feedback throughout the project.

REFERENCES

[1] Tarfah Alrashed, Jumana Almahmoud, Amy X Zhang, and David R Karger. 2020. ScrAPIr: Making Web Data APIs Accessible to End Users. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–12.

[2] Tarfah Alrashed, Lea Verou, and David R Karger. 2021. Shapir: Standardizing and Democratizing Access to Web APIs. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 1282–1304.

[3] Tim Berners-Lee, James Hendler, Ora Lassila, et al. 2001. The semantic web. *Scientific american* 284, 5 (2001), 28–37.

[4] Jill Cao, Kyle Rector, Thomas H Park, Scott D Fleming, Margaret Burnett, and Susan Wiedenbeck. 2010. A debugging perspective on end-user mashup programming. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 149–156.

[5] Kerry Shih-Ping Chang and Brad A Myers. 2017. Gneiss: spreadsheet programming using structured web service data. *Journal of Visual Languages & Computing* 39 (2017), 41–50.

[6] Kerry Shih-Ping Chang, Brad A Myers, Gene M Cahill, Soumya Simanta, Edwin Morris, and Grace Lewis. 2013. A plug-in architecture for connecting to new data sources on mobile devices. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE, 51–58.

[7] codepen. 2022. Chris Coyier and Alex Vazquez. Retrieved April 1, 2022 from https://codepen.io

[8] Conzept. 2022. A modern topic-exploration tool based on Wikipedia, Wikidata, Open Library, GBIF, YouTube and other information sources. Retrieved April 3, 2022 from https://conze.pt

[9] Marilena Daquino, Ivan Heibi, Silvio Peroni, and David Shotton. 2020. Creating RESTful APIs over SPARQL endpoints using RAMOSE. *Semantic Web* Preprint (2020), 1–19.

[10] Giusy Di Lorenzo, Hakim Hacid, Hye-young Paik, and Boualem Benatallah. 2009. Data integration in mashups. *ACM Sigmod Record* 38, 1 (2009), 59–66.

[11] Robert J Ennals and Minos N Garofalakis. 2007. MashMaker: mashups for the masses. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 1116–1118.

[12] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: evolution of structured data on the web. *Commun. ACM* 59, 2 (2016), 44–51.

[13] Björn Hartmann, Leslie Wu, Kevin Collins, and Scott R Klemmer. 2007. Programming by a sample: rapidly creating web applications with d. mix. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 241–250.

[14] Austin Haugen. 2010. The open graph protocol design decisions. In *International Semantic Web Conference*. Springer, 338–338.

[15] Adrian Holovaty. 2005. ChicagoCrime. org. *Available at http* (2005).

[16] David F Huynh, Robert C Miller, and David R Karger. 2007. Potluck: Data mash-up tool for casual users. In *The Semantic Web*. Springer, 239–252.

[17] Jona Hölderle. 2022. Die größten Nonprofit-Organisationen Deutschlands. Retrieved April 1, 2022 from https://sozialmarketing.de/die-groessten-nonprofits-deutschlands

[18] James Lin, Jeffrey Wong, Jeffrey Nichols, Allen Cypher, and Tessa A Lau. 2009. End-user programming of mashups with vegemite. In *Proceedings of the 14th international conference on Intelligent user interfaces*. ACM, 97–106.

[19] Pasquale Lisena, Albert Meroño-Peñuela, Tobias Kuhn, and Raphaël Troncy. 2019. Easy web API development with SPARQL transformer. In *International semantic web conference*. Springer, 454–470.

[20] minicomp. 2022. Omeka. Retrieved April 1, 2022 from https://omeka.org

[21] minicomp. 2022. Wax. Retrieved April 1, 2022 from https://github.com/minicomp/wax

[22] Finn Årup Nielsen, Daniel Mietchen, and Egon Willighagen. 2017. Scholia and scientometrics with Wikidata. In *Scientometrics 2017*. 237–259. https://doi.org/10.1007/978-3-319-70407-4_36

[23] Node-RED. 2013. Retrieved August 15, 2019 from https://nodered.org

[24] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. 2020. Yago 4: A reason-able knowledge base. In *European Semantic Web Conference*. Springer, 583–596.

[25] Mark Pruett. 2007. *Yahoo! pipes*. O'Reilly.

[26] Markus Schröder, Jörn Hees, Ansgar Bernardi, Daniel Ewert, Peter Klotz, and Steffen Stadtmüller. 2018. Simplified sparql rest api. In *European Semantic Web Conference*. Springer, 40–45.

[27] Swagger. 2021. Swagger Client. Retrieved Marcth 29, 2021 from https://github.com/swagger-api/swagger-js

[28] Celio Hernández Tornero. 2022. Navega por la colección. Retrieved April 1, 2022 from https://celioht.github.io/damaboba/collection

[29] Max Van Kleek, Daniel A Smith, Heather S Packer, Jim Skinner, and Nigel R Shadbolt. 2013. Carpé data: supporting serendipitous data integration in personal information management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2339–2348.

[30] Lea Verou, Tarfah Alrashed, and David Karger. 2018. Extending a Reactive Expression Language with Data Update Actions for End-User Application Authoring. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 379–387.

[31] Lea Verou, Amy X Zhang, and David R Karger. 2016. Mavo: creating interactive data-driven web applications by authoring HTML. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 483–496.

[32] Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*. 1063–1064.

[33] Denny Vrandečić. 2022. Wikidata:Schema.org. Retrieved April 3, 2022 from https://www.wikidata.org/wiki/Wikidata:Schema.org

[34] Wikidata. 2022. Wikidata Query Builder. Retrieved March 11, 2022 from https://query.wikidata.org/querybuilder

[35] Wikidata. 2022. Wikidata Query Service. Retrieved March 11, 2022 from https://query.wikidata.org

[36] Wikidata. 2022. Wikidata:Tools/Visualize data. Retrieved March 11, 2022 from https://www.wikidata.org/wiki/Wikidata:Tools/Visualize_data

[37] Jeffrey Wong and Jason I Hong. 2007. Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1435–1444.

[38] xkcd. 2022. A webcomic of romance, sarcasm, math, and language. Retrieved April 3, 2022 from https://xkcd.com